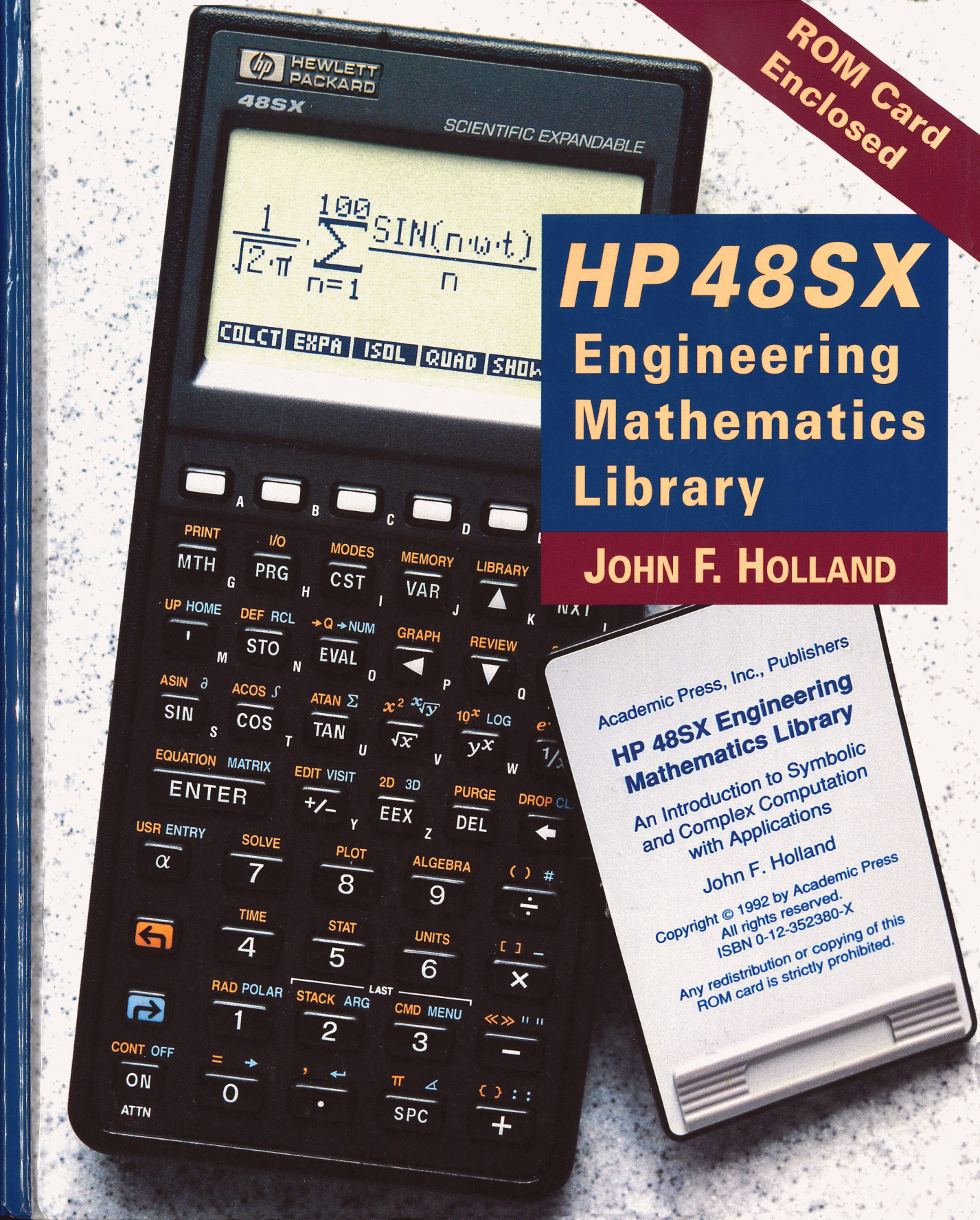
ROM Card Enclosed
HEWLETT PACKARD
48SX
SCIENTIFIC EXPANDABLE
HP 48SX
Engineering Mathematics Library
JOHN F. HOLLAND
Academic Press, Inc., Publishers
HP 48SX Engineering Mathematics Library
An Introduction to Symbolic and Complex Computation with Applications
John F. Holland
Copyright © 1992 by Academic Press
All rights reserved.
ISBN 0-12-352380-X
Any redistribution or copying of this ROM card is strictly prohibited.

HP 48SX Engineering Mathematics Library

An Introduction to Symbolic and Complex Computation with Applications

HP 48SX Engineering Mathematics Library

An Introduction to Symbolic and Complex Computation with Applications

John F. Holland

ACADEMIC PRESS, INC.
Harcourt Brace Jovanovich, Publishers

Boston San Diego New York
London Sydney Tokyo Toronto

LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY

The author and the publisher have used their best efforts in creating this book and software package. The software, this manual, the content of this mathematics library, and any examples contained herein are provided "as is" and are subject to change without notice. The author and publisher make no warranty of any kind, express or implied, with regard to this software and manual, including, but not limited to, implied warranties of merchantibility and fitness for any purpose. The author and publisher shall not be liable for any error or for incidental or consequential damages in connection with the furnishing, performance, or use of this software, manual, and examples presented herein. The hardware ROM enclosed with this manual containing the HP 48 mathematics library is warranted as discussed in Appendix B.

This book is printed on acid-free paper. ∞

ACADEMIC PRESS, INC.
1250 Sixth Avenue, San Diego, CA 92101-4311

United Kingdom Edition published by
ACADEMIC PRESS LIMITED
24–28 Oval Road, London NW1 7DX

ISBN 0-12-352380-X

Printed in the United States of America

92 93 94 95 BP 9 8 7 6 5 4 3 2 1

To my beloved wife Maria,
whose love, patience, and understanding
have helped me to finish this project

GREEK ALPHABET

A α	alpha	N ν	nu
B β	beta	Ξ ξ	xi
Γ γ	gamma	O ο	omicron
Δ δ	delta	Π π	pi
E ε	epsilon	P ρ	rho
Z ζ	zeta	Σ σ	sigma
H η	eta	T τ	tau
Θ θ	theta	Y υ	upsilon
I ι	iota	Φ ϕ	phi
K κ	kappa	X χ	chi
Λ λ	lambda	Ψ ψ	psi
M μ	mu	Ω ω	omega

VARIANTS

ε	epsilon	ϱ	rho
ϑ	theta	φ	phi

TABLE OF CONTENTS

Part 2: Basic Symbolic Computation and Matrix Algebra

Part 5: Complex Data Analysis and Signal Processing Tools

Appendices and Indices

TABLE OF CONTENTS SUMMARY

PREFACE TO

MATHLIB

INTRODUCTION

The ultimate objective of mathematics is to compute a number or a collection of numbers. Most mathematics textbooks and handbooks provide the equations, but do not provide a simple, inexpensive means for evaluation. In this age of personal and supercomputers, the hardware is available for computation, but the software currently available for implementing the computation has several major shortcomings:

- Since the operating systems available for most computers are not object-oriented and were thus never really designed for computation, the numerous collections of mathematics and engineering programs available in various languages are difficult to use because the programs require significant special user data setup and compilation. There is no immediate execution capability such as found on the HP 48, and the operating system, while understanding the difference between a real number and a string, has no idea how to multiply a real number times a complex array. You cannot just push the × key.

- Some of these limitations have been overcome in the past decade by the introduction of large mathematics software packages, which provide, in essence, an operating system on top of the one supplied with the computer which has some object-oriented computing capability. However, the affordable ones are generally very incomplete, requiring the user to purchase several incompatible packages, and the complete ones are generally very expensive, requiring also a very expensive computer with many megabytes of memory and disk space.

- Only with very expensive laptop PCs having many megabytes of memory and disk space are these mathematics packages in any sense portable.

The HP 48 Engineering Mathematics Library (MATHLIB) provides capabilities similar to these large expensive packages, but is much easier to use. In addition, both the hardware and software are less expensive by a factor of 10, are completely portable, and optionally, can be both programmed and executed from either a PC or a MAC.

OBJECT-ORIENTED COMPUTATION

The HP 48 is an object-oriented computer containing a custom CPU and a custom operating system that are designed for computation. The HP 48 supports 32 object types and does know how to multiply a real number times a complex array. It provides immediate execution in the sense that if you put the real number and a collection of complex numbers delimited with brackets [ ] to denote a complex vector on the stack and push ×, the complex product is immediately computed. The user does not even have to store the real number or complex array as variables in memory.

MATHLIB adds three additional object types to the HP 48:

- *Symbolic arrays*, which can be used to represent vector and matrix functions for which a complete set of algebra and calculus operations is defined.
- *Polynomial lists*, which provide very fast polynomial algebra and calculus.
- *Root vectors* corresponding to polynomial lists, except for a scale factor, which provide very fast inverse Laplace and z transform symbolic solutions to differential and difference equations, residue integration, and filter design calculations.

All of the HP and MATHLIB 1,110 user-programmable commands are object-oriented and immediately executable. Arrays do not have to be dimensioned.

PROGRAMMABLE COMMANDS AND MENUS

MATHLIB provides programmable commands instead of menu-driven commands for a very good reason. The fields of mathematics and engineering are pervaded by a bewildering collection of redundant parameter definitions, different subsets of which are used by different authors. Each user may prefer his or her own set as opposed to those chosen for MATHLIB. Since all of the MATHLIB commands are algebraic and programmable, it is extremely easy for each user to define his or her own version of each command, which then accepts inputs using the parameters and units of their choice.

In addition, since the HP 48 makes the creation of user–defined menus so easy, the user can very quickly create his or her own menus for his or her own commands that use the units and parameters of his or her choice. Thus, the user can evaluate the most difficult operations in mathematics with the same ease as pushing the × button on a calculator. As explained in Chapter 25, if you put matrix A on the stack and push EASOV, the symbolic matrix $\Phi(t) = e^{At}$ is returned, solving the symbolic matrix differential equation $\Phi'(t) = A\,\Phi(t)$. Similarly, matrix difference equations can be solved with z transforms.

The user can also download data sets from a PC or Macintosh, push the × key from that computer, and upload the result to the computer.

EVALUATE EQUATIONS AS THEY ARE WRITTEN

With MATHLIB, the user does not have to learn (or try to remember) a complicated programming language in order to evaluate equations. For example, consider evaluating the regular Coulomb wave function $F_L(\eta, \rho)$ defined by:

$$F_L(\eta, \rho) = 2^L\, e^{-\pi\eta/2 - i\rho}\, \frac{|\Gamma(L+1+i\eta)|}{\Gamma(2L+2)}\, \rho^{L+1}\, M(L+1-i\eta,\ 2L+2,\ i2\rho),$$

where $\Gamma(z)$ is the Gamma function, and M(a, b, z) is Kummer's confluent hypergeometric function. The following program evaluates $F_L(\eta, \rho)$:

```
« → L η ρ « '2^L×EXP(-π×η/2-i×ρ)×ABS(GAMMA(L+1+i×η))/
GAMMA(2×L+2)×ρ^(L+1)×MABZ(L+1-i×η , 2×L+2 , i×2×ρ)' →NUM » » .
```

Observe that the program looks exactly like the equation and can be stored in a variable FLηρ. To evaluate $F_3(.4, 5)$ you may either place 3, .4, 5 on the stack and push the FLηρ key, or put 'FLηρ(3 , .4, 5)' on the stack and push **EVAL**.

Next, suppose you wish to compute the dot product between two N-dimensional vectors X and Y defined as the sum

$$\mathrm{DOT}(X, Y) = \sum_{k=1}^{N} X_k Y_k \qquad N = \mathrm{SIZE}(X) = \mathrm{SIZE}(Y).$$

Then the corresponding program would be

```
« → X Y « X SIZE EVAL → N 'Σ(k=1,N,X(k)×Y(k))' » » .
```

SCOPE OF MATHLIB

The scope of MATHLIB is very broad in the sense that it provides mathematical capability to support many different fields of study and ranges from very basic to very advanced mathematics. Since the menus, and thus the manual, are built around subject areas, each menu may contain a mixture of basic, intermediate, and advanced material. MATHLIB is also an introduction to symbolic and complex computation since it provides all the tools with applications plus much tutorial material.

Part 1 is the MATHLIB tables of complex functions. It begins with the simpler, more common functions and progresses to the more difficult and esoteric ones. While the advanced student may be interested in all the mathematical subtleties of branch cuts and Riemann surfaces, the less experienced student may simply wish to evaluate them as a great improvement over interpolating some table. Numerous functions are provided, even though many of the functions are interrelated so that the user does not require advanced mathematical knowledge of these relationships in order to evaluate specific functions of interest. In addition, numerous parameter conversion commands are provided so the user again can evaluate without extensive knowledge of all the parameter relationships. The user can thus plot an elliptic function as easily as a sine function. The slower evaluating functions can be tabulated during lunch.

Part 1 also provides numerous commands for approximating data and functions.

Part 2 provides extensive symbolic algebra, calculus, and linear algebra tools. The symbolic algebra and calculus commands enhance and expand the extensive symbolic capability already built into the HP 48. A complete set of fast polynomial operations is provided in addition to an extensive set of symbolic algebraic operations commands and programs. An advanced algebra and calculus example dealing with hundreds of terms is also provided to demonstrate big problem techniques.

The Part 2 complex linear algebra commands range from eight linear solvers and simple array commands to advanced matrix decomposition theory and algorithms. The modern techniques of unitary decomposition are covered in detail with access to the internal functions for eigen analysis and singular value decomposition. Numerous commands for manipulating the parts of matrices are also provided.

Part 3 provides many of the more common statistical tests for means, variances, and distributions. Contingency table analysis, correlation and rank correlation tests are also provided in addition to analysis of variance and covariance. Over 50 forward and inverse probability distributions are provided in addition to multi–server queueing distributions and related commands. Linear regression is covered in Part 1.

Part 4 begins by extending the linear algebra operations of Part 2 to symbolic matrix functions. This provides the mathematical framework for the advanced differential equation solution programs, in addition to providing extensive vector calculus tools.

A command for order four Runge–Kutta numerical solutions to scalar, vector, and matrix nth order differential equations is then presented with several examples.

Part 4 then covers the Heaviside partial fraction expansion and residue integration topics with applications to inverse Laplace transforms. With this foundation, symbolic solution of linear differential equations is addressed with a number of examples. Then the symbolic solution of state space differential equations is covered. Finally, examples of vector line and volume integration are given. While not explicitly addressed, the commands of Part 4 can be extended to general tensor analysis.

The first chapter of Part 5 is devoted to complex data and signal processing. It begins with easy topics such as numerical differentiation and integration of data vectors, finding peaks, valleys, minimums, and maximums. Then it moves into more advanced topics such as discrete Fourier transforms, Wiener–Levinson solutions, correlation, Toeplitz matrix operations, digital spectral analysis, and linear convolution.

The next two chapters cover analog and digital filter design and analysis. Various related subjects, including computing symbolic inverse z transforms and state space controllable, observable, and Jordan canonical forms, are also covered. This material is generally intermediate to advanced subjects in electronic and controls engineering, though today good statistics packages are expected to contain much of this capability.

Part 6 contains 84 basic vector and matrix commands that extend the HP 48 scalar algebraic capability to vectors and matrices. Thus, the user can avoid dealing with the elements of arrays on the stack.

The appendices provide basic reference material in calculus and transform theory. The extensive indices allow the user to easily find the solution to any problem.

WHO WILL FIND IT USEFUL

MATHLIB is designed like a handbook in that it provides a very broad capability, though only a subset of that capability will be used during a given session. A quick scan of the Table of Contents will convince anyone of the enormous power contained in this library. However, it should be remembered that not all of the 1,110 available user-programmable commands involve enormously sophisticated mathematics.

Many of the commands simply provide basic algebra, calculus, differential equation solutions, solutions to linear systems of equations, statistics and probability, and vector analysis for physics applications. A student can use command **IXFRM** to symbolically solve differential and difference equations without understanding the details of complex residue integration. As the user grows in his or her understanding of applied mathematics and engineering, more and more of MATHLIB will become useful.

Unlike so many other software packages, MATHLIB clearly explains exactly what is computed with each command and provides access to the internal commands so the student can experiment with algorithms and learn. The top-level commands are also available so the student has an unlimited collection of correct answers. The manual contains a great deal of tutorial material, and extensive examples are given with each command. The HP 48SX with MATHLIB provides 1,110 programmable commands and thousands of real and complex functions to solve the user's problems.

MATHLIB will be useful in any of the following subject areas:

- Basic and advanced algebra with solutions of equations
- Basic and advanced statistics, probability, and queueing theory
- Basic and advanced differential and integral calculus
- Basic and advanced differential equations
- Basic and advanced linear algebra and matrix decomposition theory
- Basic and advanced vector analysis and analytic geometry
- Basic and advanced engineering mathematics
- Basic and advanced physics, general relativity, and tensor analysis
- Static and dynamic analysis of mechanical systems
- Thermodynamics and heat transfer analysis
- Electricity, magnetism, and electromagnetic propagation
- Electronics and communication theory design and analysis
- Analog and digital IIR and FIR filter design and response analysis
- Linear system theory with control analysis and synthesis
- Spacecraft dynamics and aerospace engineering
- Evaluation of higher transcendental functions and applications
- State space design, analysis, and differential equation solutions
- Applied mathematics, computer science, and numerical analysis
- Mechanical vibrations and the analysis of structures
- Fluid mechanics, hydraulics, and related problems
- Modulation theory, noise, and spectral analysis
- Mathematical approximation of data and functions
- Laplace, Fourier, Hilbert, and z transforms with applications
- Analog and digital signal processing and simulation

MATHLIB will solve a 40 x 40 linear system of equations with iterative refinement of the solution in under 4 minutes. With the powerful symbolic algebra and calculus programs provided, it can perform difficult symbolic integrations, such as

$$\int [\, A + Bt + Ct^3 + \sin(t) \,]^4 \, dt,$$

involving hundreds of terms. It supplies a full set of algebra and calculus commands for symbolic vector and matrix functions, including symbolic matrix inversion and complex matrix residue integration. Symbolic higher transcendental functions with defined derivatives are available, as discussed, in Appendix A.

MATHLIB will be useful to a student from an entry-level algebra, statistics, or calculus course clear through the doctoral level and professional practice. The scope of Chapter 22 on statistics has been limited so that calculus is not required. Similarly, the scope of Chapter 19 has been limited to basic topics in symbolic algebra and calculus. Chapter 20 on linear algebra begins with basic topics, such as solving linear systems of equations, and leaves the advanced matrix decomposition topics for last.

While the HP 48 is obviously not the fastest computer available and is no competition for a large, expensive mathematics package when computing carpet plots, most scientists and engineers rarely require more than a few numbers in a session. The slower HP 48 evaluation speed is easily offset by the log-on and setup time required by large computer packages, which are much more difficult to use. You cannot simply put your problem on the stack and push a key for the answer.

MATHLIB provides very high quality reliable software and makes symbolic and complex computation both understandable and affordable to everyone. Programming is simply keystrokes enclosed within French quotes – there is no complicated computer programming language that must be learned (remembered) to solve large problems.

Over 700 MATHLIB example, test, application, and symbolic function programs have been written. Additional applications software is in development (see Appendix A).

TO THE INSTRUCTOR

The literature is almost void of textbooks to help the student and the professional get off the real number line into the world of complex computation where the *real* scientific and engineering problems are solved. MATHLIB attempts to fill that void, while at the same time providing an inexpensive means for performing the actual symbolic and complex computations on an inexpensive, completely portable, graphics calculator.

A number of basic entry-level texts are becoming available for the HP 48. What the parent wants to hear is that the student does not need yet another software package or plug-in card for each and every college course. MATHLIB provides more symbolic and complex computation than is elsewhere available at any price. Since hundreds of examples are given in this manual, the student can quickly apply the commands to his or her problems and applications.

My four year old boy can run a singular value decomposition (SVD) program, but does he learn anything? A student can run an SVD program on a $20,000 university workstation, but does he or she learn anything? To learn, what the student needs is access to all the little building blocks which are combined to produce an SVD. The modular and object-oriented design of MATHLIB provides access to those internal building blocks, so the student can experiment and learn. Since the top-level commands are also available, the student has an unlimited source of correct examples.

As the list on page xxii implies, MATHLIB is designed to provide symbolic and complex computation across a broad spectrum of applications. It can be used in a large number of college courses and research activities. It can also be used as a secondary textbook which allows the student to put theory into practice. Finally, it can be used as a textbook for basic to advanced courses in symbolic and complex numerical analysis, where the theory is proved by actual computation.

By providing students custom menus containing those specific commands the instructor wishes the students to use, a teacher can focus the students on the relevant methodology, and application areas of interest, in any particular course. Hundreds of example and application programs are available, as discussed in Appendix A.

As technology evolves, symbolic and complex computation grow in importance. Why should every scientist and engineer need also to be an expert on compilers and operating systems? Now students can solve their computational problems without becoming computer scientists. Even difficult solutions are only a push of a menu key away, on an inexpensive object-oriented graphics computer: the HP 48 with MATHLIB.

ACKNOWLEDGMENTS

I wish to thank Dr. Paul O. Scheibe for his detailed, independent complex plane testing of the software and for his comments and suggestions on the manuscript. I also thank Miss Clara F. Manders for her detailed review of the manuscript and checking of the tables. Dr. Howard L. Resnikoff's editorial comments are also appreciated. I finally wish to thank Miss Patty G. Corey for her typing support.

1

OVERVIEW OF MATHEMATICS LIBRARY

INTRODUCTION

The HP 48 Engineering Mathematics Library (MATHLIB) is one of the most powerful and complete mathematics, statistics, linear algebra, and digital signal and data processing packages ever written. *The ROM card included with this manual is the most sophisticated computer chip ever manufactured.* It consists of 715 new HP 48 commands, bringing the total number of calculator commands to 1,110. While the HP 48 does not have the horsepower of a supercomputer, the speed of the Math Library commands is adequate for all student applications, as well as many professional ones.

FEATURES OF THE HP 48 MATH LIBRARY

- One of the largest computer tabulations of complex mathematics functions ever published, over 300, and most have 10-digit accuracy. *No interpolations required and no strange confusing normalizations to remove! Call them from your HP programs or tabulate them for your PC applications.*

- Single-button plotting of all HP 48, MATHLIB, and all real and complex user defined functions stored in the VAR directory. *Linear, semi-logarithmic, and log–log plots are available with titles and labeled axes.*

- User-friendly interface via a special system of 36 menus grouped by application areas. The HP custom menu and custom keyboard features are still available.

- Vector and Matrix Language: *Numerous commands are provided for manipulating arrays without dealing with the elements. Examples include vector plots, swap column, replace row, copy column, insert row, delete column, sort row, as well as numerous vector and matrix scalar commands that perform the same arithmetic operation on each element of an array, such as absolute value and the SIN function, without the user having to put the elements on the stack. Random uniformly and normally distributed arrays are created with MATHLIB commands, including bivariate normally distributed real and complex vectors and matrices.*

- Over 100 statistical operations and tests, including *T tests for means, F tests for variances,* χ^2 *and Kolmogorov–Smirnov tests for distributions, correlation and Spearman rank correlation tests, contingency table analysis, one- and two-way analysis of variance, analysis of covariance, median and mode estimation, histograms, vector and matrix sorting, and multiple regression of data with complex polynomials and functions, plus analysis of variance and computation of singular values.* Mann–Whitney, Wilcoxon signed rank, and Kruskal–Wallis tests.

- Over 50 statistical probability distributions and their inverses. *Confidence intervals are directly computed with the inverse distributions.* Multi-server queueing distributions and related calculations.

- Over 100 data and signal processing operations, including *FFTs, time domain and FFT convolution, deconvolution, correlation, IIR filter convolution, interpolation, decimation, Wiener–Levinson design of least squares filters, 10 windowing functions, and bilinear transforms with frequency warping. Butterworth, Chebyshev, elliptical, Bessel, and both IIR and FIR digital filter design. Zero and pole transformations of lowpass analog and digital designs to the highpass, bandpass, and bandstop cases. Peak and valley analysis, test waveforms, uniform and normal noise sources, analytic signals. Linear and logarithmic scale plotting of complex responses for analog and digital filters. Group delay plots.* Wavelet transforms.

- Taylor series, rational function, 15 polynomial, and very general least squares approximations with analysis of variance and singular values.

- Over 200 vector and matrix commands including *rank, LU, QR, bidiagonal, tridiagonal, upper Hessenberg, singular value, Schur, and Cholesky decompositions with Householder reflections, and Givens rotations. Complete computation of eigenvalues and eigenvectors of complex square matrices as well as the singular values and singular vectors of arbitrary complex matrices.* With the 15-digit HP 48 internal precision, small matrices can be decomposed at 10-digit accuracy.

- Over 50 symbolic vector and matrix commands. Algebra and calculus operations include *dot and cross products, determinants, inverses, gradients, divergences, curls, and Laplacians in eight orthogonal curvilinear coordinate systems.* Evaluation of vector line and volume integrals.

- Evaluation of complex contour integrals, complex residues, and symbolic inverse Laplace and z transforms. Applications to differential and difference equations.

- Both Runge–Kutta numerical and symbolic solutions to scalar, vector, and simultaneous differential equations. Many examples are given.

- Both symbolic and numerical solutions of state space matrix differential and difference equations. *Reduction of state space equations to transfer functions. Conversion between controllable, observable, and Jordan canonical forms.*

- Over 200 algebra operations, including three powerful complex coefficient polynomial root solvers. Techniques for algebra involving hundreds of terms.

- Over 50 data editing, sorting, windowing, clipping, and peak and valley analysis commands. Numerical differentiation and integration of data.

- Detailed command explanations with hundreds of example software applications are provided. Extensive indices for searching and finding commands in library.

- Up to 160 K of RAM still available for user applications and data. Serial port connects to PCs and MACs for programming, execution of commands, and data transfer. *Over 700 application, example, and test programs are available.*

- Except when plotting poles and zeros of transfer functions, all of the ΣDAT commands are available in *parallel* with the Math Library, including linear, logarithmic, exponential, and power data curve fitting.

GETTING STARTED

See the HP 48 owner's manual. The HP 48SX has two ports for installing plug–in cards. The Math Library may be plugged into either port. Be sure that the calculator is turned OFF when inserting or removing the card. After the card is installed, turn the calculator ON, and the library will automatically ATTACH to the HOME directory. Now push LIBRARY and you will see MATH displayed on one of the soft menu keys. Pushing the MATH key displays the first page of the Math Library main menu.

The first page of the main menu is shown below and on the next page. Six commands appear on each page of the menu; there are 120 menu pages. This manual has been carefully written to follow the format shown below where the command definition is given in a table, and explanations and examples are given with the command. All of the Math Library command tables are organized like this one.

MATHLIB MAIN MENU

FUNCTION	COMMAND	INPUTS	OUTPUTS
DISPLAY COMMAND DIRECTORY	**MENUE**	NONE	MATHLIB MENU OF COMMANDS

MENUE (menu evaluate) is a command which computes and displays 36 MATHLIB menus. The names of the 34 command menus appear like VAR directories, though they are neither directories nor commands. Each command menu is a collection of programmable commands grouped by application area. MATHLIB adds 715 new commands (824 programs including internal hidden functions) to the HP 48's 395 commands for a total of 1,110 user commands (programmable operations).

PLOT 1 FUNCTION LABELS	**FPLOT**(FL,δ)	FL	REAL OR COMPLEX PLOT
PLOT 1 FUNCTION WITH LABELS	**FPLT1**(FL,PL,δ)	FL PL	REAL OR COMPLEX PLOT
PLOT 2 FUNCTIONS WITH LABELS (REAL ONLY)	**FPLT2**(FL1,FL2, PL,δ)	FL1 FL2 PL	TWO REAL PLOTS

FL = { { DIRECTORY PATH } 'FUNCTION' INDEPENDENT VARIABLE
PLOT INCREMENT STARTING VALUE NUMBER OF POINTS }

See **CSERS** function in Chapter 17.

MATHLIB MAIN MENU

FUNCTION	COMMAND	INPUTS	OUTPUTS
PL = { (X,Y) "X AXIS TITLE" "Y AXIS TITLE" } where (X,Y) is location where axes cross. See the plot functions in the { PLOT } menu discussed in Chapter 2. For example: { {HOME} '**DNUK**¡(z,.2)' z .4 0 20 } computes the Jacobian elliptic function dn(z, .2) versus z = 0, .4, .8, . . . 7.6 and plots it. For a labeled plot, { (10,.5) "2.5X" "dn(X,.2)" } can be used. δ = 0 for linear scale and δ = 1 for logarithmic scale of independent variable.			
COMPLEX INTEGRATION	**CINTG**(L,U,F,z)	L ∈ ℂ U ∈ ℂ F(z) ∈ ℂ z ∈ ℂ	VALUE
Numerically integrates analytic F(z) from z = L to z = U. See Chapters 4 and 6 for examples.			
KEEP	**KEEP**(n)	n ∈ ℕ OR 0	n STACK OBJECTS
Keeps the first n objects on the stack while deleting all objects above n.			

The **MENUE** command displays 36 menus that allow the user to easily access the 715 MATHLIB commands. There are two top-level menus which display the 34 programmable command menus like VAR directories. When **MENUE** is pushed, 18 menu labels are displayed on 3 menu pages. These are summarized on the next page.

FTNS is also a top-level menu which displays 18 menu labels. It provides access to the 17 function menus summarized on page 7 of this chapter. This application oriented menu system provides a far more user-friendly interface than that afforded by the main menu. As explained in the HP 48 owner's manual, when MATHLIB is attached, all of the 715 commands are global commands just like the 395 HP 48 commands. All of the library commands may be used in user application programs.

MENUE OVERVIEW		
MENU NAME	**MENU DESCRIPTION**	**NUMBER OF COMMANDS**
PLOT	PLOTTING OPERATIONS	12
FTNS	MENUS FOR MATHEMATICS FUNCTIONS	301
NUMB	NUMBER THEORY CALCULATIONS	22
ALGB	SYMBOLIC ALGEBRA OPERATIONS	42
LINAG	LINEAR ALGEBRA OPERATIONS	66
MATR	SPECIAL MATRIX OPERATIONS	58
STAT	STATISTICAL ANALYSIS TOOLS	48
PROB	PROBABILITY DISTRIBUTIONS	30
IPROB	INVERSE PROBABILITY CALCULATIONS	20
BIVN	BIVARIATE NORMAL DISTRIBUTION	7
QUE	MULTI-SERVER QUEUEING THEORY	12
SYMB	SYMBOLIC VECTORS AND MATRICES PLUS DIFFERENTIAL EQUATIONS	57
PROC	DATA PROCESSING OPERATIONS	48
FILTR	FILTER DESIGN OPERATIONS	54
WIND	WINDOW, CLIPPING AND FIR DESIGN	36
VECTR	SPECIAL VECTOR OPERATIONS	24
VSAG	VECTOR SCALAR ALGEBRA	30
MSAG	MATRIX SCALAR ALGEBRA	32

The detailed contents of these 36 menus and most of the HP 48 system menus are given in the menu maps of Appendix H on pages 579 through 586.

FUNCTIONS OVERVIEW { FTNS }		
MENU NAME	**MENU DESCRIPTION**	**NUMBER OF COMMANDS**
TRIG	ADDITIONAL TRIGONOMETRIC FUNCTIONS	12
HYP	ADDITIONAL HYPERBOLIC FUNCTIONS	17
EXPIN	EXPONENTIAL INTEGRAL FUNCTIONS	18
GAMA	GAMMA AND RELATED FUNCTIONS	12
ERROR	ERROR AND RELATED FUNCTIONS	14
BESEL	BESSEL FUNCTIONS OF INTEGER ORDER	20
SBESL	SPHERICAL BESSEL FUNCTIONS	12
ELLIPI	ELLIPTICAL INTEGRALS	25
JACOB	JACOBI ELLIPTICAL FUNCTIONS	36
THETA	THETA AND RELATED FUNCTIONS	42
CHYPR	CONFLUENT HYPERGEOMETRIC FUNCTIONS	18
PCLDR	PARABOLIC CYLINDER FUNCTIONS	12
GHYP	GAUSSIAN HYPERGEOMETRIC FUNCTIONS	6
LGDR	GENERALIZED LEGENDRE FUNCTIONS	6
STRUV	COMPLEX STRUVE FUNCTIONS	6
POLY	ORTHOGONAL POLYNOMIALS	15
MISC	MISCELLANEOUS FUNCTIONS	30
UPDIR	RETURN TO MENUE–LEVEL MENUS	598

ORGANIZATION OF THE MANUAL

This manual is organized around the 34 MATHLIB command menus. In a few cases more than one menu is discussed in a chapter. The same command is included in several menus where applicable for user convenience. Some HP 48 commands are also repeated to save the user typing time.

Each chapter contains a brief overview of the command table that follows. The command table format was chosen so that all the applicable information would be easy to find with the command.

ORGANIZATION OF THE MENUS

The menus begin with the plotting operations in PLOT. These commands are provided to simplify the plotting of functions and data, and to significantly improve speed. They allow the user to plot any HP 48, MATHLIB, or user function with one command. LOG–LOG plots are available.

Pushing FTNS moves into the numerous complex plane mathematics tables. The TRIG and HYP menus provide solutions to wave and diffusion differential equations in rectangular coordinates. These menus contain a full set of trigonometric and hyperbolic functions. These are the most widely known transcendental functions, and half of them are provided with the HP 48.

EXPIN, GAMA, and ERROR provide evaluation of some of the simpler higher transcendental functions throughout the complex plane. EXPIN contains exponential integrals and related functions. GAMA contains complete and incomplete gamma and related functions. ERROR contains the error function, Fresnel integrals, and related functions.

The BESEL and SBESL menus provide differential equation solutions in circular cylindrical coordinates, in addition to evaluating numerous complex integrals. BESEL contains all the Bessel functions of integer order and SBESL all the spherical Bessel functions. Evaluation for complex arguments is provided. Evaluation of Bessel functions with complex order is provided in the CHYPR menu.

ELLIPI, JACOB, and THETA evaluate numerous elliptic integrals and functions. ELLIPI contains all the elliptic integrals with analytic continuation. JACOB contains the numerous Jacobian elliptic and related functions. THETA evaluates the numerous theta functions and their derivatives.

Then we move into the more advanced higher transcendental functions. CHYPR provides evaluation of the complex confluent hypergeometric functions and many of their special cases. Both basic solutions to the confluent hypergeometric equation M(a, b, z) and U(a, b, z) are provided, including the logarithmic case. PCLDR provides the parabolic cylinder functions, which are differential equation solutions in parabolic cylindrical coordinates.

GHYP then provides the general Gaussian hypergeometric function and some special cases. Using these commands, LGDR evaluates the Associated Legendre functions, which are the solutions to differential equations in spherical coordinates. Both $P_v^u(z)$ and $Q_v^u(z)$ are provided for complex argument, order, and degree. STRUV provides the complex Struve functions.

POLY provides numerous orthogonal polynomials. They are available in both symbolic and numerical forms.

The MISC menu is devoted primarily to the approximation of functions. Polynomial, rational, and least squares approximations are provided. Returning to the MENUE menu level with the UPDIR key, NUMB provides the number theoretic functions, as well as Bernoulli and Euler polynomials. This completes the MATHLIB tables of complex functions.

The ALGB menu provides symbolic algebra and calculus, Taylor series, polynomial arithmetic and calculus, and complex root-solving commands. Many of the commands work with polynomial lists instead of the symbolic polynomials since it is much faster. Symbolic hypergeometric functions are also provided, in addition to many special algebraic identity, expansion, and manipulation programs.

The Math Library contains one of the most powerful and complete linear algebra packages ever written. Access is provided to all the internal functions. These commands are provided in the LINAG and MATR menus for numerical arrays and the SYMB menu discussed below for function and symbolic arrays.

The next five menus provide numerous statistical functions and probability distributions. STAT contains many standard statistical tests. PROB and IPROB provide many common probability distributions and their inverses for computing confidence intervals. BIVN contains commands to evaluate all the upper and lower combinations of the bivariate normal probability distribution. The QUE menu contains commands for solving multi-server queueing theory problems. PROB, IPROB, and BIVN contain one of the most complete sets of probability tables ever published.

The SYMB menu provides the capability of symbolic vector and matrix operations. These commands provide the foundation for solving scalar, vector, and matrix differential equations. Commands for both symbolic and numerical solutions to differential equations are provided. SYMB also provides commands for evaluating symbolic inverse Laplace and z transforms using complex residue integration and vector calculus operations such as gradients, divergences, and curls in various coordinate systems. Commands are presented for solving state space differential equations. Examples of vector line and volume integration are also given. The applications of *CHAOS* is currently a hot research topic.

The next three menus provide data and signal processing operations. PROC contains numerous commands for analyzing and processing data. Included are numerical derivatives and integrals, peak and valley analysis, fast Fourier transforms, convolution and correlation, Wiener–Levinson least squares solutions, interpolation, and decimation. FILTR contains commands for design and analysis of both analog and digital filters including elliptic ones. Additional commands are provided for conversion between transfer function and canonical state space representations of linear systems. Combined with the commands in the windowing menu WIND, digital FIR and IIR filters can be designed in the lowpass, highpass, bandpass, and bandstop cases. The computation and study of *WAVELET TRANSFORMS* is very easy with MATHLIB.

The VECTR menu is the vector analog of the matrix menu MATR and provides numerous commands for manipulating the elements of vector arrays.
Vector and matrix scalar arithmetic are provided by the commands in the VSAG and MSAG menus. They extend basic scalar functions such as LN, EXP, and ABS to array objects on the HP 48.

The HP 48 is designed to make creation of custom menus and keyboard definitions very easy. The program « { PADD PSUB PMPY PDVD PDERV PINTG } TMENU » stored in a variable MYMENU brings up a polynomial operations menu whenever MYMENU is evaluated (the MYMENU key pushed). Hence, the user can easily build his or her own menus. See Chapter 15 of the HP 48 owner's manual for more detail.

ASSUMPTIONS

This manual is not intended to be a stand-alone textbook. While Chapter 3 does give a very brief tutorial on complex analysis, as do many other chapters on various subjects, we generally assume that the reader is already familiar with the subject matter. Equations are given in this manual to make clear what each command actually evaluates, but they are not a substitute for a textbook on each subject.

We also assume that the reader carefully studies the operation of each command before using it. Many of the internal Math Library commands have been made accessible to the user for his or her applications. It is prohibitive for every command to test the validity of every input all the time because of ROM space and execution time. While the software makes reasonable checks on the inputs, ultimately the user must read the directions and make sure his or her inputs are valid. Numerous examples are given to make clear what inputs each command requires.

Apart from Appendix H, this manual is not a tutorial on how to operate and program the HP 48. We assume that the user has understood the HP 48 owner's manual.

APPLICATIONS

For the student, the HP 48 Math Library is an unlimited source of correct answers and worked examples related to basic calculus, advanced calculus, differential equations, vector analysis, vector calculus, linear algebra, statistics, signal processing, and numerous engineering problems.

The Math Library not only provides solutions to complicated calculations such as eigenvalues and eigenvectors, but also provides the tools for learning and understanding all the individual steps associated with such calculations. Since the Math Library covers many different areas of mathematics, statistics, engineering, computer science, and digital signal processing, it has broad application in many courses. Many of the recently published textbooks contain homework problems which assume the student has access to a computer and various software packages. While they might provide a SVD program, for example, they do not provide all the little building blocks so you can learn to write your own. Now that capability is available on an inexpensive calculator.

For the professional, the Math Library is the ideal tool for quick analyses and a source of correct answers when testing and debugging PC and mainframe computer programs. Most of the complex higher transcendental functions are already available for use in engineering analyses. While a typical problem using a set of Fortran programs purchased with your textbook could take several hours to set up and compile, the HP 48 with the Math Library could solve the same problem in minutes.

For those who never did care to know the theory or how to program a computer, the answer is only the push of a button away. The Math Library is also one of the most complete tabulations of complex functions and probability distributions ever published, making it an important reference for any technical library.

ARGUMENTS AND STACK DIAGRAMS

The HP 48 is an RPN calculator that also supports algebraic notation. All of the Math Library commands are algebraic, that is, both global and local variables can be used as arguments. The HP 48 command **TRN** is not algebraic. While the program « → M « M TRN » » works, the program « → M 'TRN(M)' » does not. The first program puts matrix M on the stack and then evaluates the Hermitian transpose command **TRN**. The second would symbolically produce the same result if **TRN** were algebraic. The Math Library command **TRNH** is also the Hermitian transpose, and since it is algebraic, « → M 'TRNH(M)' » works as well as « → M « M TRNH » ».

Consequently, the universal stack diagram for all of the Math Library commands is

'FUNCTION(X_N , ... , X_2 , X_1)' EVAL

which is equivalent to placing each argument on the stack and evaluating FUNCTION

$$\left\{\begin{array}{l} N:\ X_N \\ \vdots\quad\ \vdots \\ 2:\ X_2 \\ 1:\ X_1 \end{array}\right. \qquad \text{FUNCTION}$$

Thus, the N arguments of command FUNCTION are put on the stack in the order shown, and the command FUNCTION is evaluated. This is standard RPN and RPL notation and evaluation. This universal stack diagram generally also applies to the algebraic HP 48 commands, though there are a few exceptions such as **XROOT**.

In using algebraic notation, the arguments may be global variables, local variables, and numbers. Nevertheless, we frequently use the shortcut notation FUNCTION([1 2 3 4 5] , 7), where the first argument in this case happens to be a vector, but might also be an array (matrix), a list, or a program. The HP 48 editor will complain endlessly if you actually type this. When you see these notations in this manual, they are to be interpreted one of two ways:

- Evaluate 'FUNCTION(α,7)', where the variable α contains [1 2 3 4 5].
- Put the two arguments on the stack and evaluate FUNCTION.

Both interpretations are correct and are equivalent according to the MATHLIB universal stack diagram. See also the tutorial in Appendix H, which provides some explicit examples of how to evaluate commands and functions on the HP 48, in addition to explaining how the stack works.

COMMAND NAMES

All of the command names in the Math Library use uppercase English characters, plus occasionally other HP 48 symbols. The exception is lowercase p, which evaluates π.

RESERVED VARIABLES AND CHARACTERS

The variable **ROOTX** is used by the internal solver in the Math Library and will appear in the current VAR directory. The complex integration error is stored in IERR when **CINTG** is called. Similarly, the variables ρ**D1**□, ρ**D2**□, and ρ**D3**□ appear during plotting, where □ is the HP 48 character number 164. □□□□ is used and purged (unless **SVBAR**, **MAT**↑, or **MAT**↓ crash due to bad inputs). ¥, HP 48 character number 165, is used for the plot independent variable so the symbol X is available for symbolic algebraics. All uses of ϕ in MATHLIB commands correspond to HP 48 character 216, which is typed as **O** followed by **ETC** (α, blue right arrow, **9**). See page 54 of HP 48 owner's manual. ¡ is HP 48 character 161 (α, blue right arrow, **DEL**).

SYSTEM FLAGS

RADIAN MODE (Flag −17 set and −18 clear) must be set for the Math Library commands to compute correctly. The commands **D→R** and **R→D** are available for conversion between degrees and radians.

WORD SIZE (Flags −5 to −10) should be set to its default value of 64.

NUMERICAL RESULTS (Flag −3) must be clear so symbolic operations can be performed.

USER FLAGS

MATHLIB uses none of the user flags.

CUSTOM MENU AND KEYBOARD FEATURES

All of the custom menu and keyboard features are still available for the user. Collect the commands you need into a custom menu and define the user keys to save yourself typing time. See Appendix H and Chapter 15 of the HP 48 owner's manual.

FUNCTION NOTATION AND DEFINITIONS

The notation and definitions in the Math Library are based on those chosen in *Handbook of Mathematical Functions*, (Abramowitz and Stegun, National Bureau of Standards, Applied Mathematics Series (AMS) 55, June 1964), which is currently published by Dover. We reference this book as AMS 55.

ACCURACY

Estimates of accuracy are provided with the commands. Most apply to accuracy on the real number line and may not hold everywhere in the complex plane. *These are performance estimates and not guarantees.* The user is invited to use references, the complex integration command **CINTG**, and other techniques to check the accuracy of the various MATHLIB approximations. In some cases commands have been provided simply to make comparisons with AMS 55 easier. The MATHLIB functions and matrix commands have also been rigorously tested against several other large computer mathematics packages throughout the complex plane. The symbolic commands and operations have also been extensively tested. Over 700 application, test, and symbolic function programs have been written and tested, and are available. See Appendix A.

AMS 55 depends on Lagrange interpolation to extend its tables. Should you still need it, you may type in the below program where vectors Y and X of the same size N are the dependent and independent data values, N is the number of values used in the interpolation, and x is the value of the independent variable for the interpolation. By interchanging the roles of X and Y, PLINT also does inverse interpolation for x.

```
PLINT: ≪ → Y X x ≪ Y SIZE EVAL X x VSUB VABS 1 FINDN 'Y' OVER
       GET Y Y → N n y C D ≪ -1 'n' STO+ 1 N 1 - FOR m 1 N m
       - FOR j 'X' j GET x - 'X' j m + GET x - DUP2 - 'C' j 1 + GET
       'D' j GET - SWAP / ROT OVER × 'C' j ROT PUT × 'D' j ROT PUT
       NEXT IF n 2 × N m - < THEN 'C' n 1 + GET ELSE 'D' n GET
       -1 'n' STO+ END 'y' STO+ NEXT y ≫ ≫ ≫
```

PLINT([2 3.5 4.9 5.3 5.8] , [1 1.9 2.6 3.2 3.7] , 3) = 5.21339822729

The accuracy of this type of interpolation does not always improve as the number of terms is increased, so use it with caution. For an arbitrary choice of points, there exists a continuous function for which the polynomial approximation error goes to infinity as the number of interpolation points goes to infinity. MATHLIB offers numerous orthogonal approximation techniques, which are much better.

MULTIPLICATION SYMBOL

While the HP 48 uses a 90–degree rotated * to denote multiplication, this manual uses the symbol ×. When the meaning is clear, the multiplication symbol is often omitted, even though you must type it in on the calculator.

STYLE

Having searched through many thousand-page software manuals desperately seeking the bottom line amongst all the words, I have written this manual very differently. A picture is worth a thousand words, and this manual is a collection of pictures documenting the operation of each command. The bottom line and examples are presented with each command. The command table format neatly organizes all this information. When commands appear in more than one menu, generally so does their description. I hope you like my manual.

COMMON PROBLEMS

As explained above, the HP 48 must be in RADIAN MODE. If you notice that the answers are wrong in calculations, first check to see if you hit the wrong key and the HP 48 is no longer in radian mode. RAD should always be displayed in the status area of the display when you use MATHLIB.

Many of the MATHLIB internal programs use symbolic algebraic operations. The numerical results Flag –3 must be clear. Many of the examples also assume that Flag –2 is clear, though the MATHLIB software itself does not care.

On the HP 48, '1+i×2' is a *symbolic object*, and (1,2) is a *complex number*. Matrix commands generally assume a 2 × 2 or larger matrix, and not row or column vectors.

The HP commands **QUOTE** and **APPLY** do not work with Library objects. Consequently, there is no way to make a numerical command such as **GAMMA** work symbolically. Chapter 19 discusses our approach to symbolic arguments and provides examples. Programs are available, as discussed, in Appendix A.

If the program argument to **V1F1**, **V2F1**, **M1F1**, **M2F1**, **S1F1**, or **L1F1** contains a local variable that contains an object that is not a number, it will not properly evaluate. You must instead store the object in a global variable, which can be later purged. See the EASOV program near the end of Chapter 25 for an example.

MENUE is a programmable command and will not execute while the HP 48 is in edit mode. To use the typing aids in **MENUE**, put the program to be edited on the stack and push the LIBRARY MATH **MENUE** keys to get into MATHLIB. Now go into the menu of interest. Then push EDIT, followed by LAST MENU. Now you are back in MATHLIB and can access the typing aids in that menu. Use the same approach with any custom or temporary menus you have defined.

MEMORY MANAGEMENT AND SPEED TIPS

The HP 48 is a LISP-like computer with a garbage collector. The same CPU you are trying to compute with may not be available due to garbage collection and memory management. Always keep unnecessary objects off the stack. When using lists and symbolic matrices, make heavy use of the HP **NEWOB** command to release pointers so the garbage collector can keep up with the garbage creation.

Always evaluate symbolic expressions as soon as possible. Delaying can put too much on the stack and cause the memory management to hang. Execution of the program

« 'EXP(z^2)' z 8 TAYLR »

using the HP 48 command **TAYLR** will expand $EXP(z^2)$ into a Maclaurin series, but it takes 44 minutes. The same computation with MATHLIB commands **COEFL** or **TALR1** only takes 2.6 minutes. The same expansion, but computing 14 terms, only takes MATHLIB 9.3 minutes, whereas **TAYLR** will eventually abort with an "out of memory" error and then spend hours cleaning up the garbage. Evaluating and collecting terms on the fly can make huge speed differences.

The HP 48 garbage collector generally only starts when the memory manager runs out of memory. You may force it to start with the program sequence « MEM DROP ».

Techniques for working with expressions containing hundreds of terms are discussed in Chapter 19. While **EXCO** is provided on the ROM, for many problems it is the slow approach. Expanding $z^{20}(3z^{43}+5z^7)$ with **EXPAN** or **EXCO** will expand z^7 to $z \times z \times z \times z \times z \times z \times z$ and similarly z^{43} as part of the effort, whereas « PARE COLCT » will expand $z^{20}(3z^{43}+5z^7)$ to $z^{20} \times (3 \times z^{43}) + z^{20} \times (5 \times z^7)$ and then collect as $5 \times z^{27} + 3 \times z^{63}$ in seconds.

When editing large equations, you will save a lot of time by pushing EDIT instead of using the equation editor. The equation editor is very nice, but very slow. Pressing any keys while the HP 48 is building an equation can cause strange results. Push **ATTN** when you tire of waiting, and if you lose the equation, try LAST STACK.

2

SPECIAL PLOTTING OPERATIONS

INTRODUCTION

This chapter presents the 12 special plotting operations in the PLOT menu. The PLOT commands are specially designed to make the plotting of functions, filters, windows, FFTs, spectrums, phases, and group delays very easy and fully automatic. For most functions, it is far faster to use **CSERS** to convert them to a vector of values for plotting than it is to present the function itself to the HP 48 plot commands. Input vectors can be HP [ ] arrays or symbolic vectors { } which are lists.

These plot commands take a vector of real or complex values. If V has 64 elements, then the abscissa (X AXIS) ranges from 0 to 63. It is assumed by the software that the values in the vector are of the form f(nT), for n = 0, 1, . . ., N–1, where N is the size of the vector, and T is the sampling period. This assumption works for almost every situation. The values which are plotted are computed via linear interpolation over the values f(nT). Most of the HP 48 plot functions such as scaling, centering, zooming, and zero–crossing determination are still available. However, differentiation is not available since there is no explicit equation. Derivatives can be plotted using **DER1** and **DER2**.

VECTOR PLOTS WITH AND WITHOUT LABELS

PLT1 and **PLT2** provide single and double real plots without labels. **PLTC** provides complex plots without labels. **PLT1L**, **PLT2L**, and **PLTCL** provide the same plots with labels.

FUNCTION PLOTS

FPLOT, **FPLT1**, and **FPLT2** provide direct plotting of all HP 48, user, and MATHLIB functions. They use the MATHLIB command **CSERS** to compute the vector of values for plotting.

LOGARITHMIC AND LOG–LOG PLOTS

The above function plot commands can produce LOG–LOG plots.

OVERLAYING PLOTS

PLT3 provides the capability to add real plots to existing plots.

PLOTTING DERIVATIVES AND INTEGRALS

Examples showing how to plot derivatives and integrals are given.

POLE AND ZERO PLOTS

Plotting poles and zeros of transfer functions is discussed in Chapter 27.

PRINTING PLOTS

Pushing **STO** in the graphics plot window will put the plot on the stack. **PR1** will then print the plot on the HP 48 printer. Alternatively, you may use the HP 48 HOT PRINT, which is discussed on page 605 of the HP 48 owner's manual.

STORING PLOTS FOR LATER VIEWING

Pushing **STO** in the graphics plot window will put the plot on the stack. This graphics plot may be stored in any variable. To view it at another time, recall it to the stack and type **PICT ENTER STO**. Now push **GRAPH**, and it will be displayed.

PLOTTING MENU { PLOT }

FUNCTION	COMMAND	INPUTS	OUTPUTS
PLOT VECTOR	**PLT1**(V)	V ∈ℝ	PLOT V
All of the plot functions provide HP 48 FUNCTION type plots. All of the scaling is automatic. The software even works when you ZOOM or center so that the values off either end of the vector are plotted, though these values themselves are meaningless. However, this gives the user the capability to zoom into any region of the plot of interest. The values of V for **PLT1** must be real, and the size of V must be at least 2. PLT1([1 2]) plots a straight line with a slope of 1.			
PLOT TWO VECTORS	**PLT2**(V1,V2)	V1, V2 ∈ℝ	PLOT V1, V2
PLT2([1 2] , [2 1]) plots two intersecting lines with slopes of 1 and –1. V1 and V2 must have the same size, ≥ 2, and both be real.			
PLOT VECTOR	**PLTC**(V)	V ∈ℂ	PLOT VR, VI
V must be complex, and the size of V must be at least 2. PLTC([(1,2) (2,1)]) plots two intersecting lines with slopes of 1 and –1.			
PLOT VECTOR WITH LABELS	**PLT1L**(V,L)	V ∈ℝ	PLOT V DRAW AXIS
PLOT TWO VECTORS WITH LABELS	**PLT2L**(V1,V2,L)	V1, V2 ∈ℝ	PLOT V1, V2 DRAW AXIS
PLOT VECTOR WITH LABEL	**PLTCL**(V,L)	V ∈ℂ	PLOT VR, VI DRAW AXIS

PLOTTING MENU { PLOT }

FUNCTION	COMMAND	INPUTS	OUTPUTS

V = [VECTOR OF LENGTH ≥ 2] such as the output of **CSERS** discussed in Chapter 17.

L = { (X,Y) "X AXIS TITLE" "Y AXIS TITLE" }

where (X,Y) is the location where the axes cross, and X ∈ [0, SIZE(V)−1]. See Chapter 19 of the HP 48 owner's manual. V1 and V2 must have the same size.

An example plot label list for the plot example dn(z,.2) on the next page is

{ (10,.5) "2.5X" "dn(X.,0.2)" }.

Multiple plots are constructed using linear interpolation of values and automatically scaled based on the first plot, which is V1 in the case of **PLT2** and the real part of V in the case of **PLTC**.

PLOT FUNCTION WITHOUT LABELS	**FPLOT**(FL,δ)	FL δ ∈[0, 1]	REAL OR COMPLEX PLOT

FL = { { DIRECTORY PATH } 'FUNCTION' INDEPENDENT VARIABLE START VALUE FINAL VALUE NUMBER OF POINTS }

All three of the function plot commands in this menu use **CSERS** to compute the real or complex vector of values for plotting. They may be called from any user directory, and the function need not be in the same user directory. HP 48 and MATHLIB functions are all in { HOME }. δ = 0 evaluates the independent variable with a linear scale, while δ = 1 evaluates the independent variable with a base 10 logarithmic scale. START VALUE < FINAL VALUE.

When δ = 1, then 0 < START VALUE < FINAL VALUE.

PLOTTING MENU { PLOT }

FUNCTION	COMMAND	INPUTS	OUTPUTS
FPLOT({ { HOME } 'DNUK$_j$(z,.2)' z 0 7.6 20 } , 0) evaluates the MATHLIB Jacobian elliptic function dn(z, .2), creating the vector of values [dn(0,.2) dn(.4,.2) dn(.8,.2) ... dn(7.6,.2)] using **CSERS** and then calls **PLT1** to produce the plot of the function.			
PLOT ONE FUNCTION WITH LABELS	**FPLT1**(FL,PL,δ)	FL PL δ ∈ [0, 1]	REAL OR COMPLEX PLOT
PLOT TWO FUNCTIONS WITH LABELS (REAL ONLY)	**FPLT2**(FL1,FL2, PL,δ)	FL1 FL2 PL δ ∈ [0, 1]	TWO REAL PLOTS
The above two commands are like **FPLOT**, but they also provide labeled plots.			
LOGARITHMIC PLOT EXAMPLES The following examples provide example semi–logarithmic and LOG–LOG plots: « { { HOME } 'LN(z)' z .1 10 20 } 1 FPLOT » « { { HOME } 'LOG(2^(4×LN(z)))' z .1 10 20 } 1 FPLOT ». Observe that the plots are linear.			

PLOTTING MENU { PLOT }

FUNCTION	COMMAND	INPUTS	OUTPUTS
PLOT VECTOR	**PLT3**(V)	V $\in \mathbb{R}$	PLOT OF V

PLT3 allows the user to add a plot to an existing plot. While all the other plot commands first erase PICT before plotting, **PLT3** plots over any existing plots. Observe that the plot of V is always full scale, and thus the scale of the **PLT3** plot may not be the same as the existing plots.

PLOTTING FROM OTHER USER DIRECTORIES

The following two programs illustrate how to plot user-defined functions located in other user directories. The first program, PLTT2, creates the directory MYDIR1 in the HOME directory and the subdirectory MYDIR2 in the MYDIR1 directory. It can be executed from any user directory and will return to that user directory after creating MYDIR1, including storing the functions COS2 and EXP2 in the MYDIR1 directory and storing the function SIN4 in the MYDIR2 subdirectory. After running PLTT2, then PLTT3 may be executed again from any user directory. PLTT3 provides a demonstration of the **FPLOT**, **FPLT1**, and **FPLT2** commands.

```
PLTT2: « PATH { HOME } EVAL 'MYDIR1' CRDIR { MYDIR1 } EVAL
       'MYDIR2' CRDIR « → α « '2×EXP(i×α)' →NUM » »
       'EXP2' STO « → α '2×COS(α)' » 'COS2' STO { MYDIR2 } EVAL
       « → α '4×SIN(α)' » 'SIN4' STO HOME EVAL »

PLTT3: « { { MYDIR1 } 'EXP2(X)' X 0 7 50 } DUP 0 FPLOT { (25,0)
"X AXIS" "Y AXIS" } 0 FPLT1 { { MYDIR1 MYDIR2 } 'SIN4(X)' X 0 7 50 }
{ { MYDIR1 } 'COS2(X)' X 0 7 50 } { (0,0) "X AXIS" "Y AXIS" } 0 FPLT2 »
```

INTERLEAVING VECTOR DATA VALUES FOR PLOTTING

→**ROW**, →**COL**, →**VTR**, **RNLV**, and **CNLV** are available in the MATR menu.

PLOTTING MENU { PLOT }

FUNCTION	COMMAND	INPUTS	OUTPUTS

OVERLAYING PLOTS

The following program demonstrates overlaying plots with **PLT3**. It may be stored and executed from the same user directory in which you stored PLTT2 and PLTT3 and will overlay the last plot left in PICT after executing PLTT3.

PLTT4: « { { HOME } 'COS(X)' X 0 7 50 } 0 CSERS PLT3 »

Observe that after running PLTT4, full scale is now [–1, 1] when using COORD. Also, if you use ZOOM or one of the other replot options, only the 'COS(X)' function is replotted, since it is now the current plot stored in EQ and the only plot that the HP 48 knows about.

HP ERASE	**ERASE**	NONE	BLANK GRAPH

ERASE clears PICT.

UP DIRECTORY	**UPDIR**	NONE	PARENT MENU

DEGLITCHING DATA PLOTS

Occasionally, the data vector you wish to plot may have some small values (the ones whose shape you wish to see in the plot) and one or more large values which will consume the plot range if you plot the vector as it is. There are a number of remedies. See the commands **DGLIT**, **VSUBS**, **VREPL**, **CLIPB**, **CLIPN**, and **CLIPP** for ideas.

PLOTTING MENU { PLOT }

FUNCTION	COMMAND	INPUTS	OUTPUTS

PLOTTING DERIVATIVES AND INTEGRALS

The commands **DER1** and **DER2** are available for plotting derivatives. The commands **LINT** and **RINT** are available for plotting integrals.

```
DEMS: « "NUMERICAL DERIVATIVES" 4 DISP { { HOME} 'SIN(X)' X .1 0
  100 } 0 CSERS "COMPUTE FIRST DERIV" 3 DISP DUP DER1 10 ×
  SWAP "COMPUTE SECOND DERIV" 3 DISP DER2 100 × { (49,0) "10X"
  "COS(X) & –SIN(X)" } PLT2L »

DEMI: « "NUMERICAL INTEGRATION" 4 DISP { { HOME } 'SIN(X)' X .1 0
  100 } 0 CSERS "COMPUTING INTEGRAL" 3 DISP DUP LINT .1 × 1
  VSUB { (49,0) "10X" "SIN(X) & –COS(X)" } PLT2L »
```

The 1 VSUB in the above program inserts the constant of integration.

REAL PLOTS WITH LABELS

```
DEMY: « "REAL BESSEL Y(0,X)" 4 DISP { { HOME } 'YNOX(0,X)' X .5 .5
  25 } 0 CSERS { (12,0) "2X" "Y0(X)" } PLT1L »
```

COMPLEX PLOTS WITH LABELS

```
DEMD: « "COMPLEX SPH HANKEL" 4 DISP { { HOME } 'SH1Z(1,X)' X .5
  .5 20 } 0 CSERS { (9.5,0) "2X" "h1(1,X)=j1 + iy1" } PLTCL »
```

The HP 48 has a very general plotting and graphics capability. If you experiment with the many options and find that the commands in this menu no longer work, try the **RESET** command discussed on page 323 of the HP 48 owner's manual.

3

TRIGONOMETRIC AND HYPERBOLIC FUNCTIONS

INTRODUCTION

This chapter presents the 29 commands in the trigonometric and hyperbolic functions menus. It also offers a short tutorial on the basic concepts associated with complex analysis to clarify notation. These concepts are used in this and the following chapters. A program for the Kronecker delta function is also given in the TRIG menu.

Throughout this manual we make use of set theory notation. Define $\mathbb{N}$ to be the set of positive integers that are called the natural numbers. The symbols $x \in \mathbb{N}$ say that variable x must be a natural number such as 1, 2, ... Similarly, $x \in \mathbb{I}$ says that x is an integer variable, that is, a variable that takes on integer values. The integers are the natural numbers, their negatives, and zero. We also use $x \in \mathbb{R}$ and $z \in \mathbb{C}$ to denote that x is a real variable and z is a complex variable.

COMPLEX VARIABLES

The concept of complex variables arises out of the observation that no real number can be the solution of $z^2 + 1 = 0$. In fact, the solution must be something like $z = \sqrt{-1}$. The concept of $\sqrt{-1}$ results in a paradox. While

$$(\sqrt{-1})^2 = -1, \qquad (\sqrt{-1})^2 = \sqrt{-1}\sqrt{-1} = \sqrt{(-1)(-1)} = \sqrt{1} = 1.$$

The paradox is resolved by the introduction of the notation i for $\sqrt{-1}$ and properly choosing the positive root. Observing that $(\pm 2)^2 = 4$, on the HP 48, $\sqrt{4} = 2$, which is the positive root. This subject is addressed in more depth at the end of the chapter.

In 1779 Euler adopted the notation $i^2 = -1$ and the vector space approach to describing complex numbers as $x + iy = z$, where x and y are real. We denote this by $x, y \in \mathbf{R}$ and $z \in \mathbf{C}$. The *complex conjugate* of z is $z^* = x - iy$. All the properties of complex numbers then follow from the properties of the real numbers x, y and the definition of i. If you put lowercase i on the stack of the HP 48 and push →**NUM**, it will evaluate to (0,1), representing a complex number with 0 for its real part and 1 for its imaginary part.

TRANSCENDENTAL FUNCTIONS

Transcendental functions are ones which cannot be produced by a finite number of the ordinary algebraic operations: addition, subtraction, multiplication, and division. Thus, they must be represented as infinite series. Simple ones like sin(z) are often called elementary functions since they are widely known. Less known ones generally get the title of higher transcendental functions or special functions, even though they are just functions.

Consider the simplest of all differential equations: $\frac{df(z)}{dz} = f(z)$, where f(z), is some unknown solution. If we assume f(z) is some power series, then

$$f(z) = \sum_{n=0}^{N} a_n z^n \qquad \frac{df(z)}{dz} = \sum_{n=1}^{N} n\, a_n z^{n-1} = \sum_{n=0}^{N-1} (n+1)\, a_{n+1} z^n.$$

Consequently, we can write

$$f(z) - \frac{df(z)}{dz} = \sum_{n=0}^{N-1} [a_n - (n+1)\, a_{n+1}]\, z^n + a_N z^N.$$

Choosing $a_n = 1/n!$, we discover that the above sum exactly equals zero for all N, and since $\lim_{N\to\infty} \frac{z^N}{N!} = 0$ for all $|z| < \infty$, we have proved that the solution f(z) is a transcendental function.

COMPLEX EXPONENTIAL FUNCTION

One can show that the function f(z) has several interesting properties, such as $f(z + w) = f(z)\, f(w)$. f(1) is known as Euler's constant e. If you put lowercase e on the stack and push →**NUM**, the HP 48 will evaluate it for you. f(z) is the complex exponential function **EXP** on your calculator. Since e^z with $z = x + iy$ equals

$$\begin{aligned} e^{x+iy} &= e^x \sum_{n=0}^{\infty} \frac{(iy)^n}{n!} \\ &= e^x \left[\sum_{n=0}^{\infty} \frac{i^{2n} y^{2n}}{(2n)!} + \sum_{n=0}^{\infty} \frac{i^{2n+1} y^{2n+1}}{(2n+1)!} \right] \\ &= e^x \left[\sum_{n=0}^{\infty} (-1)^n \frac{y^{2n}}{(2n)!} + i \sum_{n=0}^{\infty} (-1)^n \frac{y^{2n+1}}{(2n+1)!} \right] \\ &= e^x \left[\cos(y) + i \sin(y) \right], \end{aligned}$$

we have an interpretation for e^z in terms of real functions with real arguments. The above equation is called Euler's formula in honor of its discoverer.

COMPLEX NATURAL LOGARITHM

Suppose $f(z) = e^z$. Then what is the inverse function f^{-1} such that $f^{-1}(f(z)) = z$? Obviously it's the **LN** command on the HP 48, but observe that since $\cos(y) = \cos(y + i2n\pi)$ and $\sin(y) = \sin(y + i2n\pi)$ for any integer $n \in \mathbb{I}$, in general Ln is not unique. The $n = 0$ solution is called the principal value and is what **LN** actually computes. This principal value is denoted by ln in AMS 55. In general, $\mathrm{Ln}(e^z) = z \pm i2n\pi = x + iy \pm i2n\pi$. Since for real $x \in \mathbb{R}$, Ln(x) is only defined for $x > 0$, in the complex plane Ln(z) is defined everywhere except at the origin and the negative real axis.

Logarithms with other bases are defined by $\mathrm{Log}_a(z) = \mathrm{LN}(z)/\mathrm{LN}(a)$. Similarly, $z^a = \mathrm{EXP}(a\ \mathrm{Ln}(z)) = e^{a\ \mathrm{Ln}(z)}$. The base 10 commands **LOG** and **ALOG** are examples.

COMPLEX CIRCULAR FUNCTIONS

By the same steps we used to express e^{iy} in terms of cos y and sin y, we have for complex z

$$e^{iz} = \cos z + i \sin z, \qquad e^{-iz} = \cos z - i \sin z.$$

Rearranging these two equations gives definitions for complex cos z and sin z.

$$\cos z = \tfrac{1}{2}\left[e^{iz} + e^{-iz}\right], \qquad \sin z = \tfrac{1}{2i}\left[e^{iz} - e^{-iz}\right].$$

Now we can define all of the other trigonometric functions.

$$\tan z = \frac{\sin z}{\cos z}, \quad \cot z = \frac{\cos z}{\sin z}, \quad \csc z = \frac{1}{\sin z}, \quad \sec z = \frac{1}{\cos z}.$$

SIN, **COS**, and **TAN** are available on the HP 48 keyboard, while **COT**, **CSC**, and **SEC** are in the TRIG menu. The sin and cos functions are the solutions of the differential equation

$$\frac{d^2 f(z)}{dz^2} + f(z) = 0.$$

COMPLEX HYPERBOLIC FUNCTIONS

Two other useful sums of exponentials define the complex hyperbolic functions cosh z and sinh z.

$$\cosh z = \tfrac{1}{2}\left[e^z + e^{-z}\right], \qquad \sinh z = \tfrac{1}{2}\left[e^z - e^{-z}\right].$$

Now we can define all of the other hyperbolic functions.

$$\tanh z = \frac{\sinh z}{\cosh z}, \quad \coth z = \frac{\cosh z}{\sinh z}, \quad \operatorname{csch} z = \frac{1}{\sinh z}, \quad \operatorname{sech} z = \frac{1}{\cosh z}.$$

HP 48 commands **SINH**, **COSH**, and **TANH** and Math Library commands **COTH**, **CSCH**, and **SECH** are all available in the HYP menu. With these definitions we have the relations

$$\begin{aligned} \cos z &= \cosh iz, & \tan z &= -i \tanh iz, & \csc z &= i \operatorname{csch} iz, \\ \sin z &= -i \sinh iz, & \cot z &= i \coth iz, & \sec z &= \operatorname{sech} iz. \end{aligned}$$

The sinh and cosh functions are the solutions of the differential equation

$$\frac{d^2 f(z)}{dz^2} - f(z) = 0.$$

CIRCULAR FUNCTIONS IN TERMS OF REAL AND IMAGINARY PARTS

With the above definitions, explicit formulas can be written for the real and imaginary parts of the trigonometric functions. Let z = x + iy. Then

$$\cos z = \cos x \cosh y - i \sin x \sinh y, \qquad \sin z = \sin x \cosh y + i \cos x \sinh y,$$

$$\tan z = \frac{\sin 2x + i \sinh 2y}{\cos 2x + \cosh 2y}, \qquad \cot z = \frac{\sin 2x - i \sinh 2y}{\cosh 2y - \cos 2x}.$$

HYPERBOLIC FUNCTIONS IN TERMS OF REAL AND IMAGINARY PARTS

With the above definitions, explicit formulas can be written for the real and imaginary parts of the hyperbolic functions. Let $z = x + iy$. Then

$$\cosh z = \cosh x \cos y + i \sinh x \sin y, \qquad \sinh z = \sinh x \cos y + i \cosh x \sin y,$$

$$\tanh z = \frac{\sinh 2x + i \sin 2y}{\cosh 2x + \cos 2y}, \qquad \coth z = \frac{\sinh 2x - i \sin 2y}{\cosh 2x - \cos 2y}.$$

INVERSE CIRCULAR FUNCTIONS

From the equations for the real and imaginary parts we observe that cos z and sin z are periodic with respect to x, but single–valued with respect to y. Consequently,

$$\begin{aligned} \cos w &= z, & \operatorname{Arccos} z &= \pm \arccos z + 2k\pi = \pm w + 2k\pi, \\ \sin w &= z, & \operatorname{Arcsin} z &= (-1)^k \arcsin z + k\pi = (-1)^k w + k\pi, \\ \tan w &= z, & \operatorname{Arctan} z &= \arctan z + k\pi = w + k\pi \qquad z^2 \neq -1, \end{aligned}$$

where $k \in \mathbb{I}$, arccos, arcsin, and arctan are the principal values and are the HP 48 keyboard commands **ACOS**, **ASIN**, and **ATAN**, respectively. The other three inverse circular functions are

$$\operatorname{arccot} z = \arctan 1/z, \qquad \operatorname{arccsc} z = \arcsin 1/z, \qquad \operatorname{arcsec} z = \arccos 1/z,$$

which are the commands **ACOT**, **ACSC**, and **ASEC**, respectively, in the TRIG menu.

INVERSE HYPERBOLIC FUNCTIONS

From the equations for the real and imaginary parts we observe that cosh z and sinh z are periodic with respect to y, but single–valued with respect to x. This is just the reverse of the circular functions. In Chapters 9 through 11 we will present the elliptic functions that are doubly periodic, that is, periodic in both x and y. The single periodicity in y results in the definitions

$$\begin{aligned}
\cosh w = z, &\quad \text{Arccosh } z = \pm\text{arccosh } z + 2k\pi i = \pm w + 2k\pi i,\\
\sinh w = z, &\quad \text{Arcsinh } z = (-1)^k \text{ arcsinh } z + k\pi i = (-1)^k w + k\pi i,\\
\tanh w = z, &\quad \text{Arctanh } z = \text{arctanh } z + k\pi i = w + k\pi i,
\end{aligned}$$

where $k \in \mathbb{I}$, arccosh, arcsinh, and arctanh are the principal values and are the HP 48 keyboard commands **ACOSH**, **ASINH**, and **ATANH**, respectively. The other three inverse hyperbolic functions are

$$\text{arccoth } z = \text{arctanh } 1/z, \qquad \text{arccsch } z = \text{arcsinh } 1/z, \qquad \text{arcsech } z = \text{arccosh } 1/z,$$

which are the commands **ACOTH**, **ACSCH**, and **ASECH**, respectively, in the HYP menu.

SOME USEFUL TRIGONOMETRIC IDENTITIES

$$\sin z = \cos\left(\frac{\pi}{2} - z\right) \qquad \cos z = \sin\left(\frac{\pi}{2} - z\right)$$

$$\tan z = \cot\left(\frac{\pi}{2} - z\right) \qquad \cot z = \tan\left(\frac{\pi}{2} - z\right)$$

$$\begin{aligned}
\sin(A \pm B) &= \sin A \cos B \pm \cos A \sin B\\
\cos(A \pm B) &= \cos A \cos B \mp \sin A \sin B\\
\tan(A \pm B) &= \frac{\tan A \pm \tan B}{1 \mp \tan A \tan B}\\
\cot(A \pm B) &= \frac{\cot A \cot B \mp 1}{\cot A \pm \cot B}
\end{aligned}$$

$$\begin{aligned}
2 \cos A \cos B &= \cos (A - B) + \cos (A + B)\\
2 \sin A \sin B &= \cos (A - B) - \cos (A + B)\\
2 \sin A \cos B &= \sin (A - B) + \sin (A + B)\\
2 \cos^2 A &= 1 + \cos 2A\\
2 \sin^2 A &= 1 - \cos 2A\\
\sin^2 A + \cos^2 A &= 1
\end{aligned}$$

$$\begin{aligned}
\sin 2A &= 2 \sin A \cos A\\
\cos 2A &= \cos^2 A - \sin^2 A\\
\tan 2A &= \frac{2 \tan A}{1 - \tan^2 A}
\end{aligned}$$

$$\sin\frac{A}{2} = \sqrt{\frac{1 - \cos A}{2}}$$

$$\cos\frac{A}{2} = \sqrt{\frac{1 + \cos A}{2}}$$

$$\tan\frac{A}{2} = \frac{\sin A}{1 + \cos A} = \frac{1 - \cos A}{\sin A}$$

$$\sin A \pm \sin B = 2 \sin\frac{A \pm B}{2} \cos\frac{A \mp B}{2}$$

$$\cos A + \cos B = 2 \cos\frac{A + B}{2} \cos\frac{A - B}{2}$$

$$\cos A - \cos B = -2 \sin\frac{A + B}{2} \sin\frac{A - B}{2}$$

$$\tan A \pm \tan B = \frac{\sin(A \pm B)}{\cos A \cos B}$$

$$\begin{aligned}
\arcsin a \pm \arcsin b &= \arcsin\left(a\sqrt{1 - b^2} \pm b\sqrt{1 - a^2}\right) \\
&= \arccos\left(\sqrt{1 - a^2}\sqrt{1 - b^2} \mp ab\right) \\
\arccos a \pm \arccos b &= \arccos\left(ab \mp \sqrt{1 - a^2}\sqrt{1 - b^2}\right) \\
&= \arcsin\left(b\sqrt{1 - a^2} \pm a\sqrt{1 - b^2}\right) \\
\arctan a \pm \arctan b &= \arctan\frac{a \pm b}{1 \mp ab}
\end{aligned}$$

SOME USEFUL HYPERBOLIC IDENTITIES

$$\begin{aligned}
\sinh(A \pm B) &= \sinh A \cosh B \pm \cosh A \sinh B \\
\cosh(A \pm B) &= \cosh A \cosh B \pm \sinh A \sinh B \\
\tanh(A \pm B) &= \frac{\tanh A \pm \tanh B}{1 \pm \tanh A \tanh B} \\
\coth(A \pm B) &= \frac{\coth A \coth B \pm 1}{\coth A \pm \coth B}
\end{aligned}$$

$$2\cosh A\cosh B = \cosh(A+B) + \cosh(A-B)$$
$$2\sinh A\sinh B = \cosh(A+B) - \cosh(A-B)$$
$$2\sinh A\cosh B = \sinh(A+B) + \sinh(A-B)$$
$$2\cosh^2 A = 1 + \cosh 2A$$
$$2\sinh^2 A = \cosh 2A - 1$$
$$\cosh^2 A - \sinh^2 A = 1$$

$$\sinh 2A = 2\sinh A\cosh A$$
$$\cosh 2A = \cosh^2 A + \sinh^2 A$$
$$\tanh 2A = \frac{2\tanh A}{1+\tanh^2 A}$$

$$\sinh\frac{A}{2} = \sqrt{\frac{\cosh A - 1}{2}}$$
$$\cosh\frac{A}{2} = \sqrt{\frac{\cosh A + 1}{2}}$$
$$\tanh\frac{A}{2} = \frac{\sinh A}{1+\cosh A} = \frac{\cosh A - 1}{\sinh A}$$

$$\sinh A \pm \sinh B = 2\sinh\frac{A \pm B}{2}\cosh\frac{A \mp B}{2}$$
$$\cosh A + \cosh B = 2\cosh\frac{A+B}{2}\cosh\frac{A-B}{2}$$
$$\cosh A - \cosh B = 2\sinh\frac{A+B}{2}\sinh\frac{A-B}{2}$$
$$\tanh A \pm \tanh B = \frac{\sinh(A \pm B)}{\cosh A\cosh B}$$

$$\operatorname{arcsinh} a \pm \operatorname{arcsinh} b = \operatorname{arcsinh}\left(a\sqrt{1+b^2} \pm b\sqrt{1+a^2}\right)$$
$$= \operatorname{arccosh}\left(\sqrt{1+a^2}\sqrt{1+b^2} \pm ab\right)$$
$$\operatorname{arccosh} a \pm \operatorname{arccosh} b = \operatorname{arccosh}\left(ab \pm \sqrt{a^2-1}\sqrt{b^2-1}\right)$$
$$= \operatorname{arcsinh}\left(b\sqrt{a^2-1} \pm a\sqrt{b^2-1}\right)$$
$$\operatorname{arctanh} a \pm \operatorname{arctanh} b = \operatorname{arctanh}\frac{a \pm b}{1 \pm ab}$$

COMPLEX ANALYSIS

A very brief, high-level summary of a few of the fundamental concepts associated with calculus on the complex plane is given below for those who have never been introduced to it. It is designed to help those who have never had the joy of complex analysis courses grasp and use the powerful capability built into the HP 48 and the MATHLIB. A tutorial review of real calculus is given in Appendices C and D.

COMPLEX LIMITS

The notions of continuity and differentiability are more complicated in the complex plane. Whereas in the real case the concept of limits dealt with a point on a line, now we must deal with a point in a two-dimensional vector space for which there is an infinite number of lines passing through it, and the limit for each line could be different. Let $z = x + iy$ and $f(z) = u(x, y) + i\, v(x, y)$ where $u(x, y)$ and $v(x, y)$ are real. Then if $\Delta z = \Delta x + \Delta y$, we have that $f(z + \Delta z) = u(x + \Delta x, y + \Delta y) + i\, v(x + \Delta x, y + \Delta y)$, and the order in which Δx and Δy go to zero can matter in the complex plane. For example, let us evaluate the limits $LN(0 + i\Delta y)$, $LN(0 - i\Delta y)$, and $LN(\Delta x)$ as Δx and Δy go to zero. The first two cases evaluate the limit on a line coinciding with the imaginary axis, but the second case corresponds to a 180-degree rotation of the line in the first case. The third case corresponds to a line coinciding with the positive real axis. We have

```
LN( (0,1.E-200) )  = (-460.517018599,1.57079632679),
LN( (0,-1.E-200) ) = (-460.517018599,-1.57079632679),
LN( (1.E-200,0) )  = (-460.517018599,0).
```

The limits are all different. The HP 48 is great for evaluating limits numerically.

A limit at a point in the complex plane only exists when the limit on all lines passing through that point have the same value. More rigorously, the limit of $f(z)$ as z goes to z_0 exists only if for some complex number w_0 (the limit) corresponding to each positive number ε, there exists a positive number δ such that $|f(z) - w_0| < \varepsilon$ whenever $|z - z_0| < \delta$ and $z \neq z_0$. When, in addition, $f(z_0) = w_0$, then $f(z)$ is continuous at z_0.

COMPLEX DERIVATIVES

Derivatives in the complex plane are defined in terms of the partial derivatives $\partial u/\partial x$, $\partial u/\partial y$, $\partial v/\partial x$, and $\partial v/\partial y$. Based upon the above limit considerations, the derivative of f exists if and only if the *Cauchy–Riemann conditions* are satisfied.

$$\frac{\partial u\ (x,\ y)}{\partial x} = \frac{\partial v\ (x,\ y)}{\partial y} \quad \text{and} \quad \frac{\partial u\ (x,\ y)}{\partial y} = -\frac{\partial v\ (x,\ y)}{\partial x}$$

Consequently, the derivative of f equals

$$\frac{df\ (z)}{dz} = \frac{\partial u}{\partial x} + i\,\frac{\partial v}{\partial x} = \frac{\partial v}{\partial y} - i\,\frac{\partial u}{\partial y}.$$

A function f(z) is *analytic* at z_0 if its derivative exists at every point z in the neighborhood of z_0. f(z) is analytic in a domain if it is analytic at every point in that domain. An *entire* function is one which is analytic throughout the entire complex z plane. If two functions are analytic in a domain D, then their sum, difference, and product are analytic in D. Their quotient is also analytic, except at points where the divisor equals zero. An analytic function of an analytic function is analytic.

e^z is an entire function that equals its derivative. Ln(z) is analytic with derivative 1/z everywhere except the origin and the negative real axis. It follows that sin z, cos z, sinh z, and cosh z are entire, and the other circular and hyperbolic functions are analytic except at those points where they are not defined because the divisor goes to zero (e.g., tan z is not defined at $z = \pi/2$ because $\cos \pi/2 = 0$). All polynomials are entire functions. The nice thing about analyticity is that the calculus rules for complex functions of a complex variable follow the same rules as those of a real function of a real variable. Thus, d sin z/dz = cos z.

$$\frac{d\ \sin z}{dz} = \frac{\partial[\sin x\ \cosh y]}{\partial x} + i\,\frac{\partial[\cos x\ \sinh y]}{\partial x} = \cos x\ \cosh y - i\ \sin x\ \sinh y = \cos z,$$

and using the second half of the derivative formula

$$\frac{d\ \sin z}{dz} = \frac{\partial[\cos x\ \sinh y]}{\partial y} - i\,\frac{\partial[\sin x\ \cosh y]}{\partial y} = \cos x\ \cosh y - i\ \sin x\ \sinh y = \cos z.$$

COMPLEX INTEGRALS

The integral of a real function with respect to a real variable was easy to understand because it was simply the area under the curve. However, the integral from z_1 to z_2 of f(z) is a far more complicated concept. First of all, there is an infinite number of paths between z_1 and z_2. For example, we might go from z_1 over to z_3 and loop 50 times before heading over to z_2. For this reason, complex integrals are called *contour integrals*, and it is often necessary to specify the path.

Consider the integral definition of the natural logarithm:

$$\mathrm{Ln}(z) = \int_1^z \frac{dt}{t} = \ln(z) + i2k\pi,$$

where $k \in \mathbb{I}$. Remembering that t is a complex variable and that 1/t is analytic everywhere except the origin, the value of k must have something to do with the path t takes from 1 to z. Let us review some of the basic notions associated with complex integration.

$$\begin{aligned}\int_{z_1}^{z_2} f(z)\,dz &= \int_C [u(x, y) + i\,v(x, y)][dx + i\,dy] \\ &= \int_C u(x, y)\,dx - v(x, y)\,dy + i\int_C v(x, y)\,dx + u(x, y)\,dy,\end{aligned}$$

where C is the path of the integrals from z_1 to z_2. The above integrals are actually line integrals defined by path C. Consequently, many of the properties of complex integrals come out of vector analysis. For example, if f(z) is analytic with a continuous derivative at all points in a region R, then the above integral is independent of path. In fact, if contour C consists of a finite number of simple closed curves, then by Green's theorem the above line integrals can be written as the surface integrals

$$\int_C f(z)\,dz = \iint_R \left(-\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right)dx\,dy + i\iint_R \left(\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right)dx\,dy.$$

By the Cauchy–Riemann conditions, both integrands equal zero, so the entire integral equals zero.

If R is a simply connected region bounded by curve C, f(z) is analytic in R, and z_0 is in R, then Cauchy's formula says that

$$f(z_0) = \frac{1}{2\pi i}\int_C \frac{f(z)}{z - z_0}\,dz.$$

Consequently, our integral definition of the natural logarithm may be interpreted as follows:

- The principal value of the integral, ln(z), is that part whose path does not pass through or enclose the origin. Since the function 1/t is analytic in the region of the path, the integral over this path only depends on the end points ln(z) and ln(1) = 0.
- The i2kπ part of the integral corresponds to a path which encircles the origin k times. k > 0 for counterclockwise paths, and k < 0 for clockwise ones.
- The total integral is the sum of the integrals along each of the paths.

See pages 58 and 62 for a numerical demonstration. The integral definitions for the inverse circular and hyperbolic functions are interpreted similarly.

$$\text{Arccos } z = \int_z^1 \frac{dt}{\sqrt{1-t^2}} \qquad \text{Arcsin } z = \int_0^z \frac{dt}{\sqrt{1-t^2}} \qquad \text{Arctan } z = \int_0^z \frac{dt}{1+t^2}$$

$$\text{Arccosh } z = \int_1^z \frac{dt}{\sqrt{t^2-1}} \qquad \text{Arcsinh } z = \int_0^z \frac{dt}{\sqrt{1+t^2}} \qquad \text{Arctanh } z = \int_0^z \frac{dt}{1-t^2}$$

In general, if f(z) is analytic over the region of integration, then its integral follows the same rules as those for real functions of a real variable. Complex integration over analytic paths is provided by the command **CINTG**.

BRANCHES

Consider the function $f(z) = \sqrt{z}$. This function is not continuous on the negative real axis. Evaluate the following limits on your HP 48:

- √(−1,+1.E−200) = (5.E−201,+1) = (1,∠+π/2)
- √(−1,−1.E−200) = (5.E−201,−1) = (1,∠−π/2)

Thus, the function $\sqrt{z}$ is analytic everywhere except the negative real axis. The line from the origin to −∞ is called a *branch cut*, and the origin is called a *branch point*. LN(z) has the same branch cut. By excluding the branch cut we make **LN** analytic so its derivative and integral obey the usual rules of calculus. This we call the *principal branch*. In general, Ln(z) = LN(z) + i2kπ, so each value of k defines a branch, and the cut between each of these branches is the negative real axis.

Less obvious is the fact that the square command **SQ** has a branch cut along the negative real axis due to the fact that it computes the argument in the range $(-\pi, \pi]$. Thus $\sqrt{}(SQ(z)) \neq z$ for $|\arg z| > \pi/2$ because neither command is analytically continued beyond $(-\pi, \pi]$. The square of $(-1, 1) = \sqrt{}2\angle 3\pi/4$ is actually $2\angle 3\pi/2$, not $2\angle -\pi/2$. The square root of $2\angle 3\pi/2$ is $\sqrt{}2\angle 3\pi/4 = (-1,1)$ whereas the square root of $2\angle -\pi/2 = (0,-2)$ is $\sqrt{}2\angle -\pi/4 = (1,-1)$. Similar comments apply to the **XROOT** and power **^** commands. Therefore, care must be used when programming the HP 48 for complex values. In Chapter 15 on the Legendre functions, we encounter expressions like $\sqrt{}(z^2-1)$, which must be evaluated as $\sqrt{}(z-1)\sqrt{}(z+1)$ in order to avoid wrong answers due to this issue. Analytically continued definitions are easy to create, but it is easier and faster to simply use care in programming.

Similarly, ATAN(TAN((−2,1))) = (1.14159265359,1). Subtracting π gives the expected answer, (−2,1). Hence, care must be exercised in evaluating complex functions.

ANALYTIC CONTINUATION

Now let us change over to polar coordinates by defining $z = x + iy = re^{i\theta}$:

$$x = r\cos\theta, \qquad y = r\sin\theta, \qquad r = \sqrt{}(x^2 + y^2), \qquad \theta = \arg z = \arctan(y/x),$$

where r is the *magnitude* and θ is the *argument* of z. Observe that $\sqrt[n]{z} = \sqrt[n]{r}\, e^{i(k2\pi+\theta)/n}$ for $k = 0, 1, \ldots, n-1$. The value is thus not unique, but the HP 48 command **XROOT** will only give you one of these values. Both (1,−1) and (−1,1) are equal to $\sqrt{}(-2i)$, but the HP 48 will only give you the first one. Consequently, care must be used in complex computations to insure that you have the branch that you want. The right-shifted 1 key switches between rectangular and polar argument displays on the HP 48.

Now if $f(z) = u(r, \theta) + i\, v(r, \theta)$, where $u(r, \theta)$ and $v(r, \theta)$ are real functions, then the Cauchy–Riemann conditions for f(z) to be analytic at $z = re^{i\theta}$ are

$$\frac{\partial u\ (r,\ \theta)}{\partial r} = \frac{1}{r}\,\frac{\partial v\ (r,\ \theta)}{\partial \theta}, \qquad \frac{1}{r}\,\frac{\partial u\ (r,\ \theta)}{\partial \theta} = -\,\frac{\partial v\ (r,\ \theta)}{\partial r}.$$

Given the principal branch **LN** defined over $-\pi < \theta < \pi$, how do we extend the definition to all values of θ? The process of extending this definition is called analytic continuation. There are a number of ways to do it, but the one of interest here is contour integration. The generalization of the restricted integral definition of **LN**, which excludes loops around the origin, to paths which include loops around the origin extends the definition of **LN** to Ln. Contour integrals are thus one way functions can be analytically continued.

Consider next Euler's integral of the second kind, which is commonly called the gamma function.

$$\Gamma(z) = \int_0^\infty e^{-t}\, t^{z-1}\, dt \qquad Re(z) > 0.$$

Now how do we extend the definition to $Re(z) \leq 0$? One way is by the reflection formula

$$\Gamma(z)\,\Gamma(1-z) = -z\,\Gamma(-z)\,\Gamma(z) = \pi \csc \pi z,$$

and another way is Hankel's contour integral:

$$\frac{1}{\Gamma(z)} = \frac{i}{2\pi} \int_C (-t)^{-z}\, e^{-t}\, dt,$$

where the path of integration C starts at $+\infty$ on the real axis, circles the origin in the counterclockwise direction, and returns to the starting point. While the second method is of theoretical interest, the first method is of practical use in evaluation.

Where possible and practical, the Math Library provides evaluation of analytically continued definitions of the higher transcendental functions over $[0, \infty)$ and $(-\pi, \pi]$.

SUMMARY

If you wish to evaluate complex integrals, complex integration over analytic paths is provided by **CINTG** in the next chapter. The residue evaluation commands **RESDP** and **RESDA** given in Chapter 25 are also available to help you. The integral tables in Appendix D also apply to complex integrals along analytic paths.

More often, you may simply wish to know what number the integral equals. The following 13 chapters of this manual present commands which evaluate hundreds of difficult complex integrals for you. The transcendental functions they evaluate are also solutions to complex differential equations.

ACCURACY

The accuracy of the functions in the TRIG and HYP menus is about 10 digits.

TRIGONOMETRIC FUNCTIONS MENU { FTNS TRIG }

FUNCTION	COMMAND	INPUTS	OUTPUT
COSECANT	**CSC**(z)	$z \in \mathbb{C}$	VALUE
ARC COSECANT	**ACSC**(z)	$z \in \mathbb{C}$	VALUE
SECANT	**SEC**(z)	$z \in \mathbb{C}$	VALUE
ARC SECANT	**ASEC**(z)	$z \in \mathbb{C}$	VALUE
COTANGENT	**COT**(z)	$z \in \mathbb{C}$	VALUE
ARC COTANGENT	**ACOT**(z)	$z \in \mathbb{C}$	VALUE
ARG [0, 2π]	**ARG2**(z)	$z \in \mathbb{C}$	VALUE
ATAN [0, 2π]	**ATN2**(x,y)	$x, y \in \mathbb{R}$	VALUE

ARG2 is like **ARG** except it returns the argument in radians in the range [0,2π]. **ATN2** is the parallel function to **ARG2** whose inputs are the real part x and the imaginary part y. **ARG** is in the HP MTH PARTS menu.

HP RAD → DEG	**R→D**(r)	$r \in \mathbb{R}$	VALUE
HP DEG → RAD	**D→R**(d)	$d \in \mathbb{R}$	VALUE
Γ(z)	**GAMMA**(z)	$z \in \mathbb{C}$	VALUE
UP DIRECTORY	**UPDIR**	NONE	PARENT MENU

KRONECKER DELTA FUNCTION δ_{mn}

Kδ: « → m n « m n == » »

TRIGONOMETRIC FUNCTIONS MENU { FTNS TRIG }

FUNCTION	COMMAND	INPUTS	OUTPUT

The additional trigonometric functions are implemented by inversion of the appropriate HP trigonometric functions; consequently, a divide–by–zero error is equivalent to an infinite result error for these commands.

Please note that neither the HP 48 nor the MATHLIB trigonometric functions are analytically continued beyond $(-\pi, \pi]$ and as a result ATAN(TAN((–2,1))) = (1.14159265359,1). Subtracting π gives the expected answer of (–2,1).

For definitions see Chapter 4 of Abramowitz, M., and Stegun, I. *Handbook of Mathematical Functions*, AMS 55, Washington, D.C. 1964.

EVALUATION OF PRODUCTS

Infinite product representations of functions such as $\sin z = \prod_{n=1}^{\infty}\left(1 - \frac{z^2}{n^2\pi^2}\right)$ are rarely used in numerical approximations because they tend to converge very slowly. For this example, 100 terms only get you 2 digits of accuracy for z = 1.5.

Nevertheless, you can evaluate the finite product $\prod_{n=0}^{N} f(n)$ with this program:

```
« → FOFn N « 1 FOFn 1 N FOR n n FOFn × NEXT » ».
```

A more useful type of product evaluation is

$$e^z \sim \sum_{n=0}^{N}\frac{z^n}{n!} = 1 + \frac{z}{1}\left(1 + \frac{z}{2}\left(1 + \frac{z}{3}\left(1 + \ldots + \frac{z}{N}\right)\right)\right) = \bigwedge_{n=1}^{N}\left[1 + \frac{z}{n}\right]$$

```
« → z N « 1 N 1 FOR n z n / × 1 + -1 STEP » ».
```

HYPERBOLIC FUNCTIONS MENU { FTNS HYP }

FUNCTION	COMMAND	INPUTS	OUTPUTS
HYPERBOLIC SINE (HP)	SINH(z)	$z \in \mathbb{C}$	VALUE
ARC HYPERBOLIC SINE (HP)	ASINH(z)	$z \in \mathbb{C}$	VALUE
HYPERBOLIC COSINE (HP)	COSH(z)	$z \in \mathbb{C}$	VALUE
ARC HYPERBOLIC COSINE (HP)	ACOSH(z)	$z \in \mathbb{C}$	VALUE
HYPERBOLIC TANGENT (HP)	TANH(z)	$z \in \mathbb{C}$	VALUE
ARC HYPERBOLIC TANGENT (HP)	ATANH(z)	$z \in \mathbb{C}$	VALUE
HYPERBOLIC COSECANT	CSCH(z)	$z \in \mathbb{C}$	VALUE
ARC HYPERBOLIC COSECANT	ACSCH(z)	$z \in \mathbb{C}$	VALUE
HYPERBOLIC SECANT	SECH(z)	$z \in \mathbb{C}$	VALUE
ARC HYPERBOLIC SECANT	ASECH(z)	$z \in \mathbb{C}$	VALUE
HYPERBOLIC COTANGENT	COTH(z)	$z \in \mathbb{C}$	VALUE

HYPERBOLIC FUNCTIONS MENU { FTNS HYP }

FUNCTION	COMMAND	INPUTS	OUTPUTS
ARC HYPERBOLIC COTANGENT	**ACOTH**(z)	$z \in \mathbb{C}$	VALUE
EXPM (HP)	**EXPM**(x)	$x \in \mathbb{R}$	VALUE
Evaluates $e^x - 1$ very accurately for small values of x.			
LNP1 (HP)	**LNP1**(x)	$x > -1$	VALUE
Evaluates LN(x + 1) more accurately than the **LN** command for small values of x.			
$\Gamma(z)$	**GAMMA**(z)	$z \in \mathbb{C}$	VALUE
$\psi(z)$	**PSI**(z)	$z \in \mathbb{C}$	VALUE
UP DIRECTORY	**UPDIR**	NONE	PARENT MENU

The additional hyperbolic functions are implemented by inversion of the appropriate HP hyperbolic functions; consequently, a divide-by-zero error is equivalent to an infinite result error for these commands.

Please note that neither the HP 48 nor the MATHLIB hyperbolic functions are analytically continued beyond $(-\pi,\pi]$, and as a result ATANH(TANH((1, 2))) = (1,–1.14159265359). Adding $i\pi$ gives the expected answer of (1, 2).

For definitions see Chapter 4 of Abramowitz, M., and Stegun, I. *Handbook of Mathematical Functions*, AMS 55, Washington, D.C. 1964.

4

EXPONENTIAL INTEGRAL AND RELATED FUNCTIONS

INTRODUCTION

This chapter presents the 18 exponential integral and related functions menu. These functions are special integrals of the transcendental functions presented in the last chapter. The functions are rigorously defined in the command table.

EXPONENTIAL INTEGRALS

EIOX, **E1OZ**, and **ENOZ** evaluate the exponential integrals Ei(x), $E_1(z)$, and $E_n(z)$.

LOGARITHMIC INTEGRAL

LIOX evaluates the logarithmic integral Li(x).

SINE AND COSINE INTEGRALS

SINT, **CINT**, **SHIZ**, and **CHIZ** evaluate the integrals Si(z), Ci(z), Shi(z), and Chi(z). The functions Shi(z) and Chi(z) are often called the hyperbolic sine and cosine integrals. These commands include asymptotic expansions. You can easily check the accuracy of these commands using **CINTG**. Asymptotic expansions 5.2.34 and 5.2.35 with 5.2.8 and 5.2.9 in AMS 55 are only correct for positive real z.

SINC SQUARED INTEGRAL

The sinc function used in Fourier analysis is defined by sinc(z) = sin(z)/z. Its indefinite integral is given by **SINT**, and the indefinite integral of $sinc^2(z)$ is given by **SCINT**.

RELATED INTEGRALS

Evaluation of several related functions and integrals is also provided.

COMPLEX PLANE NUMERICAL INTEGRATION

CINTG provides complex integration over analytic paths. See Chapter 25 for residue integration. See also pages 58 and 62 for a numerical demonstration of Cauchy's integral formula.

CAUCHY PRINCIPAL VALUE

The integral from minus infinity to infinity is actually a double limit defined by

$$\int_{-\infty}^{\infty} f(t)\ dt = \lim_{T_1,T_2 \to \infty} \int_{-T_1}^{T_2} f(t)\ dt.$$

If f(t) is not integrable in the Riemann or Lebesgue sense, the integral may only exist for the special Cauchy principal value definition where $T_1 = T_2 = T$ and

$$\int_{-\infty}^{\infty} f(t)\ dt = \lim_{T \to \infty} \int_{-T}^{T} f(t)\ dt.$$

By this definition the integral of odd functions over $(-\infty, \infty)$ is zero. The definition of the exponential integral Ei(x) is a variation on this concept. Here the sum of the integrals

$$\lim_{\varepsilon_1,\varepsilon_2 \to 0} \left[\int_{-x}^{-\varepsilon_1} t^{-1}e^{-t} + \int_{\varepsilon_2}^{\infty} t^{-1}e^{-t} \right] dt$$

in general is not defined. By defining $\varepsilon_1 = \varepsilon_2 = \varepsilon$ and then taking the limit $\varepsilon \to 0$, the infinite parts of the integrals exactly cancel, resulting in a finite, well-defined integral for $x > 0$.

EXPONENTIAL INTEGRAL MENU { FTNS EXPIN }

FUNCTION	COMMAND	INPUTS	OUTPUTS
Ei(x)	**EIOX**(x)	x > 0	VALUE

$$Ei(x) = -\int_{-x}^{\infty} \frac{e^{-t}}{t}\,dt = -\lim_{\varepsilon \to 0}\left[\int_{-x}^{-\varepsilon} t^{-1}e^{-t} + \int_{\varepsilon}^{\infty} t^{-1}e^{-t}\right] dt = \int_{-\infty}^{x} \frac{e^{t}}{t}\,dt$$

where the Cauchy principal value is used at the origin. Accuracy is about 10 digits.

ANALYTIC CONTINUATION

EIOX is analytically continued for complex arguments by the relationships

$Ei(z) = -E_1(-z) \quad |\arg z| > 0, \qquad E_1(z) = -Ei(-z) \quad |\arg z| < \pi,$

$E_1(-x \pm i0) = -Ei(x) \mp i\pi, \qquad -Ei(x) = \frac{1}{2}[E_1(-x + i0) + E_1(-x - i0)] \quad x > 0.$

Li(x)	**LIOX**(x)	x > 1	VALUE

The logarithmic integral Li(x) = Ei(LN(x)). Accuracy is about 10 digits.

$E_1(z)$	**E1OZ**(z)	$z \in \mathbb{C}$	VALUE

$$E_1(z) = \int_{z}^{\infty} \frac{e^{-t}}{t}\,dt \qquad |\arg z| < \pi$$

This command has an accuracy of about 10 digits. E1OZ(z) = ENOZ(1,z).

$$E_{n+1}(z) = [\, e^{-z} - z\, E_n(z)\,]/n \qquad n = 1, 2, \ldots$$

EXPONENTIAL INTEGRAL MENU { FTNS EXPIN }

FUNCTION	COMMAND	INPUTS	OUTPUTS
$E_n(z)$	**ENOZ(n,z)**	$n \in \mathbf{N}$ OR 0 $z \in \mathbf{C}$	VALUE

$$E_n(z) = \int_1^{\infty} \frac{e^{-zt}}{t^n}\, dt \qquad \text{Re } z > 0$$

By analytic continuation the definition can be extended to $|\arg z| < \pi$.

This command has an accuracy of about 10 digits.

Si(z)	**SINT(z)**	$z \in \mathbf{C}$	VALUE
Ci(z)	**CINT(z)**	$z \in \mathbf{C}$	VALUE

$$\text{Si}(z) = \int_0^z \frac{\sin t}{t}\, dt \qquad \text{Ci}(z) = \gamma + \text{LN}(z) + \int_0^z \frac{\cos t - 1}{t}\, dt \qquad |\arg z| < \pi$$

These commands have an accuracy of about 10 digits.

SCINT	**SCINT(z)**	$z \in \mathbf{C}$	VALUE

$$\int_0^z \text{sinc}^2(t)\, dt \qquad |\arg z| < \pi \qquad \text{where} \quad \text{sinc}(z) = \frac{\sin(z)}{z}$$

This command has an accuracy of about 10 digits.

sin(z)/z	**SINC(z)**	$z \in \mathbf{C}$	VALUE
$\text{sinc}^2(z)$	**SINC2(z)**	$z \in \mathbf{C}$	VALUE

EXPONENTIAL INTEGRAL MENU { FTNS EXPIN }

FUNCTION	COMMAND	INPUTS	OUTPUTS
Shi(z)	**SHIZ**(z)	$z \in \mathbf{C}$	VALUE
Chi(z)	**CHIZ**(z)	$z \in \mathbf{C}$	VALUE

$$\text{Shi}(z) = \int_0^z \frac{\sinh t}{t}\, dt \qquad \text{Chi}(z) = \gamma + \text{LN}(z) + \int_0^z \frac{\cosh t - 1}{t}\, dt \qquad |\arg z| < \pi$$

These commands have an accuracy of about 10 digits.

.577215664902	γ	NONE	VALUE

This is Euler's constant.

$\alpha_n(z)$	α**NOZ**(n,z)	$n \in \mathbf{N}$ OR 0 $z \in \mathbf{C}$	VALUE
$\beta_n(z)$	β**NOZ**(n,z)	$n \in \mathbf{N}$ OR 0 $z \in \mathbf{C}$	VALUE

$$\alpha_n(z) = \int_1^\infty t^n e^{-zt}\, dt \quad \text{Re } z > 0 \qquad \beta_n(z) = \int_{-1}^1 t^n e^{-zt}\, dt$$

These commands are evaluated using the relationships

$$\alpha_n(z) = z^{-n-1}\, \Gamma(n+1, z) \text{ and } \beta_n(z) = z^{-n-1}\, [\, \Gamma(n+1, -z) - \Gamma(n+1, z)\,],$$

using the analytically continued definition of the incomplete gamma function **INCGC** given in Chapter 12. See the accuracy comments given there.

$\Gamma(z)$	**GAMMA**(z)	$z \in \mathbf{C}$	VALUE

EXPONENTIAL INTEGRAL MENU { FTNS EXPIN }

FUNCTION	COMMAND	INPUTS	OUTPUTS
$\psi(z)$	**PSI(z)**	$z \in \mathbb{C}$	VALUE
The above two commands are discussed in Chapter 5.			
COMPLEX INTEGRATION	**CINTG(L,U,F,z)**	$L \in \mathbb{C}$ $U \in \mathbb{C}$ $F(z) \in \mathbb{C}$ $z \in \mathbb{C}$	VALUE
CINTG numerically integrates analytic F(z) from z = L to z = U. The integration is along the straight-line path from L to U. F(z) must be analytic on both the path and all neighborhoods of the path. Singularities can be avoided by performing integrations along several simply connected paths. The aggregate path may encircle any poles of F. For example, the integral of 1/z from 1 to (2, 3) equals LN((2,3)) = (1.28247467873, .982793723247). Consider setting 7 SCI to speed integration. The complex integration error is stored in HP reserved variable IERR. For evaluation of complex residues, see **RESDP** and **RESDA** in Chapter 25.			
UP DIRECTORY	**UPDIR**	NONE	PARENT MENU
For definitions see Chapter 5 of Abramowitz, M., and Stegun, I. *Handbook of Mathematical Functions*, AMS 55, Washington, D.C. 1964.			

5

GAMMA AND RELATED FUNCTIONS

INTRODUCTION

This chapter presents the 12 gamma function and related commands. The gamma function $\Gamma(z)$ is a generalization of the factorial function. In fact, $\Gamma(n + 1) = n!$, where ! is the factorial function on your calculator. The **PERMF** command given in the MISC menu in Chapter 17 is the complex generalization of the permutation function **PERM** on the HP 48. The reciprocal of the complex beta function $B(w, z)$ is a complex generalization of the combinations function **COMB** on the calculator. Both the gamma function and the beta function have definite integral representations. If these integrals are turned into indefinite integrals, one obtains the incomplete gamma and beta functions.

GAMMA AND BETA FUNCTIONS

Commands **GAMMA** and **BETA** evaluate the complex gamma and beta functions analytically continued to the entire complex plane. The only singularities of $\Gamma(z)$ are simple poles with residues $(-1)^n/n!$ for $z = -n$ where $n = 0, 1, \ldots$. $1/\Gamma(z)$ is an entire function. The factorial function ! is sometimes denoted as the Π function. $\Pi(z) = z!$.

INCOMPLETE GAMMA FUNCTION

INCG and **INC**γ are the incomplete gamma functions $\Gamma(a, z)$ and $\gamma(a, z)$. **PINCG** is the Pearson incomplete gamma function I(u, p). **INC**γ¡ is the normalized incomplete gamma function $\gamma^*(a, z)$. These are analytically continued for complex arguments using the incomplete gamma functions given in Chapter 12 as a special case of the confluent hypergeometric function.

INCOMPLETE BETA FUNCTION

INCB and **INC**β are the incomplete beta functions $I_z(a, b)$ and $B_z(a, b)$. These are analytically continued for complex arguments using the incomplete beta function given in Chapter 14 as a special case of the Gaussian hypergeometric function.

DIGAMMA AND POLYGAMMA FUNCTIONS

The commands **PSI** and **DNPSI** provide the complex digamma and polygamma functions.

DOUBLE FACTORIAL FUNCTION

The command ¡¡ provides the double factorial function !!.

RELATIONSHIPS ASSOCIATED WITH THE RATIO OF GAMMA FUNCTIONS

Using the reflection formula for the gamma function, relationships are defined for the ratio of gamma functions.

GAMMA FUNCTION MENU
{ FTNS GAMA }

FUNCTION	COMMAND	INPUTS	OUTPUTS
$\Gamma(z)$	**GAMMA**(z)	$z \in \mathbf{C}$	VALUE

GAMMA is the analytically continued gamma function often defined by Euler's integral of the second kind given by

$$\Gamma(z) = \int_0^\infty e^{-t}\, t^{z-1}\, dt \qquad Re(z) > 0.$$

Two relationships which hold for the Gamma function are the recurrence formula

$$\Gamma(z + 1) = z\,\Gamma(z) = z! = z\,(z - 1)!$$

and the reflection formula

$$\Gamma(z)\,\Gamma(z - 1) = -z\,\Gamma(-z)\,\Gamma(z) = \pi \csc \pi z.$$

This command yields about 10–digit accuracy throughout the complex plane, except in the neighborhood of the poles located at z = 0, –1, –2, ...

$\psi(z)$	**PSI**(z)	$z \in \mathbf{C}$	VALUE

PSI is the digamma function defined by $\psi(z) = d\,[\ln \Gamma(z)]/dz = [d\Gamma(z)/dz]/\Gamma(z)$.

Recurrence formula: $\psi(z + 1) = \psi(z) + 1/z.$
Reflection formula: $\psi(1 - z) = \psi(z) + \pi \cot \pi z.$

This command yields about 10–digit accuracy throughout the complex plane, except in the neighborhood of the poles located at z = 0, –1, –2, ...

For special arguments, see the equations at the end of this menu.

GAMMA FUNCTION MENU
{ FTNS GAMA }

FUNCTION	COMMAND	INPUTS	OUTPUTS
$\Gamma(a, z)$	**INCG**(a,z)	$a \in \mathbb{C}$ $z \in \mathbb{C}$	VALUE

$$\Gamma(a, z) = \int_z^\infty e^{-t}\, t^{a-1}\, dt$$

For $z \geq 0$ and $a > 0$ this command yields about 10–digit accuracy. For $a = 0, -1, -2, \ldots$, $\Gamma(a, z) = z^a\, E_{1-a}(z)$, where $E_n(z)$ is the exponential integral defined in Chapter 4. For the other cases, **INCGC** defined in Chapter 12 is used for the evaluation. Numerical stability of **INCGC** is very poor near $a = 0, -1, -2, \ldots$. Thus, the numerical limit does not approach the actual value given by

$$z^a\, E_{1-a}(z)$$

in the neighborhood of the poles of $\Gamma(z)$.

$\gamma(a, z)$	**INC**γ(a,z)	$a \in \mathbb{C}$ $z \in \mathbb{C}$	VALUE

$\gamma(a, z) = \Gamma(a) - \Gamma(a, z)$ = GAMMA(a) – INCG(a, z). The result is undefined for $z = 0$ and Re $a \leq 0$, as well as the poles located at $a = 0, -1, -2, \ldots$.

PEARSON I(u, p)	**PINCG**(u,p)	$u \in \mathbb{C}$ $p \in \mathbb{C}$	VALUE

$$I(u, p) = \frac{1}{\Gamma(p+1)} \int_0^{u\sqrt{p+1}} e^{-t}\, t^p\, dt = 1 - \Gamma\left(p+1,\, u\sqrt{p+1}\right)/\Gamma(p+1)$$

This command is evaluated with the above equation using **INCG** for $\Gamma(a, z)$.

GAMMA FUNCTION MENU
{ FTNS GAMA }

FUNCTION	COMMAND	INPUTS	OUTPUTS
B(w, z)	**BETA**(w,z)	$w \in \mathbb{C}$ $z \in \mathbb{C}$	VALUE

The beta function may be defined in terms of Euler's integral of the first kind:

$$B(z, w) = \int_0^1 t^{z-1} (1 - t)^{w-1} \, dt = \frac{\Gamma(z)\,\Gamma(w)}{\Gamma(z + w)}.$$

This command yields about 10-digit accuracy throughout the complex plane except in the neighborhoods of the singularities of the complex gamma function.

$I_z(a, b)$	**INCB**(a,b,z)	$a, b \in \mathbb{C}$ $\lvert z \rvert \leq 1$	VALUE

$$I_z(a, b) = B_z(a, b)/B(a, b)$$

For $a, b > 0$ and $z \geq 0$, this command yields about 10-digit accuracy. For other input values, **INCBH** discussed in Chapter 14 is evaluated.

$\psi^{(n)}(z)$	**DNPSI**(n,z)	$n = 0, 1, \ldots$ $z \in \mathbb{C}$	VALUE

DNPSI(1, z) is the trigamma function which is the derivative of $\psi(z)$.

This command yields about 10-digit accuracy throughout the complex plane except in the neighborhood of the singularities at 0, –1, –2, ...

$$\psi^{(n)}(z) = \frac{d^n \psi(z)}{dz^n} = (-1)^{n+1} \int_0^\infty \frac{t^n e^{-zt}}{1 - e^{-t}} \, dt \qquad \text{Re } z > 0,$$

where $\psi^{(n)}(z)$ is the polygamma function. See also page 56.

GAMMA FUNCTION MENU
{ FTNS GAMA }

FUNCTION	COMMAND	INPUTS	OUTPUTS
$\gamma^*(a, z)$	INCγ(a,z)	$a \in \mathbf{C}$ $z \in \mathbf{C}$	VALUE

For real z, γ^* is a single-valued analytic function of a and z possessing no finite singularities.

$$\gamma^*(a, x) = \frac{x^{-a}}{\Gamma(a)} \int_0^x e^{-t}\, t^{a-1}\, dt \qquad \text{Re } a > 0$$

It is evaluated using the confluent hypergeometric function discussed in Chapter 12 as:

$$\gamma^*(a, z) = \begin{cases} z^{-a} & a = 0, -1, -2, \ldots \\ M(a, a + 1, -z)/\Gamma(a + 1). \end{cases}$$

This function is plotted for real a and z on page 261 of AMS 55.

$B_z(a, b)$	INCβ(a,b,z)	$a, b \in \mathbf{C}$ $\lvert z\rvert \le 1$	VALUE

$$B_z(a, b) = \int_0^z t^{a-1}\, (1 - t)^{b-1}\, dt \qquad \text{Re } a > 0, \quad \text{Re } b > 0, \quad |z| \le 1$$

INCβ is analytically continued and evaluated as $z^a F(a, 1 - b, a + 1, z)/a$ where F is the Gaussian hypergeometric function discussed in Chapter 14.

(n)!!	;;(n)	$n \in \mathbf{N}$ OR 0	VALUE

$$(n)!! = \begin{cases} (2m)!! = 2\ 4\ 6 \ldots (2m) = 2^m\, m! & n = 2m \\ (2m - 1)!! = 1\ 3\ 5 \ldots (2m - 1) = 2^n\, \Gamma(m + .5)/\sqrt{\pi} & n = 2m-1 \end{cases}$$

GAMMA FUNCTION MENU
{ FTNS GAMA }

FUNCTION	COMMAND	INPUTS	OUTPUTS

RELATIONSHIPS ASSOCIATED WITH THE RATIO OF GAMMA FUNCTIONS

The complex generalization of the permutations command **PERM** is defined by

$$\text{PERMF}(z, r) = \Gamma(z + 1)/\Gamma(z + 1 - r)$$

and similarly Pochhammer's symbol

$$\text{POCH}(z, n) = (z)_n = \Gamma(z + n)/\Gamma(z).$$

These two commands are available in the MISC menu discussed in Chapter 17.

The complex generalization of the binomial coefficient (combinations) can be defined by the following equation for COMBF:

$$\binom{z}{r} = \text{COMBF}(z, r) = \begin{cases} \text{PERMF}(z, r)/r! & r = 0, 1, 2, \ldots \\ 0 & r = -1, -2, \ldots \end{cases}$$

PERMF, **POCH**, and COMBF are all numerically stabilized in MATHLIB for integer values of the second argument. While the second argument may be real or complex, these cases are not stabilized. From the reflection formula for the gamma function, the following identities hold:

$$\text{PERMF}(-z, n) = (-1)^n \text{ PERMF}(z + n - 1, n)$$

$$\text{POCH}(-z, n) = (-1)^n \text{ POCH}(z - n + 1, n)$$

$$\text{COMBF}(-z, n) = (-1)^n \text{ COMBF}(z + n - 1, n).$$

UP DIRECTORY	UPDIR	NONE	PARENT MENU

GAMMA FUNCTION MENU
{ FTNS GAMA }

FUNCTION	COMMAND	INPUTS	OUTPUTS

THE GAMMA AND DIGAMMA FUNCTION FOR SPECIAL ARGUMENTS

$$\Gamma(\tfrac{1}{2}) = \sqrt{\pi} \qquad \Gamma(n + \tfrac{1}{2}) = 2^{-n}\,(2n - 1)!!\,\sqrt{\pi}$$

$\psi(1) = -\gamma = -.577215664902$ where γ is Euler's constant and command γ.

$$\psi(n) = -\gamma + \sum_{k=1}^{n-1} k^{-1} \qquad n \geq 2$$

$$\psi(\tfrac{1}{2}) = -\gamma - 2 \ln 2 = -1.963510026021$$

$$\psi(n + \tfrac{1}{2}) = -\gamma - 2 \ln 2 + 2\left(1 + \frac{1}{3} + \ldots + \frac{1}{2n - 1}\right) \qquad n \geq 1$$

SYMBOLIC POLYGAMMA FUNCTION

Program DNPSIS(n,z) is the symbolic $\psi^{(n)}(z)$, and derDNPSIS(n,z,dn,dz) is its defined derivative. See the end of Chapter 19 and Appendix A. Observe that

$$\psi^{(n)}(z) = \Gamma(n + 1)\,(-1)^{n+1}\,\zeta(n + 1, z) = \Gamma(n + 1)\,(-1)^{n+1}\,\Phi(1, n + 1, z)$$

where $\zeta(s, \alpha)$ and $\Phi(z, s, \alpha)$ are discussed on page 112.

For definitions see Chapter 6 of Abramowitz, M., and Stegun, I. *Handbook of Mathematical Functions*, AMS 55, Washington, D.C. 1964.

6

ERROR FUNCTION AND FRESNEL INTEGRALS

INTRODUCTION

This chapter presents the 14 error function and Fresnel integral commands in the ERROR menu. These are special integrals of the exponential, sine, and cosine functions where the argument is quadratic in the independent variable.

ERROR FUNCTION

ERFZ and **ERFCZ** provide evaluation of the complex error function erf(z) and the complementary error function erfc(z). These are related to the normal distribution function.

FRESNEL INTEGRALS

There are three sets of definitions for these functions that are in use. **SOFZ**, **COFZ**, **S1OZ**, **C1OZ**, **S2OZ**, and **C2OZ** evaluate all three of the sets S(z), C(z), $S_1(z)$, $C_1(z)$, $S_2(z)$, $C_2(z)$.

RELATED FUNCTIONS

ZOFZ evaluates Z(z), which is the derivative of the normal probability distribution, which is called the normal probability density function. The relations are

$$\text{UTPN}(0,1,x) = \text{erfc}(x / \sqrt{2}) / 2, \qquad \text{LTPN}(0,1,x) = \text{erfc}(-x / \sqrt{2}) / 2,$$

where **UTPN** and **LTPN** are the MATHLIB commands for the upper and lower tails of the normal probability distribution discussed in Chapter 23.

INERFC evaluates the nth integral of the complementary error function i^n erfc(z).

COMPLEX PLANE NUMERICAL INTEGRATION

CINTG provides complex integration over analytic paths. See Chapter 25 for residue integration. Program AINTG on page 62 provides the capability of doing a sequence of complex integrations. AINTG({ Z_0 , Z_1 , Z_2 , . . ., Z_N } , F , z) computes the sum:

$$\int_{Z_0}^{Z_1} F(z)\ dz + \int_{Z_1}^{Z_2} F(z)\ dz + \ldots + \int_{Z_{N-1}}^{Z_N} F(z)\ dz.$$

NUMERICAL DEMONSTRATION OF CAUCHY'S INTEGRAL FORMULA

Cauchy's integral formula states that the integral of f(z) around a closed contour C encircling a pole z_0 of f(z) is equal to $2\pi i$ times the residue of f(z) at $z = z_0$:

$$\oint_C f(z)\ dz = 2\pi i \times \text{Res}_f\ (z_0).$$

More generally, the integral is equal to $2\pi i$ times the sum of the residues of f(z) for each of the poles of f(z) which are enclosed by contour C. Consider the gamma function with residues of $(-1)^n / n!$ at $z = -n$ for $n = 0, 1, \ldots$. Then the integral of $\Gamma(z)$ around a given pole $z = -n$ must equal $2\pi i \times (-1)^n/n!$. The program CIF on page 62 numerically computes the integral as the three legs of a triangle around the pole of the gamma function at $z = -n$ for input $n = 0, 1, \ldots$ in a counterclockwise sense. Then it computes the integral using Cauchy's integral formula.

DAWSON'S INTEGRAL

A program is given for evaluating Dawson's integral using **ERFZ**.

ERROR FUNCTION AND FRESNEL INTEGRAL MENU { FTNS ERROR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
S(z)	SOFZ(z)	$z \in \mathbb{C}$	VALUE

$$S(z) = \int_0^z \sin\left(\frac{\pi}{2}t^2\right) dt$$

In the region $2 < |z| < 5$, the complex approximation is less than 10 digits. Consequently, you may want to use **CINTG** to perform the integration numerically.

CINTG(0 , (1,2) , 'SIN(πt^2/2)' , t) = (36.7254648839, 15.5877511043)

C(z)	COFZ(z)	$z \in \mathbb{C}$	VALUE

$$C(z) = \int_0^z \cos\left(\frac{\pi}{2}t^2\right) dt$$

In the region $2 < |z| < 5$, the complex approximation is less than 10 digits. Consequently, you may want to use **CINTG** to perform the integration numerically.

CINTG(0 , (1,2) , 'COS(πt^2/2)' , t) = (16.087871374, –36.2256879928)

$S_1(z)$	S1OZ(z)	$z \in \mathbb{C}$	VALUE
$C_1(z)$	C1OZ(z)	$z \in \mathbb{C}$	VALUE

$S_1(z) = S(z/\sqrt{\pi/2})$ and $C_1(z) = C(z/\sqrt{\pi/2})$

ERROR FUNCTION AND FRESNEL INTEGRAL MENU { FTNS ERROR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
$S_2(z)$	**S2OZ(z)**	$z \in \mathbb{C}$	VALUE
$C_2(z)$	**C2OZ(z)**	$z \in \mathbb{C}$	VALUE

$$S_2(z) = S(\sqrt{z/(\pi/2)}\,) \quad \text{and} \quad C_2(z) = C(\sqrt{z/(\pi/2)}\,)$$

erf(z)	**ERFZ(z)**	$z \in \mathbb{C}$	VALUE

$$\text{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2}\, dt$$

ERFZ and **ERFCZ** have an accuracy of about 10 digits. For real z, the upper tail **ERFCZ** is actually computed and then subtracted from 1. For complex inputs, including ones with an imaginary part of 1E–200, the lower tail **ERFZ** is computed. This value is subtracted from 1 to get **ERFCZ**.

erfc(z)	**ERFCZ(z)**	$z \in \mathbb{C}$	VALUE

$$\text{erfc}(z) = 1 - \text{erf}(z)$$

$Z(z) = (2\pi)^{-1/2} \exp(-z^2/2)$	**ZOFZ(z)**	$z \in \mathbb{C}$	VALUE

ERROR FUNCTION AND FRESNEL INTEGRAL MENU { FTNS ERROR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
i^n erfc(z)	**INERFC**(n,z)	$n \in \mathbb{N}$ OR 0 $z \in \mathbb{C}$	VALUE

$$i^n \operatorname{erfc}(z) = \int_z^\infty i^{n-1} \operatorname{erfc}(t)\, dt = e^{-z^2} \left[\frac{M\left(\frac{n+1}{2}, \frac{1}{2}, z^2\right)}{2^n\, \Gamma\left(\frac{n}{2}+1\right)} - \frac{z\, M\left(\frac{n}{2}+1, \frac{3}{2}, z^2\right)}{2^{n-1}\, \Gamma\left(\frac{n+1}{2}\right)} \right]$$

where M is the confluent hypergeometric function.

$\Gamma(z)$	**GAMMA**(z)	$z \in \mathbb{C}$	VALUE
$\psi(z)$	**PSI**(z)	$z \in \mathbb{C}$	VALUE

See Chapter 5.

COMPLEX INTEGRATION	**CINTG**(L,U,F,z)	$L \in \mathbb{C}$ $F(z) \in \mathbb{C}$	$U \in \mathbb{C}$ $z \in \mathbb{C}$	VALUE

CINTG numerically integrates analytic F(z) from z = L to z = U. The integration is along the straight-line path from L to U. F(z) must be analytic on both the path and all neighborhoods of the path. Singularities can be avoided by performing integrations along several simply connected paths. The aggregate path may encircle any poles of F. For example, the integral of 1/z from 1 to (2, 3) equals LN((2,3)) = (1.28247467873, .982793723247). Consider setting 7 SCI to speed integration. The complex integration error is stored in HP reserved variable IERR. For evaluation of complex residues, see **RESDP** and **RESDA** in Chapter 25.

ERROR FUNCTION AND FRESNEL INTEGRAL MENU { FTNS ERROR }

FUNCTION	COMMAND	INPUTS	OUTPUTS

NUMERICAL DEMONSTRATION OF CAUCHY'S INTEGRAL FORMULA

Program AINTG below performs a sequence of N complex integrations between each pair of numbers in list (vector) L of SIZE N + 1 ≥ 2. See page 58.

« → L F z « 0 1 L SIZE EVAL 1 - FOR K 'CINTG(L(K),L(K+1),F,z)' →NUM + NEXT » »

Program CIF runs in about 21 minutes. Input n = 0, 1, 2, ... See page 58.

CIF: « 7 → n N « N SCI [(-.5,-1) (.5,0) (-.5,1) (-.5,-1)] n VSUB 'GAMMA(Z)' Z AINTG '2×π×i×(-1)^n / n!' →NUM STD » » »

For n = 3, the integral equals (0, -1.0471975512).
For n = 4, the integral equals (0, .261799387799).

EVALUATION OF DAWSON'S INTEGRAL

$$\int_0^z e^{t^2}\,dt = \frac{i\sqrt{\pi}}{2}\,\mathrm{erf}(-iz)$$

The following program may be used to evaluate Dawson's integral:

DAWS: « → α « 'i×√π/2×ERFZ(-i×α)' →NUM » ».

UP DIRECTORY	UPDIR	NONE	PARENT MENU

For definitions see Chapter 7 of Abramowitz, M., and Stegun, I. *Handbook of Mathematical Functions*, AMS 55, Washington, D.C. 1964.

7

BESSEL FUNCTIONS OF INTEGER ORDER

INTRODUCTION

This chapter presents the 20 commands in the BESEL menu which evaluate Bessel functions of integer order. Chapter 8 gives commands for evaluation of spherical Bessel functions. For evaluation of Bessel functions of complex order and complex argument, additional commands are provided in Chapter 12. We present a short overview of these functions here.

Sine, cosine, and exponential functions are associated with solutions to physics problems in rectangular coordinates. Bessel or cylinder functions are associated with solutions in circular cylindrical coordinates. Bessel's differential equation is

$$\frac{d^2w}{dz^2} + \frac{1}{z}\frac{dw}{dz} + \left(\pm 1 - \frac{v^2}{z^2}\right) w = 0,$$

where the + case gives rise to the ordinary Bessel functions and the – case gives rise to the modified Bessel functions. Remembering from Chapter 3 that the distinction between the trigonometric and hyperbolic differential equations was also the sign, we have these analogies:

trigonometric functions	ordinary Bessel functions
hyperbolic functions	modified Bessel functions
exponential functions	Hankel functions

Bessel functions are also useful in performing integrations. Bessel transforms are the analog of Fourier transforms and are very well tabulated.

ORDINARY BESSEL FUNCTIONS

$J_\nu(z)$ denotes the Bessel function of the first kind. The Bessel function of the second kind, $Y_\nu(z)$, is also called Weber's function or Neumann's Bessel function of the second kind, denoted by $N_\nu(z)$. Bessel functions of the third kind, $H_\nu^{(1)}(z)$ and $H_\nu^{(2)}(z)$, are generally called Hankel functions. In general, both the order ν and the argument z can be complex. These solutions are related by

$$Y_\nu(z) = \frac{J_\nu(z)\cos(\nu\pi) - J_{-\nu}(z)}{\sin(\nu\pi)},$$

where the right side of the equation is replaced by its limiting value when ν is an integer or zero.

$$H_\nu^{(1)}(z) = J_\nu(z) + i\,Y_\nu(z)$$
$$H_\nu^{(2)}(z) = J_\nu(z) - i\,Y_\nu(z)$$

Other relations include

$$J_{-n}(z) = (-1)^n J_n(z), \qquad H_{-\nu}^{(1)}(z) = e^{i\nu\pi} H_\nu^{(1)}(z),$$
$$Y_{-n}(z) = (-1)^n Y_n(z), \qquad H_{-\nu}^{(2)}(z) = e^{-i\nu\pi} H_\nu^{(2)}(z).$$

By analytic continuation, we have for integer m:

$$J_\nu(z\,e^{i\,m\pi}) = e^{i\,m\nu\pi} J_\nu(z),$$

$$Y_\nu(z\,e^{i\,m\pi}) = e^{-i\,m\nu\pi} Y_\nu(z) + 2i\,\sin(m\nu\pi)\,\cot(\nu\pi)\,J_\nu(z),$$

$$H_\nu^{(1)}(z\, e^{i\pi}) = -e^{-i\nu\pi} H_\nu^{(2)}(z), \qquad H_\nu^{(2)}(z\, e^{-i\pi}) = -e^{i\nu\pi} H_\nu^{(1)}(z).$$

$J_\nu(z)$, $K_\nu(z)$, $H_\nu^{(1)}(z)$, and $H_\nu^{(2)}(z)$ satisfy the recurrence relations:

$$\mathcal{C}_{\nu-1}(z) + \mathcal{C}_{\nu+1}(z) = \frac{2\nu}{z}\mathcal{C}_\nu(z)$$

$$\mathcal{C}_{\nu-1}(z) - \mathcal{C}_{\nu+1}(z) = 2\,\mathcal{C}'_\nu(z)$$

$$\mathcal{C}'_\nu(z) = \mathcal{C}_{\nu-1}(z) - \frac{\nu}{z}\mathcal{C}_\nu(z)$$

$$\mathcal{C}'_\nu(z) = -\mathcal{C}_{\nu+1}(z) + \frac{\nu}{z}\mathcal{C}_\nu(z)$$

$J_n(z)$ may be represented by the integrals

$$J_n(z) = \frac{1}{\pi}\int_0^\pi \cos(z\sin\theta - n\theta)\,d\theta = \frac{i^{-n}}{\pi}\int_0^\pi e^{iz\cos\theta}\cos(n\theta)\,d\theta.$$

The following identities are useful:

$$\cos(z\sin\theta) = J_0(z) + 2\sum_{k=1}^{\infty} J_{2k}(z)\cos(2k\theta) \qquad \sin(z\sin\theta) = 2\sum_{k=0}^{\infty} J_{2k+1}(z)\sin\{(2k+1)\theta\}$$

$$\cos(z\cos\theta) = J_0(z) + 2\sum_{k=1}^{\infty}(-1)^k J_{2k}(z)\cos(2k\theta) \qquad \cos z = J_0(z) + 2\sum_{k=1}^{\infty}(-1)^k J_{2k}(z)$$

$$\sin(z\cos\theta) = 2\sum_{k=0}^{\infty}(-1)^k J_{2k+1}(z)\cos\{(2k+1)\theta\} \qquad \sin z = 2\sum_{k=0}^{\infty}(-1)^k J_{2k+1}(z)$$

$$1 = J_0(z) + 2\sum_{k=1}^{\infty} J_{2k}(z)$$

MODIFIED BESSEL FUNCTIONS

$I_\nu(z)$ denotes the modified Bessel function of the first kind, and $K_\nu(z)$ denotes the modified Bessel function of the second kind. In general, both the order ν and the argument can be complex. These solutions are related by

$$K_\nu(z) = \frac{\pi}{2} \frac{I_{-\nu}(z) - I_\nu(z)}{\sin(\nu\pi)},$$

where the right side of the equation is replaced by its limiting value when ν is an integer or zero.

$$\begin{aligned}
I_\nu(z) &= e^{-i\nu\pi/2} J_\nu(z\, e^{i\pi/2}) && -\pi < \arg z \le \pi/2 \\
I_\nu(z) &= e^{i\nu 3\pi/2} J_\nu(z\, e^{-i3\pi/2}) && \pi/2 < \arg z \le \pi \\
K_\nu(z) &= \tfrac{1}{2} i\pi\, e^{i\nu\pi/2} H_\nu^{(1)}(z\, e^{i\pi/2}) && -\pi < \arg z \le \pi/2 \\
K_\nu(z) &= -\tfrac{1}{2} i\pi\, e^{-i\nu\pi/2} H_\nu^{(2)}(z\, e^{-i\pi/2}) && -\pi/2 < \arg z \le \pi
\end{aligned}$$

$$Y_\nu(ze^{i\pi/2}) = e^{i(\nu+1)\pi/2} I_\nu(z) - \frac{2}{\pi} e^{-i\nu\pi/2} K_\nu(z) \qquad -\pi < \arg z \le \pi/2$$

Other relations include

$$I_{-n}(z) = I_n(z), \qquad K_{-\nu}(z) = K_\nu(z).$$

By analytic continuation we have for integer m:

$$I_\nu(z\, e^{im\pi}) = e^{im\nu\pi} I_\nu(z),$$

$$K_\nu(z\, e^{im\pi}) = e^{-im\nu\pi} K_\nu(z) - i\pi \sin(m\nu\pi) \csc(\nu\pi) I_\nu(z).$$

$I_\nu(z)$ and $e^{i\nu\pi} K_\nu(z)$ satisfy the recurrence relations:

$$\mathfrak{C}_{\nu-1}(z) - \mathfrak{C}_{\nu+1}(z) = \frac{2\nu}{z} \mathfrak{C}_\nu(z)$$

$$\mathfrak{C}_\nu'(z) = \mathfrak{C}_{\nu-1}(z) - \frac{\nu}{z}\mathfrak{C}_\nu(z)$$

$$\mathfrak{C}_{\nu-1}(z) + \mathfrak{C}_{\nu+1}(z) = 2\,\mathfrak{C}_\nu'(z)$$

$$\mathfrak{C}_\nu'(z) = \mathfrak{C}_{\nu+1}(z) + \frac{\nu}{z}\mathfrak{C}_\nu(z)$$

$I_n(z)$ may be represented by the integral

$$I_n(z) = \frac{1}{\pi}\int_0^\pi e^{z\cos\theta}\cos(n\theta)\,d\theta.$$

The following identities are useful:

$$e^{z\cos\theta} = I_0(z) + 2\sum_{k=1}^{\infty} I_k(z)\cos(k\theta)$$

$$e^{z\sin\theta} = I_0(z) + 2\sum_{k=0}^{\infty}(-1)^k I_{2k+1}(z)\sin\{(2k+1)\theta\} + 2\sum_{k=1}^{\infty}(-1)^k I_{2k}(z)\cos(2k\theta)$$

$$\cosh z = I_0(z) + 2\sum_{k=1}^{\infty} I_{2k}(z) \qquad \sinh z = 2\sum_{k=0}^{\infty} I_{2k+1}(z)$$

$$1 = I_0(z) + 2\sum_{k=1}^{\infty}(-1)^k I_{2k}(z)$$

$$e^{z} = I_0(z) + 2\sum_{k=1}^{\infty} I_k(z) \qquad e^{-z} = I_0(z) + 2\sum_{k=1}^{\infty}{}' (-1)^k I_k(z)$$

KELVIN FUNCTIONS

For real ν and $x \geq 0$, the Kelvin (Thomson) functions are defined as

$$\mathrm{ber}_\nu x + i\,\mathrm{bei}_\nu x = J_\nu(x\,e^{i3\pi/4}) = e^{i\nu\pi/2} I_\nu(x\,e^{i\pi/4}),$$
$$\mathrm{ker}_\nu x + i\,\mathrm{kei}_\nu x = e^{-i\nu\pi/2} K_\nu(x\,e^{i\pi/4}) = \tfrac{1}{2}i\pi\, H_\nu^{(1)}(x\,e^{i3\pi/4}).$$

They are solutions to the differential equation

$$\frac{d^2w}{dx^2} + \frac{1}{x}\frac{dw}{dx} - \left(i + \frac{\nu^2}{x^2}\right) w = 0.$$

Relations between solutions include

$$\begin{aligned}
\mathrm{ber}_{-\nu}\, x &= \cos(\nu\pi)\, \mathrm{ber}_{\nu}\, x + \sin(\nu\pi)\, \mathrm{bei}_{\nu}\, x + (\pi/2)\sin(\nu\pi)\, \mathrm{ker}_{\nu}\, x,\\
\mathrm{bei}_{-\nu}\, x &= -\sin(\nu\pi)\, \mathrm{ber}_{\nu}\, x + \cos(\nu\pi)\, \mathrm{bei}_{\nu}\, x + (2/\pi)\sin(\nu\pi)\, \mathrm{kei}_{\nu}\, x,\\
\mathrm{ker}_{-\nu}\, x &= \cos(\nu\pi)\, \mathrm{ker}_{\nu}\, x - \sin(\nu\pi)\, \mathrm{kei}_{\nu}\, x,\\
\mathrm{kei}_{-\nu}\, x &= \sin(\nu\pi)\, \mathrm{ker}_{\nu}\, x + \cos(\nu\pi)\, \mathrm{kei}_{\nu}\, x,
\end{aligned}$$

$$\mathrm{ber}_n(-x) = (-1)^n\, \mathrm{ber}_n(x), \qquad \mathrm{bei}_n(-x) = (-1)^n\, \mathrm{bei}_n(x).$$

Recurrence relations for the Kelvin functions include

$$f_{\nu+1} + f_{\nu-1} = -\frac{\nu\sqrt{2}}{x}(f_\nu - g_\nu), \qquad f'_\nu = \frac{1}{2\sqrt{2}}(f_{\nu+1} + g_{\nu+1} - f_{\nu-1} - g_{\nu-1}),$$

$$f'_\nu - \frac{\nu}{x} f_\nu = \frac{1}{\sqrt{2}}(f_{\nu+1} + g_{\nu+1}), \qquad f'_\nu + \frac{\nu}{x} f_\nu = -\frac{1}{\sqrt{2}}(f_{\nu-1} + g_{\nu-1}),$$

where f_ν and g_ν are defined in pairs as either of the four groups

$$\begin{aligned}
f_\nu &= \mathrm{ber}_\nu\, x \text{ and } g_\nu = \mathrm{bei}_\nu\, x, & f_\nu &= \mathrm{bei}_\nu\, x \text{ and } g_\nu = -\mathrm{ber}_\nu\, x,\\
f_\nu &= \mathrm{ker}_\nu\, x \text{ and } g_\nu = \mathrm{kei}_\nu\, x, & f_\nu &= \mathrm{kei}_\nu\, x \text{ and } g_\nu = -\mathrm{ker}_\nu\, x.
\end{aligned}$$

Other relations for the derivatives are

$$\begin{aligned}
\sqrt{2}\, \mathrm{ber}'_0\, x &= \mathrm{ber}_1\, x + \mathrm{bei}_1\, x, & \sqrt{2}\, \mathrm{bei}'_0\, x &= -\mathrm{ber}_1\, x + \mathrm{bei}_1\, x,\\
\sqrt{2}\, \mathrm{ker}'_0\, x &= \mathrm{ker}_1\, x + \mathrm{kei}_1\, x, & \sqrt{2}\, \mathrm{kei}'_0\, x &= -\mathrm{ker}_1\, x + \mathrm{kei}_1\, x.
\end{aligned}$$

BESSEL FUNCTIONS OF INTEGER ORDER MENU { FTNS BESEL }

FUNCTION	COMMAND	INPUTS	OUTPUTS
$J_n(z)$	**JNOZ**(n,z)	$n \in \mathbb{I}$ $z \in \mathbb{C}$	VALUE
JNOZ uses downward recurrence with normalization – accuracy is about 10 digits.			
$J_0(x)$	**J0OX**(x)	$x \in \mathbb{R}$	VALUE
$J_1(x)$	**J1OX**(x)	$x \in \mathbb{R}$	VALUE
AMS 55 says the accuracy of these fast polynomial approximations is 7 digits.			
$Y_n(x)$	**YNOX**(n,x)	$n \in \mathbb{I}$ $x \in \mathbb{R}$	VALUE
For real x, **YNOX** uses upward recurrence, starting with polynomial approximations **Y0OX** and **Y1OX**. For complex z, use the slower **YNOZ** discussed later in this menu. $Y_n(z)$ has a branch point at the origin, $Y_n(0) = \infty$, so $Y_n(-2)$ is complex. This command has an accuracy of about 7 digits.			
$Y_0(x)$	**Y0OX**(x)	$x \in \mathbb{R}$	VALUE
$Y_1(x)$	**Y1OX**(x)	$x \in \mathbb{R}$	VALUE
AMS 55 says the accuracy of these fast polynomial approximations is 7 digits.			
$I_n(z)$	**INOZ**(n,z)	$n \in \mathbb{I}$ $z \in \mathbb{C}$	VALUE
INOZ uses downward recurrence with normalization – accuracy is about 10 digits.			
$I_0(x)$	**I0OX**(x)	$x \in \mathbb{R}$	VALUE

BESSEL FUNCTIONS OF INTEGER ORDER MENU { FTNS BESEL }

FUNCTION	COMMAND	INPUTS	OUTPUTS
$I_1(x)$	**I1OX**(x)	$x \in \mathbb{R}$	VALUE
AMS 55 says the accuracy of these fast polynomial approximations is 7 digits.			
$K_n(x)$	**KNOX**(n,x)	$n \in \mathbb{I}$ $x \in \mathbb{R}$	VALUE
For real x, **KNOX** uses upward recurrence, starting with polynomial approximations **K0OX** and **K1OX**. For complex z, use the slower **KNOZ** discussed later in this menu. $K_n(z)$ has a branch point at the origin, $K_n(0) = \infty$, so $K_n(-2)$ is complex. This command has an accuracy of about 7 digits.			
$K_0(x)$	**K0OX**(x)	$x \in \mathbb{R}$	VALUE
$K_1(x)$	**K1OX**(x)	$x \in \mathbb{R}$	VALUE
AMS 55 says the accuracy of these fast polynomial approximations is 7 digits.			
$ber_n(x) + i\, bei_n(x)$	**BEN&**(n,x)	$n \in \mathbb{I}$ $x \geq 0$	VALUE
BEN& uses downward recurrence with normalization – accuracy is about 10 digits.			
$ker_n(x) + i\, kei_n(x)$	**KEN&**(n,x)	$n \in \mathbb{I}$ $x > 0$	VALUE
KEN& is evaluated using the relation $ker_n(x) + i\, kei_n(x) = e^{-i\pi n/2} K_n(x\, e^{i\pi/4})$.			

BESSEL FUNCTIONS OF INTEGER ORDER MENU { FTNS BESEL }

FUNCTION	COMMAND	INPUTS	OUTPUTS
$H_n^{(1)}(z)$	**H1NZ**(n,z)	$n \in \mathbb{I}$ $z \in \mathbb{C}$	VALUE
$H_n^{(2)}(z)$	**H2NZ**(n,z)	$n \in \mathbb{I}$ $z \in \mathbb{C}$	VALUE

$$H_v^{(1)}(z) = J_v(z) + i\, Y_v(z)$$
$$H_v^{(2)}(z) = J_v(z) - i\, Y_v(z)$$

The accuracy of the Hankel function commands is set by **YNOZ**. **YNOZ** and **KNOZ** provide about 10 digits, except in the transition region. Around 6 to 12, accuracy drops to about 7 digits.

$Y_n(z)$	**YNOZ**(n,z)	$n \in \mathbb{I}$ $z \in \mathbb{C}$	VALUE

YNOZ uses upward recurrence based on the values of $Y_0(z)$ and $Y_1(z)$. It provides 10–digit accuracy except in the transition region around 6 to 12, where it drops to about 7 digits.

$K_n(z)$	**KNOZ**(n,z)	$n \in \mathbb{I}$ $z \in \mathbb{C}$	VALUE

KNOZ uses upward recurrence based on the values of $K_0(z)$ and $K_1(z)$. It provides 10–digit accuracy except in the transition region around 6 to 12, where it drops to about 7 digits.

$\Gamma(z)$	**GAMMA**(z)	$z \in \mathbb{C}$	VALUE
UP DIRECTORY	**UPDIR**	NONE	PARENT MENU

BESSEL FUNCTIONS OF INTEGER ORDER MENU { FTNS BESEL }

FUNCTION	COMMAND	INPUTS	OUTPUTS

REAL ARGUMENT FASTER HANKEL FUNCTION EVALUATION

When the argument is real, the below programs evaluate the Hankel functions significantly faster than **H1NZ** and **H2NZ**.

H1NX: « → n x « 'JNOZ(n,x) + i × YNOX(n,x)' →NUM » »

H2NX: « → n x « 'JNOZ(n,x) – i × YNOX(n,x)' →NUM » »

See Chapter 9 of Abramowitz, M., and Stegun, I. *Handbook of Mathematical Functions*, AMS 55, Washington D.C., 1964.

Erdelyi, A., Magnus,W., Oberhettinger, F., and Tricomi, F., *Tables of Integral Transforms*, 2 Volumes, New York, McGraw–Hill, 1954.

Luke, Y., *Integrals of Bessel Functions*, New York, McGraw–Hill, 1962.

Oberhettinger, F., *Tables of Bessel Transforms*, New York, Springer–Verlag, 1972.

Watson, G., *A Treatise on the Theory of Bessel Functions*, New York, Cambridge University Press, 1966.

8

SPHERICAL BESSEL FUNCTIONS

INTRODUCTION

This chapter presents the 12 spherical Bessel function commands in the SBESL menu. Like the Bessel functions in Chapter 7 they are of integer order (actually integer and a half), but they do support complex arguments.

ORDINARY SPHERICAL BESSEL FUNCTIONS

Ordinary Spherical Bessel Function of the First Kind: $j_n(z) = \sqrt{\pi/(2z)}\ J_{n+.5}(z)$

Ordinary Spherical Bessel Function of the Second Kind: $y_n(z) = \sqrt{\pi/(2z)}\ Y_{n+.5}(z)$

Ordinary Spherical Bessel Functions of the Third Kind:

$$h_n^{(1)}(z) = j_n(z) + i\ y_n(z) = \sqrt{\pi/(2z)}\ H_{n+.5}^{(1)}(z)$$

$$h_n^{(2)}(z) = j_n(z) - i\ y_n(z) = \sqrt{\pi/(2z)}\ H_{n+.5}^{(2)}(z)$$

MODIFIED SPHERICAL BESSEL FUNCTIONS

Modified Spherical Bessel Function of the First Kind: $i_n^{(1)}(z) = \sqrt{\pi/(2z)}\ I_{n+.5}(z)$

Modified Spherical Bessel Function of the Second Kind: $i_n^{(2)}(z) = \sqrt{\pi/(2z)}\ I_{-n-.5}(z)$

Modified Spherical Bessel Function of the Third Kind: $k_n(z) = \sqrt{\pi/(2z)}\ K_{n+.5}(z)$

AIRY FUNCTIONS

AIOZ and **BIOZ** provide evaluation of the Airy functions Ai(z) and Bi(z).

NOTATION DIFFERENCES

MATHLIB notation differs slightly from AMS 55. AMS 55 denotes $i_n^{(1)}(z)$ by $i_n(z)$ and gives no name to $i_n^{(2)}(z)$.

EVALUATION COMMENTS

The commands in this menu are interrelated in that for negative order they call each other. **SJNZ** provides the basic downward recurrence with normalization operation for the group. The function $j_1(z)$ = [sinc(z) - cos(z)] / z is the normalizing function. If you are so unlucky as to hit upon a value which causes **SJNZ** to *exactly* equal zero, the software will crash with a divide by zero error. The only value of z which I have found is z = 4.49340945791 which is the first zero of $j_1(z)$. If you hit upon another value, simply change the least significant digit of z by ±1. When using the HP 48 solve application to find the roots of $j_n(z)$, the equation 'SJNZ(n,z)=T' for T = 0 works just fine even for n = 1. With an initial guess above or below the first zero, the root solver computes either 4.49340945790 or 4.49340945792 for the first root of $j_1(z)$. There is a very small probability that you will ever have a divide by zero error on the HP 48 since only numbers less than ±1E–499 underflow to zero. However, if you do have a problem, use the following program to change the least significant digit. Since the desired accuracy is only 10 digits, it will not significantly change the result.

```
SJNZP: « → n z « IFERR n z SJNZ THEN DROP2 n z
       .999999999998 × SJNZP END » »
```

SPHERICAL BESSEL FUNCTIONS MENU { FTNS SBESL }

FUNCTION	COMMAND	INPUTS	OUTPUTS
$j_n(z)$	**SJNZ**(n,z)	$n \in \mathbb{I}$ $z \in \mathbb{C}$	VALUE
SJNZ uses downward recurrence with normalization for $n \geq 0$. The accuracy is about 10 digits.			
$y_n(z)$	**SYNZ**(n,z)	$n \in \mathbb{I}$ $z \in \mathbb{C}$	VALUE
SYNZ uses upward recurrence for $n \geq 0$. The accuracy is about 10 digits.			
$h_n^{(1)}(z)$	**SH1Z**(n,z)	$n \in \mathbb{I}$ $z \in \mathbb{C}$	VALUE
$h_n^{(2)}(z)$	**SH2Z**(n,z)	$n \in \mathbb{I}$ $z \in \mathbb{C}$	VALUE
$h_n^{(1)}(z) = j_n(z) + i\, y_n(z)$ $\quad$ $h_n^{(2)}(z) = j_n(z) - i\, y_n(z)$			
$i_n^{(1)}(z)$	**SI1NZ**(n,z)	$n \in \mathbb{I}$ $z \in \mathbb{C}$	VALUE
SI1NZ uses downward recurrence with normalization for $n \geq 0$. The accuracy is about 10 digits.			
$i_n^{(2)}(z)$	**SI2NZ**(n,z)	$n \in \mathbb{I}$ $z \in \mathbb{C}$	VALUE
SI2NZ uses upward recurrence for $n \geq 0$. The accuracy is about 10 digits.			

SPHERICAL BESSEL FUNCTIONS MENU { FTNS SBESL }

FUNCTION	COMMAND	INPUTS	OUTPUTS
$k_n(z)$	**SKNZ**(n,z)	$n \in \mathbb{I}$ $z \in \mathbb{C}$	VALUE
SKNZ uses upward recurrence. The accuracy is about 10 digits.			
Ai(z)	**AIOZ**(z)	$z \in \mathbb{C}$	VALUE
Bi(z)	**BIOZ**(z)	$z \in \mathbb{C}$	VALUE

AIOZ and **BIOZ** evaluate the ascending series defined below. Accuracy degrades for ABS(z) > 5. For asymptotic approximation, see page 123.

$$Ai(z) = 0.355028053888\, f(z) - 0.258819403793\, g(z)$$

$$Bi(z) = \sqrt{3}\,[0.355028053888\, f(z) + 0.258819403793\, g(z)]$$

$$f(z) = \sum_{n=0}^{\infty} 3^n\, (1/3)_n\, \frac{z^{3n}}{(3n)!} \qquad g(z) = \sum_{n=0}^{\infty} 3^n\, (2/3)_n\, \frac{z^{3n+1}}{(3n+1)!},$$

where $(\bullet)_n$ is Pochhammer's symbol defined as

$$(\alpha + 1/3)_0 = 1 \qquad 3^n\, (\alpha + 1/3)_n = (3\alpha + 1)(3\alpha + 4) \ \ldots \ (3\alpha + 3n - 2),$$

where α is arbitrary and n = 1, 2, ...

$\Gamma(z)$	**GAMMA**(z)	$z \in \mathbb{C}$	VALUE
$\psi(z)$	**PSI**(z)	$z \in \mathbb{C}$	VALUE
UP DIRECTORY	**UPDIR**	NONE	PARENT MENU

For definitions, see Chapter 10 of Abramowitz and Stegun, AMS 55.

9

ELLIPTIC INTEGRALS

INTRODUCTION

This chapter presents the 25 commands in the ELLIPI menu. This and the next two chapters deal with elliptic functions and integrals. The singularly periodic trigonometric and hyperbolic functions discussed in Chapter 3 are special cases. A doubly periodic analytic function is called an elliptic function. Seventeen Weierstrass and related elliptic function programs are also available. See Appendix A.

PARAMETER DEFINITIONS

The area of elliptic integrals and related functions is pervaded by a bewildering collection of redundant parameters, different subsets of which are used by different authors. The Math Library provides numerous conversion functions so the user can use the set he or she prefers.

- AMS 55 defines m to be "the parameter" and $m_1 = 1 - m$ to be the "complementary parameter" where $0 \le m, m_1 \le 1$.
- AMS 55 defines k to be the "modulus" and k' to be the "complementary modulus" where by definition $0 \le k, k' \le 1$, $k^2 = m$ and $(k')^2 = m_1$.
- AMS 55 defines α to be the "modular angle" and $\pi/2 - \alpha$ to be the "complementary modular angle" where $\sin \alpha = k$ and $\cos \alpha = k'$.

- AMS 55 defines K and iK ' to be the "quarter periods" where K is the complete elliptic integral of the first kind with parameter m, and K ' is the complete elliptic integral of the first kind with parameter m_1, which we just defined as the complementary parameter.

Clearly, given any one of the eight, the other seven are determined. The following commands are available for the conversions:

M↑M¡	Converts between m and m_1	**SIN**	Converts α (rad) into k
K↑K¡	Converts between k and k'	**ASIN**	Converts k into α (rad)
M→K	Converts m into k	**COS**	Converts α (rad) into k'
M←K	Converts k into m	**ACOS**	Converts k' into α (rad)
α→K¡	Converts α (deg) into k'	**KOKP**	Converts k' into K
α←K¡	Converts k' into α (deg)	**K¡OK¡**	Converts k' into K '
KP→M	Converts k' into m	**KP←M**	Converts m into k'

where deg is degrees and rad is radians. The **K¡** and **KP** symbols that suffix command names are used as a reminder that the input is k' and not k.

AMS 55 expresses its elliptic functions in terms of α and m, whereas in most problems k' is what is specified, and k' is often a very small number. Since

$$\text{COS(ACOS(1E-10)) = 1.048966E-10,}$$

precision is easily lost by working with α and m. Consequently, the Math Library uses k' throughout its commands. However, the above commands allow the user to work with the parameters of his or her choice.

Two related definitions are the nome q and the complementary nome q_1, defined by:

$$q = \text{EXP}[-\pi K ' / K], \qquad q_1 = \text{EXP}[-\pi K / K '].$$

The command **KP→Q**(k') computes nome q, and **KP→Q**(k) computes the complementary nome q_1.

There are two basic forms for the elliptic integrals, which differ by a trigonometric substitution. The trigonometric form is nice for understanding the properties of these integrals, and the polynomial form is the one most useful in applications. The upper limit of the integral in trigonometric form is defined as the "amplitude," which is the angle φ. In polynomial form, the upper limit is x = sin φ. While AMS 55 tabulates using the parameter φ, we prefer to work with the parameter x.

Conversions may be performed with the commands

φ→X	Converts φ (deg) to x	**SIN**	Converts φ (rad) to x
φ←X	Converts x to φ (deg)	**ASIN**	Converts x to φ (rad).

With the provided conversion software, the user can write and store in the VAR directory elliptic functions and integrals that use the parameters of his or her own choice.

JACOBI ELLIPTIC FUNCTIONS

Trigonometric and hyperbolic functions were discussed in Chapter 3. The trigonometric functions have periods of 2π and π:

$$\sin(z + 2n\pi) = \sin(z), \qquad \cos(z + 2n\pi) = \cos(z),$$

$$\tan(z + n\pi) = \tan(z),$$

for $n \in \mathbb{I}$ while the hyperbolic functions have periods of $i2\pi$ and $i\pi$

$$\sinh(z + i2n\pi) = \sinh(z), \qquad \cosh(z + i2n\pi) = \cosh(z),$$

$$\tanh(z + in\pi) = \tanh(z),$$

for $n \in \mathbb{I}$. Consider the integral definition of the inverse sin function

$$\arcsin(z) = \int_0^z \frac{dt}{\sqrt{1 - t^2}}.$$

By a change in integration variable of $t = iw$, we have

$$\arcsin(z) = \int_0^{iz} \frac{idw}{\sqrt{1 + w^2}} = -i\,\mathrm{arcsinh}(iz).$$

Suppose z is real. Then the arcsin integral goes from 0 to z on the real axis, while the arcsinh integral goes from 0 to iz on the imaginary axis. We can thus say that the sin function is periodic on the real axis and the sinh function is periodic on the imaginary axis. Elliptic functions are periodic on both the real and imaginary axes, and the trigonometric and hyperbolic functions are special cases of them.

Consider the following integral which we define to be the function u:

$$u = \int_0^x \frac{dt}{\sqrt{(1 - t^2)(1 - mt^2)}}.$$

It is immediately obvious that for m = 0, u = arcsin(x). With the change of variable t = sin θ, u equals

$$u = \int_0^{\arcsin(x)} \frac{d\theta}{\sqrt{1 - m \sin^2 \theta}} = \int_0^{\varphi} \frac{d\theta}{\sqrt{1 - m \sin^2 \theta}}.$$

The first form is the polynomial form, and the second is the trigonometric form. As the integral on the previous page defined the inverse sin function arcsin, this integral defines the inverse of the Jacobi elliptic function sn(u, k) = sn(u|m) = sn(φ\α), where φ = arcsin x in the notation of AMS 55. Similarly, we have the definitions

$$\mathrm{sn}(u, k) = \sin \varphi, \quad \mathrm{cn}(u, k) = [1 - \mathrm{sn}^2(u, k)]^{1/2} = \cos \varphi, \quad \mathrm{dn}(u, k) = [1 - k^2 \, \mathrm{sn}^2(u, k)]^{1/2},$$

where cn(u, k) is a generalization of the cos function: sn(u, 0) = sin u, and cn(u, 0) = cos u. The nine other Jacobi elliptic functions can be defined in terms of these three. An additional function that is defined is am(u, k) = φ = arcsin(sn(u, k)). Commands for evaluating all of the Jacobi elliptic functions are given in the JACOB menu discussed in Chapter 10.

The theta functions given in the THETA menu discussed in Chapter 11 are important because every one of the Jacobi elliptic functions can be expressed as the ratio of two theta functions.

ELLIPTIC INTEGRALS

The incomplete elliptic integral of the first kind is in fact the above integral u. It defines the inverse of the Jacobi elliptic function sn(u, k) which is denoted by $F(\varphi\backslash\alpha) = F(\varphi|m) = F(x, k) = u$. The complete elliptic integral of the first kind is $F(\pi/2\backslash\alpha) = F(\pi/2|m) = F(1, k)$. The complete integral specifies the quarter periods K and iK' of the elliptic functions since

$$K = F(1, k), \qquad K' = F(1, k').$$

The commands **KOKP** and **K¡OK¡** compute the complete integrals, and **FXKP** computes the incomplete integral. **UZKP** also computes the incomplete integral F(z, k), but it allows z to be complex.

If R(x, P(x)) is a rational function of x, and P^2 is equal to a cubic or quartic polynomial in x, then the integral

$$\int R(x, P(x))\, dx$$

is called an elliptic integral. In general, integrals of rational functions can be factored into a part that can be evaluated by elementary functions and a part which is the sum of integrals that take on one of three canonical forms. These forms are the elliptic integrals of the first, second, and third kind and are the nontrivial part of the numerical evaluation problem. The ELLIPI menu commands evaluate all three, plus several related functions defined in AMS 55.

The incomplete elliptic integral of the second kind is defined as

$$\begin{aligned} E(\varphi\backslash\alpha) &= E(u|m) = E(x, k) = E(u\|k) \\ &= \int_0^u dn^2(w,k)\, dw = m_1 u + m \int_0^u cn^2(w,k)\, dw = u - m \int_0^u sn^2(w,k)\, dw \\ &= \int_0^x (1 - t^2)^{-½} (1 - m t^2)^{½}\, dt = \int_0^{\varphi} (1 - m \sin^2\theta)^{½}\, dt \end{aligned}$$

and is evaluated by the command **EXKP**. $E(\pi/2\backslash\alpha) = E(1, k)$ is the complete elliptic integral of the second kind and is denoted by E in AMS 55. This is evaluated by **EOKP**. AMS 55 also defines $E' = E(1, k')$, which may be evaluated as EOKP(k). Note carefully the AMS 55 shift in notation: $F(\varphi|m)$ versus $E(u|m) = E(u\|k)$, where $u = F(\varphi|m)$.

Two related elliptic integrals in use which are not discussed in AMS 55 are

$$B(\varphi\backslash\alpha) = B(\varphi|m) = B(x, k) = \int_0^{\varphi} \frac{\cos^2\theta \; d\theta}{\sqrt{1 - m \sin^2\theta}}$$

$$D(\varphi\backslash\alpha) = D(\varphi|m) = D(x, k) = \int_0^{\varphi} \frac{\sin^2\theta \; d\theta}{\sqrt{1 - m \sin^2\theta}}$$

and are evaluated by the commands **BXKP** and **DXKP**.

The incomplete elliptic integral of the third kind is defined as

$$\begin{aligned}\Pi(n; \varphi\backslash\alpha) &= \Pi(n; u|m) = \Pi(n, x, k) = \Pi(n, u\|k) \\ &= \int_0^{\varphi} (1 - n \sin^2\theta)^{-1} (1 - m \sin^2\theta)^{-1/2} \, d\theta \\ &= \int_0^{x} (1 - n t^2)^{-1}[(1 - t^2)(1 - m t^2)]^{-1/2} \, dt \\ &= \int_0^{u} [1 - n \, sn^2(w, k)]^{-1} \, dw\end{aligned}$$

with $x = sn(u|m) = sn(u, k) = \sin\varphi$, and is evaluated by Π**NXKP**. The complete elliptic integral of the third kind is $\Pi(n, 1, k) = \Pi(n; \pi/2 \backslash \alpha)$.

ZUKP evaluates Jacobi's zeta function, and **λXKP** computes Heuman's lambda function. **UZKP** computes complex u where complex $z = \sin\varphi$. **SNUK¡** computes complex $z = \sin\varphi$, given complex u. **K¡OK** evaluates the ratio of quarter periods K'/K.

The functions $E(u\|k)$ and $\Pi(n, u\|k)$ are available. They are EUKPS and ΠNUKPS, which are available with defined derivatives. So is ZXKPS. See Appendix A.

Other relations include

$F(x,k) = F(kx, k^{-1})/k \qquad E(u\|k) = k \, E(ku\|k^{-1}) - m_1 \, u \qquad$ for $k > 1$

$F(\varphi| - m) = (1 + m)^{-1/2} \{ K(m/[1 + m]) - F(\pi/2 - \varphi|m/[1+m])\}$

$E(u| - m) = (1 + m)^{1/2} E(u[1+m]^{1/2}|m/[1 + m])$

$\qquad - m \, sn(u[1 + m]^{1/2}|m/[1+m]) \, cd(u[1+m]^{1/2}|m/[1 + m])$

ELLIPTICAL INTEGRALS MENU
{ FTNS ELLIPI }

FUNCTION	COMMAND	INPUTS	OUTPUTS
F(x, k)	**FXKP**(x,k')	x, k' ∈[0,1]	VALUE

$$F(x, k) = \int_0^x \frac{dt}{\sqrt{(1 - t^2)(1 - k^2 t^2)}} \qquad F(.7, .3) = 0.856338019724$$

The accuracy is about 10 digits. F(1, 1) = FXKP(1, 0) = ∞.

FXKP, **EXKP**, **BXKP**, **DXKP**, **KOKP**, **K¡OK¡**, **EOKP**, and **ZXKP** all use the method of Arithmetic–Geometric Mean for the computation. Accuracy degrades near singularities. However, K and K' remain accurate for 1E–499 ≤ k' ≤ .999999999999. Note that K↑KP(.999999999999) = 1.41421356237E–6 > 1E–499.

E(x, k)	**EXKP**(x,k')	x, k' ∈[0,1]	VALUE

$$E(x, k) = \int_0^{\varphi} \sqrt{1 - m \sin^2 \theta}\, d\theta \qquad E(.7, .3) = 0.707444537922$$

The accuracy is about 10 digits.

B(x, k)	**BXKP**(x,k')	x, k' ∈[0,1]	VALUE

$$B(x, k) = \int_0^{\varphi} \frac{\cos^2 \theta \; d\theta}{\sqrt{1 - m \sin^2 \theta}} \qquad B(.7, .3) = 0.692718808952$$

The accuracy is about 10 digits.

ELLIPTICAL INTEGRALS MENU
{ FTNS ELLIPI }

FUNCTION	COMMAND	INPUTS	OUTPUTS
D(x, k)	**DXKP**(x,k')	x, k' ∈[0,1]	VALUE

$$D(x, k) = \int_0^{\varphi} \frac{\sin^2 \theta \; d\theta}{\sqrt{1 - m \sin^2 \theta}} \qquad D(.7, .3) = 0.163619210767$$

The accuracy is about 10 digits. D(1, 1) = DXKP(1, 0) = ∞.

x = sin φ	**φ→X**(φ)	φ ∈[0, 90] IN DEG	VALUE
k' = cos α	**α→Kj**(α)	α ∈[0, 90] IN DEG	VALUE
K = F(1, k)	**KOKP**(k)	k' ∈[0, 1]	VALUE
K' = F(1, k')	**KjOKj**(k)	k' ∈[0, 1}	VALUE

KOKP(.3) = 2.62777333206 KjOKj(.3) = 1.60804861992

E = E(1, k)	**EOKP**(k)	k' ∈[0, 1]	VALUE

EOKP(.3) = 1.09647751739

Π(n, x, k)	**ΠNXKP**(n,x,k')	x, k' ∈[0,1] n ∈**R**	VALUE

$$\Pi(n, x, k) = \int_0^{x} \frac{dt}{(1 - n t^2)\sqrt{(1 - t^2)(1 - m t^2)}} \qquad nx^2 < 1$$

ELLIPTICAL INTEGRALS MENU
{ FTNS ELLIPI }

FUNCTION	COMMAND	INPUTS	OUTPUTS

Observe that the integrand has a simple pole at $t = 1/\sqrt{n}$. If you integrate through it, you will obtain a meaningless complex number. This rather complicated command is an implementation of the equations in Section 17.7 of AMS 55. Two comments are in order. First, in the circular case where $m < n < 1$, there is a pole in the equations at $n = m$, and as one approaches the pole the equations become very sensitive to numerical error. **ΠNXKP** gives a "Bad Argument Value" error whenever it branches into this circular case and $\sqrt{1-n}/k' \geq 0.99999995$. However, the $n = m$ and $n < m$ cases can be evaluated since different equations are used. The differences in these cases are usually in the 7th decimal of the input. This should hopefully protect the user from bad outputs. Also, Equation 17.7.21 is missing a parenthesis, and the second term should read:

$$-\tfrac{1}{2}\sqrt{n}\ \ln\{(1+\sqrt{n}\ \sin\varphi)(1-\sqrt{n}\ \sin\varphi)^{-1}\}.$$

ΠNXKP reproduces all the numbers in the examples and tables of AMS 55. Here are the examples.

$\Pi(1/16;45°\backslash 30°) = \Pi NXKP(1/16\ ,\ \phi\rightarrow X(45)\ ,\ \alpha\rightarrow K_j(30)) = 0.813845432979$
$\Pi(1/16;90°\backslash 30°) = \Pi NXKP(1/16\ ,\ 1\ ,\ \alpha\rightarrow K_j(30)) = 1.74305520342$
$\Pi(5/8;45°\backslash 30°) = \Pi NXKP(5/8\ ,\ \phi\rightarrow X(45)\ ,\ \alpha\rightarrow K_j(30)) = 0.921129573345$
$\Pi(5/8;90°\backslash 30°) = \Pi NXKP(5/8\ ,\ 1\ ,\ \alpha\rightarrow K_j(30)) = 2.80098923985$
$\Pi(5/4;45°\backslash 30°) = \Pi NXKP(5/4\ ,\ \phi\rightarrow X(45)\ ,\ \alpha\rightarrow K_j(30)) = 1.13213569491$
$\Pi(-1/4;45°\backslash 30°) = \Pi NXKP(-1/4\ ,\ \phi\rightarrow X(45)\ ,\ \alpha\rightarrow K_j(30)) = 0.769872357412$

$Z(u\|m) = Z(u\|k)$	**ZUKP**(u,k')	$u \in \mathbb{C}$ $k' \in [0, 1]$	VALUE

ZUKP is Jacobi's zeta function. Since the notation used by AMS 55 is very confusing, we present some tutorial discussion so that the relationships and commands are clear. This is given on the next page.

ELLIPTICAL INTEGRALS MENU
{ FTNS ELLIPI }

FUNCTION	COMMAND	INPUTS	OUTPUTS

Let us begin by defining two programs. The first evaluates the Jacobi zeta function using the command **ZUKP**, while the second program evaluates the definition of Jacobi's zeta function 17.4.27 in AMS 55.

ZXKP: « → X k « X k FXKP k ZUKP » »

ZXK¡: « → X k 'EXKP(X,k) – FXKP(X,k) × EOKP(k) / KOKP(k)' » ,

where k in these programs is the complementary modulus k' = cos α and X = sin φ. These two programs yield the same value for the Jacobi zeta function defined by

$Z(\varphi\backslash\alpha) = Z(x, k) = E(x, k) - E \times F(x, k)/K = Z(u\,|m) = Z(u\|k)$ where u = F(x, k).

Table 17.7 in AMS 55 is a tabulation of $Z(\varphi\backslash\alpha)\ K = Z(x, k)\ K$. In particular, the following program evaluates the values found in that table:

ZTBL: « → φ α 'ZXKP(φ→X(φ) , α→K¡(α)) × KOKP(α→K¡(α))' »

where ZXKP can be replaced by ZXK¡. The accuracy is about 7 digits. Also

$Z(u\,|m) = Z(u\|k) = E(u\,|m) - u\ E/K = E(u\|k) - u\ E/K$ where u = F(x, k)

By definition, Z(u‖k) is the logarithmic derivative of the Jacobi theta function Θ(u, k).

$$Z(u\|k) = \Theta'(u, k)/\Theta(u, k) = \frac{\partial}{\partial u} \ln \Theta(u, k)$$

and can be related to the Neville theta functions and the Jacobi elliptic functions by

ELLIPTICAL INTEGRALS MENU { FTNS ELLIPI }

FUNCTION	COMMAND	INPUTS	OUTPUTS

$$Z(u\|k) = \frac{\vartheta_s'(u,k)}{\vartheta_s(u,k)} - \frac{cn(u,k)\ dn(u,k)}{sn(u,k)}, \qquad Z(u\|k) = \frac{\vartheta_c'(u,k)}{\vartheta_c(u,k)} + \frac{dn(u,k)\ sn(u,k)}{cn(u,k)},$$

$$Z(u\|k) = \frac{\vartheta_d'(u,\ k)}{\vartheta_d(u,\ k)} + k^2\ \frac{sn(u,\ k)\ cn(u,\ k)}{dn(u,\ k)}, \qquad Z(u\|k) = \frac{\vartheta_n'(u,\ k)}{\vartheta_n(u,\ k)},$$

which corrects equation 16.34.3 in AMS 55.

The Jacobi zeta function Z(u‖k) can be expressed as a power series in nome q:

$$Z(u\|k) = Z(u\,|m) = \frac{2\pi}{K} \sum_{s=1}^{\infty} \frac{q^{\,s}}{1 - q^{2s}} \sin\,(\pi su/K),$$

where q and K are functions of the modulus k. However, **ZUKP** uses the method of Arithmetic–Geometric Mean based on Landen's transformation for computation.

ZUKP(.3,.5) = 0.124982130858

ZUKP((.3,1E–100),.5) = (0.124982130858, 3.74313467818E–101)

While the **ZUKP** command has an accuracy of about 7 digits on the real number line, accuracy does degrade with the increasing imaginary part of u. The above power series may provide more accurate results for complex u. It is command **TNUK¡** described in the THETA menu discussed in Chapter 11.

ELLIPTICAL INTEGRALS MENU { FTNS ELLIPI }

FUNCTION	COMMAND	INPUTS	OUTPUTS
$\Lambda_0(x, k)$	**λXKP**(x,k')	x, k' ∈[0,1]	VALUE

λXKP is Heuman's lambda function $\Lambda_0(\varphi\backslash\alpha) = \Lambda_0(x, k) = F(x, k')/K' + K\,Z(x, k')\,\pi/2$.

$\Lambda_0(.3 , .6)$ = λXKP(.3 , K↑KP(.6)) = 0.271095891688,

where Z(x, k) is defined above with the **ZUKP** command. The accuracy is about 7 digits.

φ = asin x	φ←**X**(x)	x ∈[0,1]	VALUE IN DEG
α = acos k'	α←**K**$_1$(k')	k' ∈[0,1]	VALUE IN DEG
$k = \sqrt{m}$	**M**→**K**(m)	m ∈[0,1]	VALUE
$m = k^2$	**M**←**K**(k)	k ∈[0,1]	VALUE
$k' = \sqrt{(1 - k^2)}$	**K**↑**KP**(k OR k')	k, k' ∈[0,1]	VALUE
$m' = m_1 = 1 - m$	**M**↑**M**$_1$(m OR m')	m, m' ∈[0,1]	VALUE
$\Delta(z, k)$	**ΔZKP**(z,k')	z ∈ℂ k' ∈[0,1]	VALUE

ΔZKP evaluates $\Delta(\varphi\backslash\alpha) = \Delta(z, k) = dn(u, k) = \sqrt{(1 - k^2 z^2)}$, defined by 16.1.5 in AMS 55.

Δ((.2 , .3) , .5) = ΔXKP((.2 , .3) , K↑KP(.5)) = (1.0063409827, −1.49054845801E−2)

where z = sn(u, k) = sin φ, and u = F(x, k). The accuracy is about 10 digits.

K'/K = F(1, k')/F(1, k)	**K**$_1$**OK**(k')	k' ∈(0,1)	VALUE

ELLIPTICAL INTEGRALS MENU
{ FTNS ELLIPI }

FUNCTION	COMMAND	INPUTS	OUTPUTS
INVERSE OF **K¡OK**	**IK¡OK**(α,β)	α = K'/K β ∈ (0,1) β = BEST GUESS	k ∈ (0,1)
K¡OK computes ratio of quarter periods. k = 0 or 1 results in an error.			
$m = 1 - (k')^2$	**KP→M**(k')	k' ∈ [0,1]	VALUE
u(z, k)	**UZKP**(z,k')	z ∈ ℂ k' ∈ [0,1]	VALUE
z(u, k) = sn(u, k)	**SNUK¡**(u,k')	u ∈ ℂ k' ∈ [0,1]	VALUE

UZKP computes the complex elliptic integral $F(z, k) = u(z, k) = sn^{-1}(z, k)$. **SNUK¡** computes its complex inverse z(u, k) = sn(u, k). **UZKP** uses **FXKP** when z is real, and much slower complex integration otherwise. See the command **CINTG** in Chapters 4 and 6. **SNUK¡** evaluates the complex Jacobian elliptic function sn(u, k) = sin φ discussed in Chapter 10.

SNUK¡(UZKP((.15,.4) , .5) , .5) = (0.15, 0.399999999999)

The accuracy is about 10 digits. Do not integrate through any poles.

UP DIRECTORY	**UPDIR**	NONE	PARENT MENU

ANALYTIC CONTINUATION

The four incomplete elliptic integral commands **FXKP**, **EXKP**, **BXKP**, and **DXKP**, plus λ**XKP**, do accept complex values of x. However, accuracy degrades from 10 digits at arg x = 0 to 0 digits for |arg x| = π. You can get a good estimate of accuracy for the complex region of interest by comparing the outputs of **UZKP** and **FXKP**. The following relations on the next page are available:

ELLIPTICAL INTEGRALS MENU
{ FTNS ELLIPI }

FUNCTION	COMMAND	INPUTS	OUTPUTS

$$F(-x, k) = -F(x, k) \qquad E(-x, k) = -E(x, k)$$

and for $\tan\theta = \sinh\varphi$ $\quad F(i\varphi\backslash\alpha) = i\,F(\theta\backslash\pi/2 - \alpha)$

$$E(i\varphi\backslash\alpha) = -i\,E(\theta\backslash\pi/2 - \alpha) + i\,F(\theta\backslash\pi/2 - \alpha) + i\tan\theta\sqrt{(1 - \cos^2\alpha\,\sin^2\theta)}$$

$$F(s\pi \pm \varphi|m) = 2sK \pm F(\varphi|m) \qquad E(u + 2K\|k) = E(u\|k) + 2E$$

$$E(u + 2iK'\|k) = E(u\|k) + 2i(K' - E')$$

$$D(z, k) = [\,F(z, k) - E(z, k)\,]/m \qquad B(z, k) = [\,E(z, k) - m_1\,F(z, k)\,]/m$$

For definitions see Chapters 16 and 17 of Abramowitz and Stegun, *Handbook of Mathematical Functions*, AMS 55, 1964.

Erdelyi, A., Magnus,W., Oberhettinger, F., and Tricomi, F., *Higher Transcendental Functions*, Volume 2, New York, McGraw–Hill, 1953.

Gradshteyn, I., and Ryzhik, I., *Table of Integrals, Series, and Products*, New York, Academic Press, 1980.

Hancock, H., *Theory of Elliptic Functions*, New York, Dover, 1958.

Korn, G., and Korn, T., *Mathematical Handbook For Scientists and Engineers*, New York, McGraw–Hill, 1961.

Magnus, W., Oberhettinger, F., Soni, R., *Formulas and Theorems for the Special Functions of Mathematical Physics*, New York, Springer–Verlag, 1966.

Morse, P., and Feshbach, H., *Methods of Theoretical Physics*, New York, McGraw–Hill, 1953.

Whittaker, E., and Watson, G., *Modern Analysis*, New York, Cambridge Univ. Press, 1969.

10

JACOBI ELLIPTIC FUNCTIONS

INTRODUCTION

This chapter presents the 36 commands in the JACOB menu. These commands evaluate the Jacobi elliptic functions. See Chapter 9 for an introduction to elliptic functions and definitions.

JACOBIAN ELLIPTIC FUNCTIONS

The basic three functions sn(u, k), cn(u, k), and dn(u, k) were defined in Chapter 9 in terms of elliptic integrals. The other nine functions are defined in terms of these as:

$$\text{cd}(u,\,k) = \frac{\text{cn}(u,\,k)}{\text{dn}(u,\,k)} \qquad \text{dc}(u,\,k) = \frac{\text{dn}(u,\,k)}{\text{cn}(u,\,k)} \qquad \text{ns}(u,\,k) = \frac{1}{\text{sn}(u,\,k)}$$

$$\text{sd}(u,\,k) = \frac{\text{sn}(u,\,k)}{\text{dn}(u,\,k)} \qquad \text{nc}(u,\,k) = \frac{1}{\text{cn}(u,\,k)} \qquad \text{ds}(u,\,k) = \frac{\text{dn}(u,\,k)}{\text{sn}(u,\,k)}$$

$$\text{nd}(u,\,k) = \frac{1}{\text{dn}(u,\,k)} \qquad \text{sc}(u,\,k) = \frac{\text{sn}(u,\,k)}{\text{cn}(u,\,k)} \qquad \text{cs}(u,\,k) = \frac{\text{cn}(u,\,k)}{\text{sn}(u,\,k)}$$

$\text{sn}(u, k) = k^{-1}\,\text{sn}(ku, k^{-1})$ $\quad \text{cn}(u, k) = \text{dn}(ku, k^{-1})$ $\quad \text{dn}(u, k) = \text{cn}(ku, k^{-1})$ $\quad$ for $k > 1$

If $m > 0$, $\mu = m/[1 + m]$, and $\eta = \sqrt{}(1 + m)$, then $0 < \mu < 1$ and

$\text{sn}(u|\,-m) = \text{sd}(\eta u|\mu)/\eta$, $\quad \text{cn}(u|\,-m) = \text{cd}(\eta u|\mu)$, $\quad \text{dn}(u|\,-m) = \text{nd}(\eta u|\mu)$.

All of these functions are available in the JACOB menu. They are computed by the method of Arithmetic–Geometric Mean. The group sn, cn, and dn are first computed with **NUKP**. Then the group conversion commands **N→D**, **N→C**, **N→S** are used to convert the "n" group into the "d", "c", or "s" groups, using the above equations. Finally, the individual function is isolated. The function am(u, k) is also available as the command **AMUKP**. The below table shows how all the trigonometric and hyperbolic functions are special cases of the Jacobian elliptic functions.

FUNCTION	m = 0	m = 1	FUNCTION	m = 0	m = 1
sn(u|m)	sin u	tanh u	dc(u|m)	sec u	1
cn(u|m)	cos u	sech u	nc(u|m)	sec u	cosh u
dn(u|m)	1	sech u	sc(u|m)	tan u	sinh u
cd(u|m)	cos u	1	ns(u|m)	csc u	coth u
sd(u|m)	sin u	sinh u	ds(u|m)	csc u	csch u
nd(u|m)	1	cosh u	cs(u|m)	cot u	csch u

CONVERSION COMMANDS

Conversion commands are again available between k, k', m, and m_1.

QUARTER PERIODS

Since all of the above Jacobian elliptic functions allow u to be complex, it is important to know the periods and location of poles. These can be determined with the commands **KOKP** and **K¡OK¡**, which compute K and K ', respectively.

ARITHMETIC–GEOMETRIC MEAN

AGMN provides user access to one of the MATHLIB internal arithmetic–geometric mean commands for experimenting with applications of Landen's transformation.

JACOBIAN ELLIPTIC FUNCTIONS MENU { FTNS JACOB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
sn(u|m) = sn(u, k)	**SNUK$_j$**(u,k')	u ∈ **C** k' ∈ [0,1]	sn
cn(u|m) = cn(u, k)	**CNUK$_j$**(u,k')	u ∈ **C** k' ∈ [0,1]	cn
dn(u|m) = dn(u, k)	**DNUK$_j$**(u,k')	u ∈ **C** k' ∈ [0,1]	dn
cd(u|m) = cd(u, k)	**CDUK$_j$**(u,k')	u ∈ **C** k' ∈ [0,1]	cd
sd(u|m) = sd(u, k)	**SDUK$_j$**(u,k')	u ∈ **C** k' ∈ [0,1]	sd
nd(u|m) = nd(u, k)	**NDUK$_j$**(u,k')	u ∈ **C** k' ∈ [0,1]	nd
dc(u|m) = dc(u, k)	**DCUK$_j$**(u,k')	u ∈ **C** k' ∈ [0,1]	dc
nc(u|m) = nc(u, k)	**NCUK$_j$**(u,k')	u ∈ **C** k' ∈ [0,1]	nc
sc(u|m) = sc(u, k)	**SCUK$_j$**(u,k')	u ∈ **C** k' ∈ [0,1]	sc
ns(u|m) = ns(u, k)	**NSUK$_j$**(u,k')	u ∈ **C** k' ∈ [0,1]	ns
ds(u|m) = ds(u, k)	**DSUK$_j$**(u,k')	u ∈ **C** k' ∈ [0,1]	ds
cs(u|m) = cs(u, k)	**CSUK$_j$**(u,k')	u ∈ **C** k' ∈ [0,1]	cs
These 12 commands evaluate the 12 Jacobian functions. Accuracy is about 10 digits for either real or complex arguments.			
SET XFORM	**N→D**(sn,cn,dn)	sn, cn, dn	cd, sd, nd
SET XFORM	**N←D**(cd,sd,nd)	cd, sd, nd	sn, cn, dn
SET XFORM	**N→C**(sn,cn,dn)	sn, cn, dn	dc, nc, sc
SET XFORM	**N←C**(dc,nc,sc)	dc, nc, sc	sn, cn, dn
SET XFORM	**N→S**(sn,cn,dn)	sn, cn, dn	ns, ds, cs
SET XFORM	**N←S**(ns,ds,cs)	ns, ds, cs	sn, cn, dn

JACOBIAN ELLIPTIC FUNCTIONS MENU { FTNS JACOB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
The above 6 commands perform transformations between the sets.			
COMPUTE N SET	**NUKP**(u,k')	$u \in \mathbb{C}$ $k' \in [0,1]$	sn, cn, dn
COMPUTE D SET	**DUKP**(u,k')	$u \in \mathbb{C}$ $k' \in [0,1]$	cd, sd, nd
COMPUTE C SET	**CUKP**(u,k')	$u \in \mathbb{C}$ $k' \in [0,1]$	dc, nc, sc
COMPUTE S SET	**SUKP**(u,k')	$u \in \mathbb{C}$ $k' \in [0,1]$	ns, ds, cs
These four commands compute each of the sets.			

EXAMPLES FROM AMS 55

nc(1.9965|.64) = NCUK$_j$(1.9965,.6) = –1392.11113637 (which corrects the answer in AMS 55)
dn(.2|.19) = DNUK$_j$(.2,KP←M(.19)) = DNUK$_j$(.2,.9) = .996252664327
dn(.2|.81) = DNUK$_j$(.2,KP←M(.81)) = .984056028964
cn(.2|.81) = CNUK$_j$(.2,KP←M(.81)) = .980278536974
dc(.672|.36) = DCUK$_j$(.672,KP←M(.36)) = 1.17401900743
sn(.672|.36) = SNUK$_j$(.672,.8) = .609519691787
cs(.5360162|.09) = CSUK$_j$(.5360162,KP←M(.09)) = 1.69180832853
sn(.61802|.5) = SNUK$_j$(.61802,KP←M(.5)) = .564575752946
sc(.61802|.5) = SCUK$_j$(.61802,.707106781187) = .68401814101

u(z, k)	**UZKP**(z,k')	$z \in \mathbb{C}$ $k' \in [0,1]$	VALUE

UZKP computes the complex elliptic integral F(z, k) = u(z, k) = $sn^{-1}(z, k)$. **SNUK$_j$** computes its complex inverse z(u, k) = sn(u, k).

JACOBIAN ELLIPTIC FUNCTIONS MENU { FTNS JACOB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
UZKP uses **FXKP** when z is real, and much slower complex integration otherwise. See the command **CINTG** in Chapters 4 and 6. SNUK$_j$(UZKP((.15,.4) , .5) , .5) = (0.15, 0.399999999999) The accuracy is about 10 digits. Do not integrate through any poles.			
am(u|m) = am(u, k)	**AMUKP**(u,k')	u $\in \mathbb{C}$ k' $\in$[0,1]	VALUE
am(u, k) = φ = arcsin(sn(u, k))			
k' = cos α	$\alpha \rightarrow$**K$_j$**(α)	$\alpha \in$[0,90]	VALUE
α = acos k'	$\alpha \leftarrow$**K$_j$**(k')	k' $\in$[0,1]	VALUE
k = $\sqrt{m}$	**M**$\rightarrow$**K**(m)	m $\in$[0,1]	VALUE
m = k^2	**M**$\leftarrow$**K**(k)	k $\in$[0,1]	VALUE
k' = $\sqrt{(1 - k^2)}$	**K**$\uparrow$**KP**(k OR k')	k, k' $\in$[0,1]	VALUE
m' = m_1 = 1 – m	**M**$\uparrow$**M$_i$**(m OR m_i)	m, m_1 $\in$[0,1]	VALUE
K = F(1, k)	**KOKP**(k')	k' $\in$[0,1]	VALUE
K' = F(1, k')	**K$_j$OK$_j$**(k')	k' $\in$[0,1]	VALUE
KOKP(.3) = 2.62777333206 K$_j$OK$_j$(.3) = 1.60804861992			
m = 1 – $(k')^2$	**KP**$\rightarrow$**M**(k')	k' $\in$[0,1]	VALUE
k' = $\sqrt{(1 - m)}$	**KP**$\leftarrow$**M**(m)	m $\in$[0,1]	VALUE

JACOBIAN ELLIPTIC FUNCTIONS MENU { FTNS JACOB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
ARITHMETIC–GEOMETRIC MEAN	**AGMN**(u,k')	$u \in \mathbf{C}$ $k' \in (0,1]$	4: C LIST 3: φ LIST 2: B LIST 1: C/A LIST

Starting with $a_0 = 1$, $b_0 = k'$, and $c_0 = k$, **AGMN** computes the squences of a_k, b_k, c_k, and c_k/a_k for k = 1, 2, . . ., n such that $|c_n| < 1E{-}11$. The sequences are:

$$a_{k+1} = (a_k + b_k)/2 \qquad b_{k+1} = \sqrt{a_k b_k} \qquad c_{k+1} = (a_k - b_k)/2$$

A LIST = { a_0 a_1 ... a_n } B LIST = { b_0 b_1 ... b_n } C LIST = { c_0 c_1 ... c_n },

where the A LIST is computed as VECTD(C LIST , C/A LIST). The φ LIST is created by defining $\varphi_n = 2^n a_n u$ and successively computing φ_{n-1}, φ_{n-2}, . . ., φ_0 to obtain the φ LIST = { φ_0 φ_1 ... φ_n } using the recurrence relation

$$\sin(2\varphi_{k-1} - \varphi_k) = \frac{c_k}{a_k} \sin \varphi_k.$$

For more detail see pages 571, 577, 578, 598, and 599 of AMS 55.

UP DIRECTORY	**UPDIR**	NONE	PARENT MENU

For definitions, see Chapter 16 of Abramowitz and Stegun, *Handbook of Mathematical Functions*, AMS 55, 1964.

11

THETA AND RELATED FUNCTIONS

INTRODUCTION

This chapter presents the 42 theta and related functions in the THETA menu. The theta functions can be used to compute not only the Jacobian elliptic functions, but also the Weierstrass elliptic and related functions discussed in Chapter 18 of AMS 55. Commands are given for evaluating the Jacobian theta and eta functions as well as Neville's theta functions. It is assumed that the reader is familiar with the definitions and relations presented in Chapters 9 and 10.

Lerch's transcendent $\Phi(z, s, \alpha)$ is discussed at the end of this menu. The Weierstrass elliptic and related functions, as well as Lerch's transcendent, are available. See Appendix A.

METHOD OF COMPUTATION

All of the theta functions in the THETA menu are evaluated by computing the first 40 terms of their expansion in the nome q defined in Chapter 9. The region of definition is $0 < k, k' < 1$. Evaluation with a modulus of 0 or 1 will eventually result in some error. The range of accurate evaluation is $1E{-}499 \leq k' \leq 0.999999999999$. See the **KP→Q** command.

Theta functions with complex nome q are also available.

NOTATION

AMS 55 uses the notation $\vartheta(z, q)$ where the nome q is a function of k. We prefer the notation $\vartheta(z, k)$ since the evaluation of the nome is internal to many of the commands. As with all the other elliptic and related functions in the Math Library, k', not k, is what is used for the command input. The nome q is evaluated by the command **KP→Q**. Thus, KP→Q(k') = q(k), and KP→Q(k) = $q_1(k)$. However, the more general $\vartheta_n(z\|q)$ for n = 1, 2, 3, and 4 are available in this menu along with their logarithmic derivatives. This allows evaluation in the cases where the nome q is complex. AMS 55 also uses the notation $\vartheta_n(z|m) = \vartheta_n(z\|q) = \vartheta_n(z, k)$ for n = 1, 2, 3, and 4. In some references, ϑ_4 is denoted as ϑ_0 or just ϑ.

CONVERSION COMMANDS

Conversion commands are available between the parameters k, k', m, m_1, and α. They are also available for conversion between degrees and radians, as well as between φ and x.

JACOBIAN THETA FUNCTIONS

Jacobi's theta functions correspond to the sigma functions of Weierstrass. Both are entire functions and hence not elliptic functions. However, they are very useful for computing elliptic functions. The four basic theta functions may be defined by the Fourier expansions given by 16.27.1–4 of AMS 55:

$$\vartheta_1(z, k) = \vartheta_1(z\|q) = 2\, q^{1/4} \sum_{n=0}^{\infty} (-1)^n\, q^{n(n+1)} \sin(2n + 1)z,$$

$$\vartheta_2(z, k) = \vartheta_2(z\|q) = 2\, q^{1/4} \sum_{n=0}^{\infty} q^{n(n+1)} \cos(2n + 1)z,$$

$$\vartheta_3(z, k) = \vartheta_3(z\|q) = 1 + 2 \sum_{n=1}^{\infty} q^{n^2} \cos 2nz,$$

$$\vartheta_4(z, k) = \vartheta_4(z\|q) = 1 + 2 \sum_{n=1}^{\infty} (-1)^n\, q^{n^2} \cos 2nz.$$

Other authors use different notation such as given by equations 16.31.1–4 in AMS 55:

$\Theta(u, k) = \Theta(u|m) = \vartheta_4(v, k) = \theta_4(v/\pi, k)$, $\Theta_1(u, k) = \Theta_1(u|m) = \vartheta_3(v, k) = \theta_3(v/\pi, k)$,

$H(u, k) = H(u|m) = \vartheta_1(v, k) = \theta_1(v/\pi, k)$, $H_1(u, k) = H_1(u|m) = \vartheta_2(v, k) = \theta_2(v/\pi, k)$,

where $v = (\pi u)/(2K)$, Θ, Θ_1, H, and H_1 are Jacobi's notation for the theta and eta functions, and $\theta_1(w, k)$ through $\theta_4(w, k)$ for $w = u/(2K)$ are alternative definitions not presented in AMS 55 but heavily used in other references. One clue to which definition is being used are the identities for the partial derivative:

$$\vartheta_1'(0, k) = \vartheta_2(0, k)\,\vartheta_3(0, k)\,\vartheta_4(0, k), \qquad \theta_1'(0, k) = \pi\,\theta_2(0, k)\,\theta_3(0, k)\,\theta_4(0, k).$$

Thus, $\theta_1(w, k) = \vartheta_1(w\pi, k)$, $\theta_2(w, k) = \vartheta_2(w\pi, k)$, $\theta_3(w, k) = \vartheta_3(w\pi, k)$, $\theta_4(w, k) = \vartheta_4(w\pi, k)$, the derivative $\theta_1'(w, k) = \pi\,\vartheta_1'(w\pi, k)$, and similarly for the other derivatives. The Math Library does not evaluate the θ functions directly, but programs using these relations are given at the end of this menu to compute them in terms of the ϑ theta functions.

Also observe that $\partial\,\Theta(u, k)/\partial u = \Theta'(u, k) = (\pi/2K)\,\vartheta_4'(v, k)$, and similarly for the other functions.

NEVILLE'S THETA FUNCTIONS

AMS 55 also gives formulas for Neville's theta functions, which are normalized versions of ϑ_1 through ϑ_4. The following relations are true:

$$\vartheta_3(0, k) = \theta_3(0, k) = \sqrt{2K/\pi}, \qquad \vartheta_3(0, k') = \theta_3(0, k') = \sqrt{2K'/\pi},$$

$$\vartheta_2(0, k) = \theta_2(0, k) = \sqrt{2kK/\pi}, \qquad \vartheta_4(0, k) = \theta_4(0, k) = \sqrt{2k'K/\pi},$$

$$\frac{\vartheta_2(0, k)}{\vartheta_3(0, k)} = \frac{\theta_2(0, k)}{\theta_3(0, k)} = \sqrt{k}, \qquad \frac{\vartheta_4(0, k)}{\vartheta_3(0, k)} = \frac{\theta_4(0, k)}{\theta_3(0, k)} = \sqrt{k'},$$

$$\vartheta_1'(0, k) = \sqrt{\frac{2^3\,K^3\,k\,k'}{\pi^3}}, \qquad \theta_1'(0, k) = \pi\,\vartheta_1'(0, k).$$

The four Neville theta functions are defined by

$$\vartheta_s(u, k) = \frac{2K\ \vartheta_1(v, k)}{\pi\ \vartheta_1'(0, k)}, \qquad \vartheta_c(u, k) = \frac{\vartheta_2(v, k)}{\vartheta_2(0, k)},$$
$$\vartheta_d(u, k) = \frac{\vartheta_3(v, k)}{\vartheta_3(0, k)}, \qquad \vartheta_n(u, k) = \frac{\vartheta_4(v, k)}{\vartheta_4(0, k)},$$

where again $v = (\pi u)/(2K)$, and the above definition of $\vartheta_s(u, k)$ corrects Equation 16.36.6 in AMS 55 to agree with Definition 16.36.1 in AMS 55 using 16.31.3. Also observe that $\vartheta_s'(0, k) = 1$.

RELATIONS TO OTHER FUNCTIONS

If p and r are any of the two letters s, c, n, d, then the Jacobian elliptic function denoted as pr(u, k) is given by

$$pr(u, k) = \frac{\vartheta_p(u, k)}{\vartheta_r(u, k)}.$$

For example:

$$sn(u, k) = \frac{\vartheta_s(u, k)}{\vartheta_n(u, k)}.$$

The Jacobi zeta function $Z(u|m) = Z(u\|k) = \partial \ln \Theta(u,k)/\partial u$. See Chapter 9 for more relationships between the Jacobi zeta function, Neville theta functions, and Jacobi elliptic functions. Since few of these functions are actually tabulated, it is through these numerous relationships that the Math Library software has been verified.

COMPLEX NOME q

As noted above, in general the nome can be complex, but is not arbitrary. Theta functions with complex nome arise in differential equations, Weierstrass elliptic functions with negative discriminant, and the theory of modular functions. The theta function series expansions only converge for realistic values of $|q| < 1$. A collection of useful Weierstrass parameter conversion programs is available. See Appendix A.

ELLIPTIC THETA FUNCTIONS MENU { FTNS THETA }

FUNCTION	COMMAND	INPUTS	OUTPUTS
x = sin φ	φ→**X**(φ)	φ ∈[0, 90]	VALUE
φ = asin x	φ←**X**(x)	x ∈[0, 1]	VALUE
u = F(z, k)	**UZKP**(z,k')	z ∈ℂ k' ∈[0,1]	VALUE
z(u, k)	**SNUK**ⁱ(u,k')	u ∈ℂ k' ∈[0,1]	VALUE

UZKP computes the complex elliptic integral F(z, k) = u(z, k) = sn^{-1}(z, k). **SNUK**ⁱ computes its complex inverse z(u, k) = sn(u, k). **UZKP** uses **FXKP** when z is real, and much slower complex integration otherwise. See the command **CINTG** in Chapters 4 and 6. **SNUK**ⁱ evaluates the complex Jacobian elliptic function sn(u, k) = sin φ discussed in Chapter 10.

SNUKⁱ(UZKP((.15,.4) , .5) , .5) = (0.15, 0.399999999999)

The accuracy is about 10 digits. Do not integrate through any poles.

k' = cos α	α→**K**ⁱ(α)	α ∈[0, 90] IN DEG	VALUE
α = acos k'	α←**K**ⁱ(k')	k' ∈[0,1]	VALUE IN DEG
ϑ_s(u, k)	θ**SUK**ⁱ(u,k')	u ∈ℂ k' ∈(0,1)	VALUE

$$\vartheta_s(u,\ k) = \sqrt{\frac{2\pi\sqrt{q}}{k\ k'\ K}} \sum_{n=0}^{\infty} (-1)^n\ q^{n(n+1)}\ \sin(2n\ +\ 1)v \qquad v = \pi u/(2K)$$

θSUKⁱ((1,1.3) , .5) = (1.31317174007, 1.19787536389)

ELLIPTIC THETA FUNCTIONS MENU { FTNS THETA }

FUNCTION	COMMAND	INPUTS	OUTPUTS
$\vartheta_c(u, k)$	θCUKj(u,k')	$u \in \mathbb{C}$ $k' \in (0,1)$	VALUE

$$\vartheta_c(u, k) = \sqrt{\frac{2\pi\sqrt{q}}{k\,K}} \sum_{n=0}^{\infty} q^{n(n+1)} \cos(2n + 1)v \qquad v = \pi u/(2K)$$

θCUKj((1,1.3) , .5) = (1.06225552965, –.774472185718)

$\vartheta_d(u, k)$	θDUKj(u,k')	$u \in \mathbb{C}$ $k' \in (0,1)$	VALUE

$$\vartheta_d(u, k) = \sqrt{\frac{\pi}{2\,K}} \left\{ 1 + 2 \sum_{n=1}^{\infty} q^{n^2} \cos 2nv \right\} \qquad v = \pi u/(2K)$$

θDUKj((1,1.3) , .5) = (.908073863295, –.472905740602)

$\vartheta_n(u, k)$	θNUKj(u,k')	$u \in \mathbb{C}$ $k' \in (0,1)$	VALUE

$$\vartheta_n(u, k) = \sqrt{\frac{\pi}{2\,k'\,K}} \left\{ 1 + 2 \sum_{n=1}^{\infty} (-1)^n q^{n^2} \cos 2nv \right\} \qquad v = \pi u/(2K)$$

θNUKj((1,1.3) , .5) = (1.12411321044, .667485006988)

ELLIPTIC THETA FUNCTIONS MENU { FTNS THETA }

FUNCTION	COMMAND	INPUTS	OUTPUTS
u(ε, α) k'(α)	εα→U(ε,α)	ε, α ∈ (0, 90) ε AND α IN DEG	2: ε K/90 1: k'
ε(u, k') α(k')	εα←U(u,k')	u ∈ ℝ k' ∈ (0,1)	2: 90 u/K 1: α IN DEG

These commands do the parameter conversions for Tables 16.1 & 2 in AMS 55.

$$\varepsilon\alpha\text{→}U(25,\ 15) = \begin{cases} 2\text{: } .443928333920 = u_{\$} \\ 1\text{: } .965925826289 = k_{\$}' \end{cases}$$

Using the values $u_{\$}$ and $k_{\$}'$ as inputs, we can compute the tables in AMS 55.

$\theta SUK_j(u_{\$},k_{\$}') = .429981306357$ $\theta NUK_j(u_{\$},k_{\$}') = 1.00312296844$
$TSUK_j(u_{\$},k_{\$}') = 2.10786899898$ $TNUK_j(u_{\$},k_{\$}') = 1.31242775141E{-}2$

Jacobi Θ(u, k)	θUKP(u,k')	u ∈ ℂ k' ∈ (0,1)	VALUE

$$\Theta(u,\ k) = \vartheta_4(\pi u/(2K),\ k)$$

θUKP((1,1.3) , .5) = (.931346790133, .553022607462)

Jacobi Θ_1(u, k)	θ1UKj(u,k')	u ∈ ℂ k' ∈ (0,1)	VALUE

$$\Theta_1(u,\ k) = \vartheta_3(\pi u/(2K),\ k)$$

θ1UKj((1,1.3) , .5) = (1.06398999473, –.554103577682)

ELLIPTIC THETA FUNCTIONS MENU { FTNS THETA }

FUNCTION	COMMAND	INPUTS	OUTPUTS
Jacobi H(u, k)	**HUKP**(u,k')	$u \in \mathbb{C}$ $k' \in (0,1)$	VALUE
$H(u, k) = \vartheta_1(\pi u/(2K), k)$ HUKP((1,1.3) , .5) = (1.01248410589, .923588080498)			
Jacobi $H_1(u, k)$	**H1UK$_j$**(u,k')	$u \in \mathbb{C}$ $k' \in (0,1)$	VALUE
$H_1(u, k) = \vartheta_2(\pi u/(2K), k)$ H1UK$_j$((1,1.3) , .5) = (1.15827233828, –.844476384867)			
K = F(1, k)	**KOKP**(k')	$k' \in [0,1]$	VALUE
K' = F(1, k')	**K$_j$OK$_j$**(k')	$k' \in [0,1]$	VALUE
KOKP(.3) = 2.62777333206 K$_j$OK$_j$(.3) = 1.60804861992			
$\vartheta_1(z\|q)$	**θ1ZQ**(z,q)	$z, q \in \mathbb{C}$	VALUE
$\vartheta_1(z\|q) = 2\, q^{1/4} \sum_{n=0}^{\infty} (-1)^n\, q^{n(n+1)} \sin(2n+1)z$ θ1ZQ((1,1.3) , KP→Q(.5)) = (1.76723769408, 1.18807415237)			

ELLIPTIC THETA FUNCTIONS MENU { FTNS THETA }

FUNCTION	COMMAND	INPUTS	OUTPUTS
$\vartheta_2(z\|q)$	**θ2ZQ**(z,q)	z, q ∈ **C**	VALUE
$\vartheta_2(z\|q) = 2\, q^{1/4} \sum_{n=0}^{\infty} q^{n(n+1)} \cos(2n+1)z$ θ2ZQ((1,1.3) , KP→Q(.5)) = (.957780179854, −1.57455611577)			
$\vartheta_3(z\|q)$	**θ3ZQ**(z,q)	z, q ∈ **C**	VALUE
$\vartheta_3(z\|q) = 1 + 2 \sum_{n=1}^{\infty} q^{n^2} \cos 2nz$ θ3ZQ((1,1.3) , KP→Q(.5)) = (.510224328668, −1.03713037683)			
$\vartheta_4(z\|q)$	**θ4ZQ**(z,q)	z, q ∈ **C**	VALUE
$\vartheta_4(z\|q) = 1 + 2 \sum_{n=1}^{\infty} (-1)^n q^{n^2} \cos 2nz$ θ4ZQ((1,1.3) , KP→Q(.5)) = (1.47693527836, 1.05199635225)			
$k' = \sqrt{(1-k^2)}$	**K↑KP**(k OR k')	k, k' ∈[0,1]	VALUE
$m' = m_1 = 1 - m$	**M↑M$_1$**(m OR m')	m, m' ∈[0,1]	VALUE

ELLIPTIC THETA FUNCTIONS MENU { FTNS THETA }

FUNCTION	COMMAND	INPUTS	OUTPUTS
$\vartheta_s'(u, k)/\vartheta_s(u, k)$	**TSUK$_j$(u,k')**	$u \in \mathbb{C}$ $k' \in (0,1)$	VALUE

$$\frac{\vartheta_s'(u, k)}{\vartheta_s(u, k)} = \frac{\pi}{2K} \frac{\vartheta_1'(v, k)}{\vartheta_1(v, k)} \qquad v = \frac{\pi u}{2K}$$

TSUK$_j$((1,1.3) , .5) = (.293926034567, –.715709042857)

$\vartheta_c'(u, k)/\vartheta_c(u, k)$	**TCUK$_j$(u,k')**	$u \in \mathbb{C}$ $k' \in (0,1)$	VALUE

$$\frac{\vartheta_c'(u, k)}{\vartheta_c(u, k)} = \frac{\pi}{2K} \frac{\vartheta_2'(v, k)}{\vartheta_2(v, k)} \qquad v = \frac{\pi u}{2K}$$

TCUK$_j$((1,1.3) , .5) = (–.278078208109, –.684889156809)

$\vartheta_d'(u, k)/\vartheta_d(u, k)$	**TDUK$_j$(u,k')**	$u \in \mathbb{C}$ $k' \in (0,1)$	VALUE

$$\frac{\vartheta_d'(u, k)}{\vartheta_d(u, k)} = \frac{\pi}{2K} \frac{\vartheta_3'(v, k)}{\vartheta_3(v, k)} \qquad v = \frac{\pi u}{2K}$$

TDUK$_j$((1,1.3) , .5) = (–.592082367372, –.388749606913)

ELLIPTIC THETA FUNCTIONS MENU { FTNS THETA }

FUNCTION	COMMAND	INPUTS	OUTPUTS
$\vartheta_n'(u, k)/\vartheta_n(u, k)$	**TNUK$_j$(u,k')**	$u \in \mathbb{C}$ $k' \in (0,1)$	VALUE

$$\frac{\vartheta_n'(u, k)}{\vartheta_n(u, k)} = \frac{\pi}{2K} \frac{\vartheta_4'(v, k)}{\vartheta_4(v, k)} \qquad v = \frac{\pi u}{2K}$$

TNUK$_j$((1,1.3) , .5) = (.715301239463, –.318307098105)

NOME q(k)	**KP→Q(k')**	k' ∈ (0,1)	VALUE

$q(k) = \text{EXP}[- \pi K' / K]$ $\quad q_1(k) = \text{EXP}[- \pi K / K'] = q(k')$

KP→Q(.5) = 8.57957337009E–2

KP→Q(.999999999999) = 1.24999999989E–13 KP→Q(1E–499) = .995719458905

These two extreme values are accurate. 1E–499 ≤ k' ≤ .999999999999 is the range of accurate evaluation.

Note that K↑KP(.999999999999) = 1.41421356237E–6.

$\vartheta_1'(0, k)$	**Dθ1K$_j$(k')**	k' ∈ (0,1)	VALUE

$$\vartheta_1'(0, k) = \sqrt{\frac{2^3 K^3 k k'}{\pi^3}}$$

Dθ1K$_j$(.5) = 1.05852086498

ELLIPTIC THETA FUNCTIONS MENU { FTNS THETA }

FUNCTION	COMMAND	INPUTS	OUTPUTS
$\vartheta_1'(u\|q)/\vartheta_1(u\|q)$	**T1UQ**(u,q)	u, q ∈ ℂ	VALUE

$$\frac{\vartheta_1'(u|q)}{\vartheta_1(u|q)} = \cot u + 4\sum_{n=1}^{\infty}\frac{q^{2n}}{1-q^{2n}}\sin 2nu$$

T1UQ((1,1.3) , KP→Q(.5)) = (.293890873496, −1.02539686221)

$\vartheta_2'(u\|q)/\vartheta_2(u\|q)$	**T2UQ**(u,q)	u, q ∈ ℂ	VALUE

$$\frac{\vartheta_2'(u|q)}{\vartheta_2(u|q)} = -\tan u + 4\sum_{n=1}^{\infty}(-1)^n\frac{q^{2n}}{1-q^{2n}}\sin 2nu$$

T2UQ((1,1.3) , KP→Q(.5)) = (−.339826244986, −.985896117324)

$\vartheta_3'(u\|q)/\vartheta_3(u\|q)$	**T3UQ**(u,q)	u, q ∈ ℂ	VALUE

$$\frac{\vartheta_3'(u|q)}{\vartheta_3(u|q)} = 4\sum_{n=1}^{\infty}(-1)^n\frac{q^{n}}{1-q^{2n}}\sin 2nu$$

T3UQ((.7,.5) , KP→Q(.5)) = (−.470804467752, −.153824793726)

ELLIPTIC THETA FUNCTIONS MENU { FTNS THETA }

FUNCTION	COMMAND	INPUTS	OUTPUTS
$\vartheta_4'(u\|q)/\vartheta_4(u\|q)$	**T4UQ**(u,q)	u, q $\in \mathbb{C}$	VALUE

$$\frac{\vartheta_4'(u|q)}{\vartheta_4(u|q)} = 4\sum_{n=1}^{\infty}\frac{q^n}{1-q^{2n}}\sin 2nu$$

T4UQ((.7,.5) , KP→Q(.5)) = (.538068742, –3.85746574343E–2)

HP RAD → DEG	**R→D**(r)	r $\in \mathbb{R}$	VALUE
HP DEG → RAD	**D→R**(d)	d $\in \mathbb{R}$	VALUE
$\mathrm{Ln}[\vartheta_1(\alpha+\beta,k)/\vartheta_1(\alpha-\beta,k)]$	**LNθ1**(α,β,k')	α, β $\in \mathbb{C}$ k' $\in$ (0,1)	VALUE

$$\ln\frac{\vartheta_1(\alpha+\beta,k)}{\vartheta_1(\alpha-\beta,k)} = \ln\frac{\sin(\alpha+\beta)}{\sin(\alpha-\beta)} + 4\sum_{n=1}^{\infty}\frac{1}{n}\frac{q^{2n}}{1-q^{2n}}\sin 2n\alpha\sin 2n\beta$$

LNθ1((.1,.2) , (.4,.5) .5) = (.746453329441, –3.04241369601)

$\mathrm{Ln}[\vartheta_2(\alpha+\beta,k)/\vartheta_2(\alpha-\beta,k)]$	**LNθ2**(α,β,k')	α, β $\in \mathbb{C}$ k' $\in$ (0,1)	VALUE

$$\ln\frac{\vartheta_2(\alpha+\beta,k)}{\vartheta_2(\alpha-\beta,k)} = \ln\frac{\cos(\alpha+\beta)}{\cos(\alpha-\beta)} + 4\sum_{n=1}^{\infty}\frac{(-1)^n}{n}\frac{q^{2n}}{1-q^{2n}}\sin 2n\alpha\sin 2n\beta$$

LNθ2((.1,.2) , (.4,.5) .5) = (.148665261996, –.247135067716)

ELLIPTIC THETA FUNCTIONS MENU { FTNS THETA }

FUNCTION	COMMAND	INPUTS	OUTPUTS
$\ln[\vartheta_3(\alpha+\beta,k)/\vartheta_3(\alpha-\beta,k)]$	**LNθ3(α,β,k')**	$\alpha,\beta\in\mathbb{C}$ $k'\in(0,1)$	VALUE

$$\ln\frac{\vartheta_3(\alpha+\beta,k)}{\vartheta_3(\alpha-\beta,k)}=4\sum_{n=1}^{\infty}\frac{(-1)^n}{n}\frac{q^n}{1-q^{2n}}\sin 2n\alpha\sin 2n\beta$$

LNθ3((.1,.2) , (.4,.5) .5) = (5.06376342136E–2, –.173164633243)

$\ln[\vartheta_4(\alpha+\beta,k)/\vartheta_4(\alpha-\beta,k)]$	**LNθ4(α,β,k')**	$\alpha,\beta\in\mathbb{C}$ $k'\in(0,1)$	VALUE

$$\ln\frac{\vartheta_4(\alpha+\beta,k)}{\vartheta_4(\alpha-\beta,k)}=4\sum_{n=1}^{\infty}\frac{1}{n}\frac{q^n}{1-q^{2n}}\sin 2n\alpha\sin 2n\beta$$

LNθ4((.1,.2) , (.4,.5) .5) = (1.35910093279E–2, .256261444948)

$m = 1 - k'^2$	**KP→M(k')**	$k'\in[0,1]$	VALUE
UP DIRECTORY	**UPDIR**	NONE	PARENT MENU

EVALUATION OF THE $\theta_n(w,k)$ THETA FUNCTIONS

Programs to evaluate these four functions and their logarithmic derivatives are given on the next page. See the equations on page 99. The k input to these programs is, as usual, the complementary modulus k'.

ELLIPTIC THETA FUNCTIONS MENU { FTNS THETA }

FUNCTION	COMMAND	INPUTS	OUTPUTS

θ1ZKj: ≪ → w k ≪ 'θ1ZQ(π×w , KP→Q(k))' →NUM ≫ ≫

θ2ZKj: ≪ → w k ≪ 'θ2ZQ(π×w , KP→Q(k))' →NUM ≫ ≫

θ3ZKj: ≪ → w k ≪ 'θ3ZQ(π×w , KP→Q(k))' →NUM ≫ ≫

θ4ZKj: ≪ → w k ≪ 'θ4ZQ(π×w , KP→Q(k))' →NUM ≫ ≫

T1UKj: ≪ → w k ≪ 'π×T1UQ(π×w , KP→Q(k))' →NUM ≫ ≫

T2UKj: ≪ → w k ≪ 'π×T2UQ(π×w , KP→Q(k))' →NUM ≫ ≫

T3UKj: ≪ → w k ≪ 'π×T3UQ(π×w , KP→Q(k))' →NUM ≫ ≫

T4UKj: ≪ → w k ≪ 'π×T4UQ(π×w , KP→Q(k))' →NUM ≫ ≫

For definitions see Chapters 16 of Abramowitz and Stegun, *Handbook of Mathematical Functions*, AMS 55, 1964.

Erdelyi, A., Magnus,W., Oberhettinger, F., and Tricomi, F., *Higher Transcendental Functions*, Volume 2, New York, McGraw–Hill, 1953.

Gradshteyn, I., and Ryzhik, I., *Table of Integrals, Series, and Products*, New York, Academic Press, 1980.

Hancock, H., *Theory of Elliptic Functions*, New York, Dover, 1958.

Jahnke, E., and Emde, F., *Tables of Functions*, New York, Dover, 1945.

Korn, G., and Korn, T., *Mathematical Handbook For Scientists and Engineers*, New York, McGraw–Hill, 1961.

ELLIPTIC THETA FUNCTIONS MENU { FTNS THETA }

FUNCTION	COMMAND	INPUTS	OUTPUTS

Magnus, W., Oberhettinger, F., Soni, R., *Formulas and Theorems for the Special Functions of Mathematical Physics*, New York, Springer–Verlag, 1966.
Morse, P., and Feshbach, H., *Methods of Theoretical Physics*, New York, McGraw–Hill, 1953.
Whittaker, E., and Watson, G., *Modern Analysis*, New York, Cambridge Univ. Press, 1969.

LERCH'S TRANSCENDENT $\Phi(z, s, \alpha)$

Lerch's transcendent is a generalization of the Riemann zeta and polylogarithm functions. Riemann showed that $\zeta(z)$ can be expressed as an integral of $\theta_3(w, k)$.

$$\Phi(z, s, \alpha) = \sum_{n=0}^{\infty} (\alpha + n)^{-s} z^n \qquad |z| < 1$$

$$\Phi(z, s, \alpha) = \frac{1}{\Gamma(s)} \int_0^{\infty} \frac{t^{s-1} e^{-\alpha t}}{1 - z e^{-t}} dt = \frac{1}{\Gamma(s)} \int_0^{\infty} \frac{t^{s-1} e^{(1-\alpha)t} dt}{e^t - z}$$

RE $\alpha > 0$, and either z is not on the branch cut from 1 to ∞ with RE $s > 0$ or $z = 1$ and RE $s > 1$.

Lerch's transcendent is available. See Appendix A. Special cases include the generalized Riemann zeta function $\zeta(s, \alpha) = \Phi(1, s, \alpha)$, Euler's dilogarithm $\mathfrak{L}_2(z) = z\,\Phi(z, 2, 1)$, the polylogarithm (Jonquiere) function $\mathfrak{L}_n(z) = z\,\Phi(z, n, 1)$, the Fermi–Dirac function $F_k(\eta) = e^{\eta}\,\Gamma(k + 1)\,\Phi(-e^{\eta}, k + 1, 1)$, and $\Phi(z, 1, \alpha) = \alpha^{-1}\,{}_2F_1(1, \alpha; 1 + \alpha; z)$, $|z| < 1$.

$$2^{1-s}\,\Gamma(s)\,\zeta(s)\cos\frac{s\pi}{2} = \pi^s\,\zeta(1 - s) \qquad 2^s\,\Gamma(1 - s)\,\zeta(1 - s)\sin\frac{s\pi}{2} = \pi^{1-s}\,\zeta(s)$$

12

CONFLUENT HYPERGEOMETRIC FUNCTIONS

INTRODUCTION

This chapter presents the 18 commands in the confluent hypergeometric function menu. Besides solving very general differential equations and providing a method of evaluating numerous integrals, they offer a means of analytically continuing many functions. Table 13.6 in AMS 55 shows the numerous special cases of these functions. The CHYPR menu contains commands for evaluating both of the confluent hypergeometric functions in addition to many special cases. Generalized hypergeometric functions are discussed at the end. Additional special cases are in the GAMA, ERROR, and BESEL menus and the parabolic cylinder function menu PCLDR.

CONFLUENT HYPERGEOMETRIC FUNCTIONS

There are two basic hypergeometric functions in use. The first is Kummer's function $M(a, b, z) = {}_1F_1(a; b; z)$:

$$M(a, b, z) = \sum_{n=0}^{\infty} \frac{(a)_n \, z^n}{(b)_n \, n!} \qquad (a)_n = a(a+1)(a+2) \ldots (a+n-1), \quad (a)_0 = 1,$$

where arguments a, b, z are complex, and $(a)_n = \Gamma(a + n)/\Gamma(a)$ is Pochhammer's symbol. Note that $(a)_n$ is well defined for a = negative integer even though neither gamma function itself is defined. Command **POCH** in the MISC menu provides numerically stable evaluation of $(a)_n$ for all complex values of a and all integer values of n.

The second basic function is Tricomi's U(a, b, z) defined by

$$U(a, b, z) = \frac{\pi}{\sin \pi b}\left\{\frac{M(a, b, z)}{\Gamma(1+a-b)\Gamma(b)} - z^{1-b}\,\frac{M(1+a-b, 2-b, z)}{\Gamma(a)\Gamma(2-b)}\right\}.$$

Observing all the possible ways U(a, b, z) might be undefined as b goes to an integer, you will be surprised to learn that U(a, b, z) is well defined for all b, while F(a, b, z) is undefined or singular for negative integer values of b. Clearly, many cases must be considered when evaluating these functions in the complex plane, requiring numerous sets of equations. The commands **MABZ** and **UABZ** evaluate these two functions. **DNM** and **DNU** evaluate their nth derivatives. When the logarithmic solution of U(a, b, z) is required, the computation can take over a minute, due to the evaluating of three infinite series of digamma functions, in addition to evaluating **MABZ** twice.

INCOMPLETE GAMMA FUNCTIONS

INCGC and **INCγC** provide analytically continued evaluation of the incomplete gamma functions $\Gamma(a, z)$ and $\gamma(a, z)$ defined in Chapter 5.

BESSEL AND KELVIN FUNCTIONS

JVOZ, **YVOZ**, **IVOZ**, **KVOZ**, **H1VZ**, and **H2VZ** provide analytically continued evaluation of the Bessel functions $J_v(z)$, $Y_v(z)$, $I_v(z)$, $K_v(z)$, $H_v^{(1)}(z)$, and $H_v^{(2)}(z)$. **BEV&** and **KEV&** provide evaluation of the Kelvin functions $\mathrm{ber}_v(x) + i\,\mathrm{bei}_v(x)$, and $\mathrm{ker}_v(x) + i\,\mathrm{kei}_v(x)$ defined in Chapter 7 for real order and non-negative argument.

WHITTAKER AND COULOMB WAVE FUNCTIONS

MKμZ and **WKμZ** provide Whittaker's functions $M_{\kappa,\mu}(z)$ and $W_{\kappa,\mu}(z)$. Programs are also given for evaluating the Coulomb wave functions $F_L(\eta, \rho)$ and $G_L(\eta, \rho)$.

ACCURACY CONSIDERATIONS

The nicest thing one can say about the hypergeometric functions is that almost every common function is a special case of them. The nastiest thing one can say is that except for a very limited range of arguments, they are useless for evaluating these functions. **MABZ** reproduces the numbers in AMS 55 and Slater to 10-digit accuracy (the tables are less than 10 digits), which of course is on the real number line.

However, the process of analytically extending these functions to arbitrary complex arguments does create numerical problems for large argument values, even though MATHLIB uses the best asymptotic expansions available. For example, consider computing $I_n(x)$ using M(a, b, z) with the equation given on page 118.

$I_5(10)$ = 777.188286404 RESULT = 777.1883936

$I_{15}(10)$ = .104371490706 RESULT = .249472373707

$I_5(100)$ = 9.47009387301E41 RESULT = 9.47009387308E41

$I_{15}(100)$ = 3.47368638152E41 RESULT = 3.47368638152E41

There is very good agreement between the numbers for z = 100, but very poor agreement for n = 15 and z = 10. This problem is due to the limited computational precision of the HP 48. *Twelve digits are not enough*. Consequently, a few test cases in the complex region of interest are worth making. Accurate cases include M(a, b, z) and U(a, b, z) for a equal to a negative integer or zero and U(a, b, z) for a + 1 - b equal to a negative integer or zero. These cases are polynomials. U(1, b, z) or U(a, a, z) for non-negative integer a is also accurate since it is evaluated using **ENOZ**. Apart from these special cases, **UABZ** uses **MABZ** in its evaluation and large magnitudes of argument a generally result in erroneous results from **MABZ** as in the $I_{15}(10)$ example above. Accurate evaluation of the hypergeometric functions for large arguments requires unlimited large precision arithmetic which is not practical on the HP 48. A useful test program for measuring the accuracy of **MABZ** and **UABZ** is given below. It compares the result from **KVOZ** with $K_v(z)$ evaluated using **UABZ** for v, z ∈ **C**.

```
« → v z « v z KVOZ v .5 + 2 v × 1 + z 2 × UABZ p
  √ × z EXP / 2 z × v ^ × » »
```

Within the range of convergence, hypergeometric functions provide analytically extended methods of evaluating $I_v(z)$ for complex v and z as well as numerous other functions. Since there are no complex plane tabulations for M(a, b, z) and U(a, b, z), special cases are also the primary means of software verification off the real number line. For sufficiently small arguments (less than 1), they provide a full 10 digits of accuracy in the complex plane. But for larger arguments, there are no general asymptotic formulas that work all the time for all combinations of parameters.

Also note that since U(a, b, z) is not analytically continued beyond $(-\pi, \pi]$, if we used it to evaluate **YVOZ**, **H1VZ**, and **H2VZ**, the values would be only valid in three of the four complex plane quadrants. Other equations are used for all the Bessel functions.

CONFLUENT HYPERGEOMETRIC FUNCTIONS MENU {FTNS CHYPR}

FUNCTION	COMMAND	INPUTS	OUTPUTS
M(a, b, z)	**MABZ**(a,b,z)	a, b, z $\in \mathbf{C}$	VALUE
NTH DERIVATIVE M(a, b, z)	**DNM**(a,b,z,n)	a, b, z $\in \mathbf{C}$ n $\in \mathbf{N}$	VALUE

$$\frac{d^n}{dz^n} M(a, b, z) = \frac{(a)_n}{(b)_n} M(a + n, b + n, z)$$

Neither is numerically defined for b = 0, −1, −2, . . . with the exception MABZ(a, a, z) = EXP(z) for a ≥ 0. The commands are not numerically stable near these poles and singularities. See the accuracy comments on page 115.

U(a, b, z)	**UABZ**(a,b,z)	a, b, z $\in \mathbf{C}$	VALUE
NTH DERIVATIVE U(a, b, z)	**DNU**(a,b,z,n)	a, b, z $\in \mathbf{C}$ n $\in \mathbf{N}$	VALUE

$$\frac{d^n}{dz^n} U(a, b, z) = (-1)^n (a)_n U(a + n, b + n, z)$$

Numerical instability may be observed in the neighborhood of b = 0, ±1, ±2, ... |z| < 1E−20 results in a bad argument error since these cases are not programmed. In the region 6 < |z| < 20 the accuracy of **MABZ** and **UABZ** can be very poor.

Γ(a, z)	**INCGC**(a,z)	a, z $\in \mathbf{C}$	VALUE

$$\Gamma(a, z) = \begin{cases} z^a E_{1-a}(z) & a = 0, -1, -2, \ldots \\ e^{-z} U(1 - a, 1 - a, z) \end{cases}$$

Numerical instability may be observed in the neighborhood of a = 0, −1, −2, ...

CONFLUENT HYPERGEOMETRIC FUNCTIONS MENU {FTNS CHYPR}

FUNCTION	COMMAND	INPUTS	OUTPUTS
$\gamma(a, z)$	INCγC(a,z)	a, z $\in \mathbf{C}$	VALUE

$$\gamma(a, z) = z^a M(a, a + 1, -z)/a$$

Undefined at and numerically unstable in the neighborhood of a = 0, –1, –2, ...

$J_v(z)$	JVOZ(v,z)	v, z $\in \mathbf{C}$	VALUE

$$J_v(z) = \frac{(z/2)^v}{\Gamma(v + 1)} e^{-iz} M(v + .5, 2v + 1, 2iz) \qquad v \neq -1, -1.5, -2, \ldots$$

$J_5(10)$ = –.234061528189 JVOZ(5,10) = –.234061527314
$J_{15}(10)$ = 4.50797314374E–3 JVOZ(15,10) = 4.50797314367E–3
$J_5(100)$ = –7.41957369642E–2 JVOZ(5,100) = –7.4195736964E–2
$J_{15}(100)$ = 1.51981212243E–2 JVOZ(15,100) = 1.519812122222E–2

$Y_v(z)$	YVOZ(v,z)	v, z $\in \mathbf{C}$	VALUE

$$Y_v(z) = -\frac{(2z)^v}{\sqrt{\pi}} [e^{i(v\pi - z)} U(v + .5, 1 + 2v, 2iz) + e^{-i(v\pi - z)} U(.5 + v, 1 + 2v, -2iz)]$$

$Y_5(10)$ = .1354030477 YVOZ(5,10) = .135403046641
$Y_{15}(10)$ = –6.364745877 YVOZ(15,10) = –6.36474587695
$Y_5(100)$ = –2.948019628E–2 YVOZ(5,100) = –2.94801962796E–2
$Y_{15}(100)$ = 7.879068695E–2 YVOZ(15,100) = 7.87906869454E–2

Accurate for v equal to a negative integer, but unstable in the neighborhood.

CONFLUENT HYPERGEOMETRIC FUNCTIONS MENU {FTNS CHYPR}

FUNCTION	COMMAND	INPUTS	OUTPUTS
$I_v(z)$	IVOZ(v,z)	v, z ∈ C	VALUE

$$I_v(z) = \frac{(z/2)^v}{\Gamma(v+1)} e^{-z} M(v + .5, 2v + 1, 2z) \qquad v \neq -1, -1.5, -2, \ldots$$

$I_5(10)$ = 777.188286404 IVOZ(5,10) = 777.188286404
$I_{15}(10)$ = .104371490706 IVOZ(15,10) = .104371490706
$I_5(100)$ = 9.47009387301E41 IVOZ(5,100) = 9.470093873E41
$I_{15}(100)$ = 3.47368638152E41 IVOZ(15,100) = 3.47368638152E41

$K_v(z)$	KVOZ(v,z)	v, z ∈ C	VALUE

$$K_v(z) = \sqrt{\pi}\,(2z)^v\, e^{-z}\, U(v + .5, 2v + 1, 2z)$$

$K_5(10)$ = 5.75418499E–5 KVOZ(5,10) = 5.75418499828E–5
$K_{15}(10)$ = .2656563849 KVOZ(15,10) = .26565638374
$K_5(100)$ = 5.27325611E–45 KVOZ(5,100) = 5.2732561133E–45
$K_{15}(100)$ = 1.42348325E–44 KVOZ(15,100) = 1.42348325115E–44

Accurate for v equal to a negative integer, but unstable in the neighborhood.

$H_v^{(1)}(z)$	H1VZ(v,z)	v, z ∈ C	VALUE

$$H_v^{(1)}(z) = 2\,\frac{(2z)^v}{\sqrt{\pi}}\, e^{-i\pi(v+.5)+iz}\, U(v + .5, 2v + 1, -2iz)$$

CONFLUENT HYPERGEOMETRIC FUNCTIONS MENU {FTNS CHYPR}

FUNCTION	COMMAND	INPUTS	OUTPUTS
$H_v^{(2)}(z)$	**H2VZ**(v,z)	$v, z \in \mathbf{C}$	VALUE
$H_v^{(2)}(z) = 2\,\frac{(2z)^v}{\sqrt{\pi}}\,e^{i\pi(v+.5)-iz}\,U(v + .5, 2v + 1, 2iz)$ Note that $H_v^{(1)}(z)$ and $H_v^{(2)}(z)$ are actually evaluated from the relations $H_v^{(1)}(z) = J_v(z) + i\,Y_v(z), \qquad H_v^{(2)}(z) = J_v(z) - i\,Y_v(z).$			
$ber_v(x) + i\,bei_v(x)$	**BEV&**(v,x)	$v \in \mathbf{R}$ $x \geq 0$	VALUE
$ber_v(x) + i\,bei_v(x) = J_v(xe^{3\pi i/4})$			
$ker_v(x) + i\,kei_v(x)$	**KEV&**(v,x)	$v \in \mathbf{R}$ $x > 0$	VALUE
$ker_v(x) + i\,kei_v(x) = .5i\pi\,H_v^{(1)}(xe^{3\pi i/4})$			
$M_{\kappa,\mu}(z)$	**MKμZ**(κ,μ,z)	$\kappa, \mu, z \in \mathbf{C}$	VALUE
$M_{\kappa,\mu}(z) = z^{\mu+.5}\,e^{-z/2}\,M(\mu - \kappa + .5, 1 + 2\mu, z)$			
$W_{\kappa,\mu}(z)$	**WKμZ**(κ,μ,z)	$\kappa, \mu, z \in \mathbf{C}$	VALUE
$W_{\kappa,\mu}(z) = z^{\mu+.5}\,e^{-z/2}\,U(\mu - \kappa + .5, 1 + 2\mu, z)$			
$\Gamma(z)$	**GAMMA**(z)	$z \in \mathbf{C}$	VALUE

CONFLUENT HYPERGEOMETRIC FUNCTIONS MENU {FTNS CHYPR}

FUNCTION	COMMAND	INPUTS	OUTPUTS

BEHAVIOR OF U(a, b, z) AT THE BRANCH CUT

The negative real axis is the branch cut of U(a, b, z). The discontinuity equals

$$U(a,b,-\xi+i0) - U(a,b,-\xi-i0) = \begin{cases} -\dfrac{2\pi i}{\Gamma(a)\,\Gamma(2-b)}\,\xi^{1-b}\,M(a-b+1,2-b,-\xi) & b\neq 2,\ 3,\ \ldots \\ (-1)^{b-2}\,\dfrac{2\pi i}{\Gamma(a-b+1)\Gamma(b)}\,M(a,\ b,\ -\xi) & b=2,\ 3,\ \ldots \end{cases}$$

for $\xi > 0$ which corrects Erdelyi, Volume I, page 263, Equation 16. This proves that **YVOZ**, **H1VZ**, and **H2VZ** would be only valid in three of the four quadrants if U(a, b, z) is used for the computation since the "i" in the argument z of U(a, b, z) is a 90–degree rotation in the complex plane.

U(1.5, 2, (–2,1E–499)) – U(1.5, 2, (–2,–1E–499)) = (0, .914051587489)

U(1.5, 2.3, (–2,1E–499)) – U(1.5, 2.3, (–2,–1E–499)) = (0, .820913698703)

Observe that the branch cut vanishes when a is a negative integer or zero. U is also one–valued when b = n + 1, n = 1, 2, . . ., and a is one of the integers 1, 2, . . ., n, so that U becomes a polynomial in z^{-1}.

COULOMB WAVE FUNCTIONS

Coulomb wave functions are the solution of the differential equation

$$\frac{d^2w}{d\rho^2} + \left[1 - \frac{2\eta}{\rho} - \frac{L(L+1)}{\rho^2}\right] w = 0.$$

The general solution is $w = C_1\,F_L(\eta,\,\rho) + C_2\,G_L(\eta,\,\rho)$, where C_1 and C_2 are constants.

CONFLUENT HYPERGEOMETRIC FUNCTIONS MENU {FTNS CHYPR}

FUNCTION	COMMAND	INPUTS	OUTPUTS

$F_L(\eta, \rho)$ is the regular Coulomb wave function, and $G_L(\eta, \rho)$ is the irregular (logarithmic) Coulomb wave function. For sufficiently small arguments, these may be evaluated in terms of the confluent hypergeometric functions M(a, b, z) and U(a, b, z). The equations are

$$F_L(\eta, \rho) = 2^L e^{-\pi\eta/2 - i\rho} \frac{|\Gamma(L+1+i\eta)|}{\Gamma(2L+2)} \rho^{L+1} M(L+1-i\eta, 2L+2, i2\rho)$$

$$G_L(\eta, \rho) = i\, 2^{L+1} e^{\pi\eta/2 - i\rho + i\pi L} \frac{\Gamma(L+1-i\eta)}{|\Gamma(L+1-i\eta)|} \rho^{L+1} U(L+1-i\eta, 2L+2, i2\rho) + i\, F_L(\eta, \rho)$$

For large arguments, use the equations in AMS 55. When $\eta = 0$, these equations reduce to ρ times a spherical Bessel function:

$$F_L(0, \rho) = \rho\, j_L(\rho) \qquad G_L(0, \rho) = -\rho\, y_L(\rho),$$

which corrects 14.6.6 in AMS 55.

FLηρ: ≪ → L η ρ ≪ '2^L×EXP(−π×η/2−i×ρ)×ABS(GAMMA(L+1+i×η))/ GAMMA(2×L+2)×ρ^(L+1)×MABZ(L+1−i×η , 2×L+2 , i×2×ρ)' →NUM ≫ ≫

GLηρ: ≪ → L η ρ ≪ 'i×2^(L+1)×EXP(π×η/2−i×ρ+i×π×L)×GAMMA(L+1−i×η)/ ABS(GAMMA(L+1−i×η))×ρ^(L+1)×UABZ(L+1−i×η , 2×L+2 , i×2×ρ) + i×FLηρ(L,η,ρ)' →NUM ≫ ≫

For real arguments, the Coulomb wave functions are real, so the magnitude of the imaginary part is a good measure of accuracy. For example:

CONFLUENT HYPERGEOMETRIC FUNCTIONS MENU {FTNS CHYPR}

FUNCTION	COMMAND	INPUTS	OUTPUTS

FLηρ(0, 20, 5) = (1.64766608015E–16, –2.1241E–27)

FLηρ(0, 20, 10) = (68046931991.5, –1.73519055679E14)

FLηρ(10, 10, 5) = (3.28278060513E–10, 5.535E–21)

FLηρ(10, 1, 5) = (6.42377335421E–4, –7.0343E–14)

where the large imaginary part in the second case indicates that the computation diverged. The magnitude of the imaginary part should be small compared to that of the real part.

Similarly, for GLηρ we have the example:

GLηρ(0, 10, 1) = (3087903858.71, –68978.60974),

which is accurate to about 4 digits. Similarly,

GLηρ(0, 1, 5) = (–.898415236924, .000000342631)

GLηρ(0, 10, 5) = (–3527957308.07, –51091675548.3)

GLηρ(5, 2, 5) = (5.62984618287, 6.016727E–7)

GLηρ(10, 5, 5) = (91871.589099, 5.08717267472E–3),

where clearly GLηρ(0, 10, 5) did not numerically converge.

UP DIRECTORY	UPDIR	NONE	PARENT MENU

CONFLUENT HYPERGEOMETRIC FUNCTIONS MENU {FTNS CHYPR}

FUNCTION	COMMAND	INPUTS	OUTPUTS

AIRY FUNCTIONS AND THEIR DERIVATIVES

The Airy functions and their derivatives are easily computed in terms of Bessel functions. For $\zeta = 2/3\ z^{3/2}$, the relationships are

$$\mathrm{Ai}(z) = \tfrac{1}{3}\sqrt{z}\ [\ I_{-1/3}(\zeta) - I_{1/3}(\zeta)\ ] = \pi^{-1}\sqrt{z/3}\ K_{1/3}(\zeta),$$
$$\mathrm{Ai}(-z) = \tfrac{1}{3}\sqrt{z}\ [\ J_{1/3}(\zeta) + J_{-1/3}(\zeta)\ ],$$

$$\mathrm{Ai}'(z) = \tfrac{1}{3}\,z\ [\ I_{-2/3}(\zeta) - I_{2/3}(\zeta)\ ] = \pi^{-1}\,z/\sqrt{3}\ K_{2/3}(\zeta),$$
$$\mathrm{Ai}'(-z) = \tfrac{1}{3}\,z\ [\ J_{-2/3}(\zeta) - J_{2/3}(\zeta)\ ],$$

$$\mathrm{Bi}(z) = \sqrt{z/3}\ [\ I_{-1/3}(\zeta) + I_{1/3}(\zeta)\ ],$$
$$\mathrm{Bi}(-z) = \sqrt{z/3}\ [\ J_{-1/3}(\zeta) - J_{1/3}(\zeta)\ ],$$

$$\mathrm{Bi}'(z) = z/\sqrt{3}\ [\ I_{-2/3}(\zeta) + I_{2/3}(\zeta)\ ],$$
$$\mathrm{Bi}'(-z) = z/\sqrt{3}\ [\ J_{-2/3}(\zeta) + J_{2/3}(\zeta)\ ].$$

The Airy functions and their derivatives may thus be evaluated using **JVOZ**, **IVOZ**, and **KVOZ**.

CONFLUENT HYPERGEOMETRIC FUNCTIONS MENU {FTNS CHYPR}

FUNCTION	COMMAND	INPUTS	OUTPUTS

GENERALIZED HYPERGEOMETRIC FUNCTIONS

$$ {}_pF_q(\alpha_1,\alpha_2,\ldots,\alpha_p;\ \beta_1,\beta_2,\ldots,\beta_q;\ z) = \sum_{n=0}^{\infty} \frac{(\alpha_1)_n(\alpha_2)_n \ \ldots \ (\alpha_p)_n}{(\beta_1)_n(\beta_2)_n \ \ldots \ (\beta_q)_n} \frac{z^n}{n!} $$

${}_1F_1(a;\ b;\ z)$ is given in this menu and ${}_2F_1(a,\ b;\ c;\ z)$ is given in Chapter 14.

$$ {}_0F_1(\beta;\ z) = \sum_{n=0}^{\infty} \frac{z^n}{(\beta)_n\ n!} \qquad {}_1F_0(\alpha;\ z) = \sum_{n=0}^{\infty} \frac{(\alpha)_n\ z^n}{n!} $$

The below programs compute these hypergeometric functions.

F0F1: « → β z 'Σ(n=0,100,z^n/POCH(β,n)/n!)' »

F1F0: « → α z 'Σ(n=0,100,z^n×POCH(α,n)/n!)' »

For definitions see Chapter 13 of Abramowitz and Stegun, *Handbook of Mathematical Functions*, AMS 55, 1964.

Erdelyi, A., Magnus,W., Oberhettinger, F., and Tricomi, F., *Higher Transcendental Functions*, 3 Volumes, New York, McGraw–Hill, 1953.

Luke, Y., *Integrals of Bessel Functions*, New York, McGraw–Hill, 1962.

Slater, J., *Confluent Hypergeometric Functions*, New York, Cambridge Univ. Press, 1960.

13

PARABOLIC CYLINDER FUNCTIONS

INTRODUCTION

This chapter presents the 12 commands in the parabolic cylinder function menu. These functions are solutions to the wave equation in parabolic coordinates. The Bessel functions discussed in Chapters 7 and 8 are circular cylinder functions, whereas these are parabolic cylinder functions. These functions can be evaluated in terms of the Confluent Hypergeometric functions discussed in Chapter 12. Commands are given for both the classical Weber–Hermite functions as well as the newer AMS 55 normalized functions. Special input test commands are also provided.

WEBER–HERMITE FUNCTIONS

The commands **DV0Z**, **EV0Z**, and **EV1Z** evaluate Weber's functions $D_v(z)$, $E_v^{(0)}(z)$, and $E_v^{(1)}(z)$, respectively. The parabolic cylinder function $D_v(z)$ is related to the Hermite polynomials $H_n(z)$ and $He_n(z)$ discussed in Chapter 16 by the equations

$$H_n(z) = 2^{n/2}\, e^{z^2/2}\, D_n(z\sqrt{2}), \qquad D_n(z) = e^{-z^2/4}\, He_n(z).$$

These are the functions found in the older references (before 1964).

AMS 55 PARABOLIC CYLINDER FUNCTIONS

In Chapter 19 of AMS 55, three newer parabolic cylinder functions are discussed. Commands **UOAX** and **VOAX** evaluate the functions U(a, x) and V(a, x) defined by

$$U(a, x) = D_{-a-0.5}(x) \qquad V(a, x) = \frac{\Gamma(a + 0.5)}{\pi}\{ \sin \pi a \, D_{-a-0.5}(x) + D_{-a-0.5}(-x)\},$$

which are solutions to the differential equation

$$\frac{d^2y}{dx^2} - \left(\frac{x^2}{4} + a\right)y = 0.$$

AMS 55 also discusses the function W(a, ±x) which is evaluated by **WOAX** and satisfies the differential equation

$$\frac{d^2y}{dx^2} + \left(\frac{x^2}{4} - a\right)y = 0.$$

The complex solutions

$$E(a, x) = k^{-\frac{1}{2}} W(a, x) + i\, k^{\frac{1}{2}} W(a, -x) \qquad E^*(a, x) = k^{-\frac{1}{2}} W(a, x) - i\, k^{\frac{1}{2}} W(a, -x)$$

where $k = \sqrt{(1 + e^{2\pi a})} - e^{\pi a}$ and $1/k = \sqrt{(1 + e^{2\pi a})} + e^{\pi a}$ are also used.

The discussion in Section 19.16 of AMS 55 implies that the second term in W(a, ±x) defined below is missing a $e^{i\pi/4}$ factor. It is not.

The Math Library generally agrees with the five digit tables in AMS 55, though some small differences have been noticed around U(5, 5) and W(5, 5).

INPUT TEST COMMANDS

Three special input test commands are also given. They test numbers to see if they are integers, negative integers, or nonpositive real numbers.

PARABOLIC CYLINDER FUNCTIONS MENU {FTNS PCLDR}

FUNCTION	COMMAND	INPUTS	OUTPUTS
U(a, x)	**UOAX**(a,x)	$a \in \mathbf{R}$ $x \geq 0$	VALUE
V(a, x)	**VOAX**(a,x)	$a \in \mathbf{R}$ $x \geq 0$	VALUE
W(a, ±x)	**WOAX**(a,x)	$a, x \in \mathbf{R}$	VALUE

$$W(a, \pm x) = \frac{(\cosh \pi a)^{1/4}}{2\sqrt{\pi}} \left[\begin{array}{l} |\Gamma(\frac{1}{4}+\frac{1}{2}ia)|\, e^{-ix^2/4}\, M(\frac{1}{2}ia+\frac{1}{4},\ \frac{1}{2},\ \frac{1}{2}ix^2) \\ \quad \mp \sqrt{2}\ |\Gamma(\frac{3}{4}+\frac{1}{2}ia)|\, xe^{-ix^2/4}\, M(\frac{1}{2}ia+\frac{3}{4},\ \frac{3}{2},\ \frac{1}{2}ix^2) \end{array} \right]$$

$D_v(z)$	**DVOZ**(v,z)	$v, z \in \mathbf{C}$	VALUE

$$D_v(z) = 2^{v/2}\, e^{-z^2/4}\, U(-v/2,\ 0.5,\ z^2/2)$$

$E_v^{(0)}(z)$	**EV0Z**(v,z)	$v, z \in \mathbf{C}$	VALUE
$E_v^{(1)}(z)$	**EV1Z**(v,z)	$v, z \in \mathbf{C}$	VALUE

$$E_v^{(0)}(z) = \sqrt{2}\, e^{-z^2/4}\, M(-v/2,\ 1/2,\ z^2/2)$$

$$E_v^{(1)}(z) = 2z\, e^{-z^2/4}\, M((1 - v)/2,\ 3/2,\ z^2/2)$$

PARABOLIC CYLINDER FUNCTIONS MENU {FTNS PCLDR}

FUNCTION	COMMAND	INPUTS	OUTPUTS
$\Gamma(z)$	**GAMMA**(z)	$z \in \mathbb{C}$	VALUE
$\psi(z)$	**PSI**(z)	$z \in \mathbb{C}$	VALUE
TEST NOT POSITIVE	**TNP**(z)	$z \in \mathbb{C}$	1 IF TRUE 0 IF FALSE
TEST IF INTEGER	**TI**(z)	$z \in \mathbb{C}$	1 IF TRUE 0 IF FALSE
TEST NEGATIVE INTEGER	**TNI**(z)	$z \in \mathbb{C}$	1 IF TRUE 0 IF FALSE

TNP returns a 1 unless z is real and $z > 0$.

TI returns a 0 unless z is a real integer in $\mathbb{I}$.

TNI returns a 0 unless z is a negative real integer or zero.

UP DIRECTORY	**UPDIR**	NONE	PARENT MENU

For definitions see Chapters 13 & 19 of Abramowitz and Stegun, *Handbook of Mathematical Functions*, AMS 55, 1964.

Erdelyi, A., Magnus,W., Oberhettinger, F., and Tricomi, F., *Higher Transcendental Functions*, Volume 2, New York, McGraw–Hill, 1953.

Whittaker, E., and Watson, G., *Modern Analysis*, New York, Cambridge Univ. Press, 1969.

14

GAUSSIAN HYPERGEOMETRIC FUNCTION

INTRODUCTION

This chapter presents the six commands in the Gaussian hypergeometric function menu. Besides solving a very general differential equation and providing a method of evaluating numerous integrals, they offer a means of analytically continuing and evaluating many functions. Both confluent hypergeometric functions discussed in Chapter 12 can be obtained as limiting cases of the Gaussian hypergeometric function. AMS 55 lists many of the special cases. MATHLIB provides the incomplete beta function in this menu and the associated Legendre functions in the next chapter and menu.

GAUSSIAN HYPERGEOMETRIC FUNCTION

The basic Gaussian hypergeometric function is defined by the series

$$F(a,\ b,\ c,\ z) = {}_2F_1(a,\ b;\ c;\ z) = \sum_{n=0}^{\infty} \frac{(a)_n (b)_n}{(c)_n} \frac{z^n}{n!} = \frac{\Gamma(c)}{\Gamma(a)\Gamma(b)} \sum_{n=0}^{\infty} \frac{\Gamma(a+n)\Gamma(b+n)}{\Gamma(c+n)} \frac{z^n}{n!},$$

where again $(a)_n = \Gamma(a + n)/\Gamma(a)$ is Pochhammer's symbol, which can be evaluated by **POCH** in the MISC menu. The convergence of this series depends on a number of conditions. For example, when either a or b is equal to a negative integer –n, then F reduces to a polynomial of degree n. Through the use of linear and nonlinear transformations, the definition of F can be extended to the entire complex plane. The trick is, of course, to find the right transformation for your application that avoids dropping negative integers or zeros into the argument of any gamma functions. For example, as c goes to a negative integer or zero:

$$\lim_{c \to -m} \frac{1}{\Gamma(c)} F(a, b, c, z) = \frac{(a)_{m+1}(b)_{m+1}}{(m + 1)!} z^{m+1} F(a + m + 1, b + m + 1, m + 2, z).$$

FOGC evaluates this case, and **D2F1** evaluates the nth derivative of F using the **F2F1** command. **F2F1** evaluates a practical subset of the numerous cases. When F reduces to a polynomial, z can be almost anything. When F does not reduce to a polynomial, then either $|z| \leq 1$, or for a – b not an integer, F can be evaluated everywhere with **F2F1** except the branch cut on the real axis from 1 to ∞. Equations for the various logarithmic solution cases where a, b, and c are integer related are given in AMS 55 and Erdelyi. These cases can generally be approximated with **F2F1**.

RELATIONS

There are a number of simple relations which apply to Gaussian hypergeometric functions. The first is symmetry of the a and b arguments: F(b, a, c, z) = F(a, b, c, z). Secondly, for all b, $F(a, b, b, z) = (1 - z)^{-a}$. When either a or b is a negative integer, say –m, the series becomes a polynomial:

$$F(-m, b, c, z) = \sum_{n=0}^{m} \frac{(-m)_n (b)_n}{(c)_n} \frac{z^n}{n!}.$$

Several of the polynomials discussed in Chapter 16 are special cases of F(a, b, c, z). AMS 55 lists a number of other special cases. In addition, for integers j, k, m for $m + c \neq 0, -1, -2, \ldots$, AMS 55 lists numerous interrelations between the six functions $F(a \pm 1, b, c, z)$, $F(a, b \pm 1, c, z)$, and $F(a, b, c \pm 1, z)$ in terms of the hypergeometric functions F(a + j, b + k, c + m, z). Also, for $c \neq 0, -1, -2, \ldots$ and RE(c – a – b) > 0, we have the special case

$$F(a, b, c, 1) = \frac{\Gamma(c)\ \Gamma(c - a - b)}{\Gamma(c - a)\ \Gamma(c - b)}.$$

EVALUATION OF F(a, b, c, z)

The circle of convergence of the Gaussian hypergeometric function is the unit circle. **F2F1** gives a bad argument error if the arguments do not correspond to a convergent case. For example, when RE(c – a – b) ≤ –1, the infinite series always diverges when |z| = 1 and z ≠ 1. There is absolute convergence for RE(c – a – b) > 0 and conditional convergence in between. When neither a or b is a negative integer, the series is undefined for c = 0, –1, –2, ...

As with the confluent hypergeometric functions, the accuracy of **F2F1** is limited by the 12–digit precision of the HP 48. Large values of the a or b arguments can result in erroneous computation.

AMS 55 lists a number of linear transformations which are useful identities. For example:

$$\begin{aligned} F(a, b, c, z) &= (1 - z)^{c-a-b}\ F\ (c - a, c - b, c, z) \\ &= (1 - z)^{-a}\ F\left(a, c - b, c, \frac{z}{z - 1}\right) \\ &= (1 - z)^{-b}\ F\left(b, c - a, c, \frac{z}{z - 1}\right). \end{aligned}$$

To evaluate cases where |z| is close to 1 and |arg (1 – z)| < π use the formula

$$\begin{aligned} F(a, b, c, z) = {} & \frac{\Gamma(c)\Gamma(c - a - b)}{\Gamma(c - a)\Gamma(c - b)}\ F\ (a, b, a + b - c + 1, 1 - z) \\ & + \frac{\Gamma(c)\Gamma(a + b - c)}{\Gamma(a)\Gamma(b)}\ (1 - z)^{c-a-b}\ F\ (c - a, c - b, c - a - b + 1, 1 - z) \end{aligned}$$

when c ≠ a + b ± m for integer m. Formulas for the other cases are given in AMS 55. Quadratic and cubic transformations are also available.

GAUSSIAN HYPERGEOMETRIC FUNCTION MENU { FTNS GHYP }

FUNCTION	COMMAND	INPUTS	OUTPUTS
F(a, b, c, z)	**F2F1**(a,b,c,z)	a, b, c $\in \mathbb{C}$ $\lvert z \rvert \leq 1$	VALUE
NTH DERIVATIVE F(a, b, c, z)	**D2F1**(a,b,c,z,n)	a, b, c $\in \mathbb{C}$ $\lvert z \rvert \leq 1$ $n \in \mathbb{N}$	VALUE
F(a, b, c, z)/Γ(c)	**FOGC**(a,b,c,z)	a, b, c $\in \mathbb{C}$ $\lvert z \rvert \leq 1$	VALUE

In the above Gaussian hypergeometric functions the case where $|1 - z| < .01$ and $c = a + b \pm m$ where $m \in \mathbb{I}$ is not programmed. To evaluate it add a small number to your choice of a, b, or c.

For small a and b and $|z| < .95$, you generally get about 10 good digits. As $|z|$ approaches 1, the approximations break down and you may only get 3 good digits. When a or b get large compared with c, the approximations go bad. For these cases, try the linear transformations or recurrence formulas. The below formulas can be used to check accuracy (AMS 55 lists many more):

$$F(1, 1, 2, z) = -z^{-1} \ln(1 - z) \qquad F(.5, 1, 1.5, -z^2) = z^{-1} \arctan z$$

$$F(a, .5 + a, .5, z^2) = [(1 + z)^{-2a} + (1 - z)^{-2a}]/2$$

See Chapter 15 for more discussion related to these functions.

ANALYTIC CONTINUATION

F2F1, **D2F1**, and **FOGC** do evaluate F for $|z| > 1$, provided arg $z \neq 0$ and $a - b$ is not an integer, using linear transformations. If you get a bad argument error, you will have to carefully think out which case (equations) you are evaluating and modify your inputs accordingly.

GAUSSIAN HYPERGEOMETRIC FUNCTION MENU { FTNS GHYP }

FUNCTION	COMMAND	INPUTS	OUTPUTS

EXAMPLES

F2F1(1.2, –3, .7, 23) = –28587.4631188

F2F1(1.2, 0.8, 3.5, 0.9) = 1.46874542178

F2F1(7.8, –5.3, 4.11, (3,–1)) = (–521.277031973 , 1576.50699403)

The case F2F1(7.8, –5.2, 4.11, (3,–1)) will result in a bad argument error since b – a = –12. To evaluate it you may either program the equations in AMS 55 or do the approximation:

F2F1(7.8, –5.2000000001, 4.11, (3,–1)) = (–71.9253996891, 1468.8278948).

F has a branch cut from 1 to $+\infty$, so F is not continuous there.

F2F1(7.8, –5.2000000001, 4.11, (2 , ± 1E–8))
= (–35.9525409978 , ± 26.1210477685)

$I_z(a, b)$	**INCBH**(a,b,z)	a, b $\in \mathbb{C}$ $\quad$ $\lvert z\rvert \leq 1$	VALUE

INCBH provides analytic continuation for the incomplete beta function. See Chapter 5.

INCBH is not defined at a, b, or a + b = 0, –1, –2, ...

$\Gamma(z)$	**GAMMA**(z)	z $\in \mathbb{C}$	VALUE
UP DIRECTORY	**UPDIR**	NONE	PARENT MENU

GAUSSIAN HYPERGEOMETRIC FUNCTION MENU { FTNS GHYP }

FUNCTION	COMMAND	INPUTS	OUTPUTS

NOTES ON THE HYPERGEOMETRIC FUNCTION

When either a or b is a negative integer, say a = –m, the series becomes a polynomial:

$$F(-m, b, c, z) = \sum_{n=0}^{m} \frac{(-m)_n (b)_n}{(c)_n} \frac{z^n}{n!}.$$

Erdelyi and AMS 55 *define this* equation to hold even when c is a negative integer, such that c = –m – j for j = 0, 1, 2, . . . even though for j = 0, F(–m, b, –m, z) = $(1 - z)^{-b}$. We follow their definition for j > 0, but use $(1 - z)^{-b}$ for j = 0. However, the user should understand that even with these definitions, **F2F1** is not numerically stable in the neighborhood of c equal to a negative integer.

F2F1(–5, 2.5, –5, .5) = 5.65685424949

F2F1(–5, 2.5, –5.0000000001, .5) = 5.09460449191

F2F1(–5.0000000001, 2.5, –4.999999999, .5) = 5.03837951961

F2F1(–5.0000000001, 2.5, (–5,1E–499), .5)
= (5.09460449248,5.62249757281E488)

For definitions see Chapters 15 of Abramowitz and Stegun, *Handbook of Mathematical Functions*, AMS 55, 1964.

Erdelyi, A., Magnus,W., Oberhettinger, F., and Tricomi, F., *Higher Transcendental Functions*, Volume 1, New York, McGraw–Hill, 1953.

15

LEGENDRE AND STRUVE FUNCTIONS

INTRODUCTION

This chapter presents the six commands in the Legendre function menu LGDR and the six commands in the Struve function menu STRUV. The Legendre functions are solutions to wave and diffusion differential equations in spherical coordinates and are thus commonly called spherical harmonics. Struve functions are useful for evaluating certain integrals, and since there are only two functions, we have included them in this chapter.

In the below discussion of Legendre functions, we assume that the reader is very familiar with the definitions, properties, and transformations of Gaussian hypergeometric functions.

ASSOCIATED LEGENDRE FUNCTIONS

The associated Legendre functions of the first and second kind are denoted by $P_v^{\mu}(z)$ and $Q_v^{\mu}(z)$, where v is the degree and μ is the order. The associated Legendre functions may be defined in terms of the Legendre functions $P_v(z)$ and $Q_v(z)$ for $\mu = m = 1, 2, \ldots$ by the relations

$$P_v^m(z) = (z^2 - 1)^{m/2} \frac{d^m P_v(z)}{dz^m}, \qquad Q_v^m(z) = (z^2 - 1)^{m/2} \frac{d^m Q_v(z)}{dz^m}.$$

Clearly, $P_v^0(z) = P_v(z)$ and $Q_v^0(z) = Q_v(z)$. When $\mu = 0$ and v is an integer n, then $P_n(z)$ is called a Legendre polynomial. $P_n(z)$ is one of the orthogonal polynomials discussed in Chapter 16. The Legendre function $Q_n(z)$ may be expressed in terms of $P_n(z)$. $P_n^m(z)$ is trivial to compute, and all the rest are more difficult.

In general, degree v and order μ are complex. The associated Legendre functions may be defined by

$$P_v^\mu(z) = \frac{1}{\Gamma(1-\mu)}\left[\frac{z+1}{z-1}\right]^{\mu/2} F(-v,\ v+1,\ 1-\mu,\ \frac{1-z}{2}) \qquad |1-z| < 2,$$

$$Q_v^\mu(z) = e^{i\mu\pi}2^{-v-1}\sqrt{\pi}\frac{\Gamma(v+\mu+1)}{\Gamma(v+1.5)}z^{-v-\mu-1}(z^2-1)^{\mu/2}\ F\left(1+\frac{v}{2}+\frac{\mu}{2},\ \frac{1}{2}+\frac{v}{2}+\frac{\mu}{2},\ v+\frac{3}{2},\ \frac{1}{z^2}\right) \qquad |z| > 1,$$

where the hypergeometric function transformation formulas discussed in Chapter 14 extend the definitions to $z \in \mathbb{C}$.

VALUES ON THE CUT

In general, neither of the associated Legendre functions is continuous at the real axis line between −1 and 1. Consequently, a special definition is used for this cut through the complex plane. It is

$$p_v^\mu(x) = e^{\pm i\mu\pi/2}\ P_v^\mu(x \pm i0) \qquad -1 < x < 1 \qquad \text{where} \quad f(x \pm i0) = \lim_{\varepsilon\to 0} f(x \pm i\varepsilon),$$

$$q_v^\mu(x) = \frac{1}{2}\ e^{-i\mu\pi}\left[e^{-i\mu\pi/2}\ Q_v^\mu(x + i0) + e^{i\mu\pi/2}\ Q_v^\mu(x - i0)\right] \qquad -1 < x < 1.$$

The commands **PμVX** and **QμVX** evaluate $p_v^\mu(x)$ and $q_v^\mu(x)$, respectively.

When v and μ equal the non-negative integers n and m, $p_n^m(x) = 0$ for $m > n$, $p_0^0(x) = 1$, and for $m = 1, 2, \ldots$

$$p_m^m(x) = (-1)^m\ (2m - 1)!!\ [1 - x^2]^{m/2}, \qquad p_{m+1}^m(x) = x\ (2m + 1)\ p_m^m(x),$$

so the computation is trivial using the recurrence formula implemented by **PQUP**.

RELATIONS BETWEEN LEGENDRE FUNCTIONS

In practice, a number of relations are useful in evaluating the associated Legendre functions.

$$P_\nu^\mu(-z) = e^{\pm i\nu\pi} P_\nu^\mu(z) - \frac{2}{\pi} e^{-i\mu\pi} \sin[\pi(\nu + \mu)] Q_\nu^\mu(z), \qquad Q_\nu^\mu(-z) = -e^{\pm i\nu\pi} Q_\nu^\mu(z),$$

where the upper sign is used when IM(z) > 0 and the lower sign when IM(z) < 0. On the cut we have

$$p_\nu^\mu(-x) = p_\nu^\mu(x) \cos[\pi(\nu + \mu)] - \frac{2}{\pi} q_\nu^\mu(x) \sin[\pi(\nu + \mu)],$$
$$q_\nu^\mu(-x) = -q_\nu^\mu(x) \cos[\pi(\nu + \mu)] - \frac{\pi}{2} p_\nu^\mu(x) \sin[\pi(\nu + \mu)],$$

where $0 < x < 1$. For negative degree we have

$$P_{-\nu-1}^\mu(z) = P_\nu^\mu(z), \qquad Q_{-\nu-1}^\mu(z) = \frac{-\pi\, e^{i\mu\pi} \cos \nu\pi\, P_\nu^\mu(z) + Q_\nu^\mu(z) \sin[\pi(\nu + \mu)]}{\sin[\pi(\nu - \mu)]},$$

and for negative order:

$$P_\nu^{-\mu}(z) = \frac{\Gamma(\nu - \mu + 1)}{\Gamma(\nu + \mu + 1)}\left[P_\nu^\mu(z) - \frac{2}{\pi} e^{-i\mu\pi} \sin(\mu\pi) Q_\nu^\mu(z)\right], \qquad Q_\nu^{-\mu}(z) = e^{-2i\mu\pi} \frac{\Gamma(\nu - \mu + 1)}{\Gamma(\nu + \mu + 1)} Q_\nu^\mu(z).$$

The recurrence formula used by **PQUP** for both $P_\nu^\mu(z)$ and $Q_\nu^\mu(z)$ is

$$(\nu - \mu) P_\nu^\mu(z) = (2\nu - 1) z P_{\nu-1}^\mu(z) - (\nu + \mu - 1) P_{\nu-2}^\mu(z).$$

Two other useful recurrence formulas satisfied by both $P_\nu^\mu(z)$ and $Q_\nu^\mu(z)$ are

$$P_\nu^{\mu+1}(z) = (z^2 - 1)^{-1/2} \{(\nu - \mu) z P_\nu^\mu(z) - (\nu + \mu) P_{\nu-1}^\mu(z)\},$$

$$P_{\nu+1}^\mu(z) = P_{\nu-1}^\mu(z) + (2\nu + 1) (z^2 - 1)^{1/2} P_\nu^{\mu-1}(z).$$

POLYNOMIALS $P_n^m(z)$

Some authors fail to carefully make the distinction between formulas in the general case and ones which only apply to the cut. Consequently, we derive a basic formula which is missing in AMS 55.

$$P_\nu^\mu(z) = \frac{1}{\Gamma(1-\mu)} \left[\frac{z+1}{z-1}\right]^{\mu/2} F(-\nu, \nu+1, 1-\mu, \frac{1-z}{2}) \qquad |1-z| < 2$$

Observe that the c argument $1 - \mu$ of the hypergeometric function F(a, b, c, z) is the same as that of the gamma function. This special case is handled by the **FOGC** command, which properly evaluates the case where μ goes to one of the integers 0, 1, 2, ... We are about to prove the following fact that you can try on the HP 48. For non-negative integers n and r, we have

$$\text{FOGC}(-n, n+1, 1-n-r, z) = 0 \qquad r > 0 \qquad z \in \mathbb{C}.$$

Hence, $P_n^m(z) = 0$ for non-negative integers $m = n + r$ and n whenever $m > n$. If we apply the second hypergeometric function linear transformation listed in Chapter 14 to our above definition of $P_\nu^\mu(z)$, we get the equation

$$P_\nu^\mu(z) = \left[\frac{z+1}{z-1}\right]^{\mu/2} \frac{[z+1]^\nu \, 2^{-\nu}}{\Gamma(1-\mu)} F\left(-\nu, -\mu-\nu, 1-\mu, \frac{z-1}{z+1}\right).$$

Taking the limit as μ goes to the non-negative integer m, using the equation in Chapter 14, we have

$$P_\nu^m(z) = \left[\frac{z+1}{z-1}\right]^{m/2} [z+1]^\nu \, 2^{-\nu} \frac{(-\nu)_m \, (-m-\nu)_m}{m!} \left[\frac{z-1}{z+1}\right]^\nu F\left(m-\nu, -\nu, m+1, \frac{z-1}{z+1}\right).$$

Observe that if $\nu = m$, then the hypergeometric function equals 1, because for any positive integers r and s we have

$$F(0, -r, s, z) = \sum_{k=0}^{0} \frac{(0)_k \, (-r)_k}{(s)_k} \frac{z^k}{k!} = 1,$$

$$F(-r, 0, s, z) = \sum_{k=0}^{r} \frac{(-r)_k \, (0)_k}{(s)_k} \frac{z^k}{k!} = 1.$$

The following formula can be derived from the reflection formula for the gamma function.

$$(-z)_s = (-1)^s \frac{\Gamma(z+1)}{\Gamma(z+1-s)} = (-1)^s (z+1-s)_s \qquad z \in \mathbb{C} \qquad s = 0, 1, 2, \ldots$$

Now if this seems hard to believe, evaluate POCH(–5.5, 3) = –POCH(3.5, 3). The **POCH** command in the MISC menu evaluates $(z)_n$ = POCH(z, n) in a numerically stable way for $z \in \mathbb{C}$ and $n \in \mathbb{L}$ Using the $(-z)_s$ formula and setting v = m – r for non-negative integer r, we have

$$\frac{(-m+r)_m\,(-2m+r)_m}{2^{m-r}\,m!} = \frac{(1-r)_m\,(m+1-r)_m}{2^{m-r}\,m!} = \frac{(m-r)!\,(2m-r)!}{(-r)!\,(m-r)!\,m!\,2^{m-r}} = \begin{cases} (2m-1)!! & r=0 \\ 0 & r>0 \end{cases}$$

since r = m – v > 0 implies 2m – r ≥ 0, because v = n equals a non-negative integer. Hence, $P_n^m(z) = 0$ for m > n, $P_0^0(z) = 1$, and for m = n = 1, 2, . . .,

$$P_m^m(z) = (2m-1)!!\,[z^2-1]^{m/2},$$

where (2m – 1)!! is the command ¡¡(2m – 1). From the recurrence formula we also have

$$P_{m+1}^m(z) = (2m+1)\,z\,P_m^m(z),$$

which can be compared with the related formulas for z = x on the cut. These formulas combined with **PQUP** are fast for computing the associated Legendre functions of integer order and degree. When m = 0, you can compute symbolically or numerically the polynomials with the command **POFX** discussed in Chapter 16. A few examples are

$$P_0^0(z) = 1 \qquad P_1^0(z) = z \qquad P_2^0(z) = \tfrac{1}{2}(3z^2-1) \qquad P_3^0(z) = \tfrac{1}{2}z[5z^2-3]$$

$$P_1^1(z) = \sqrt{z^2-1} \qquad P_2^1(z) = 3z\sqrt{z^2-1} \qquad P_3^1(z) = \tfrac{3}{2}[5z^2-1]\sqrt{z^2-1}$$

$$P_2^2(z) = 3[z^2 - 1] \qquad P_3^2(z) = 15z[z^2 - 1] \qquad P_3^3(z) = 15[z^2 - 1]^{3/2}.$$

Alternatively, using the differentiation formula for $P_v^m(z)$ we have

$$P_m^m(z) = [z^2 - 1]^{m/2} \frac{d^m P_m(z)}{dz^m} = [z^2 - 1]^{m/2} (2m - 1)!!,$$

so $P_n^m(z) = 0$ for $m > n$, since the derivative of the constant $(2m - 1)!!$ is zero.

Observe that as discussed in Chapter 3, both the √ and **SQ** commands on the HP 48 have branch cuts on the negative real axis. AS A CONSEQUENCE, ALL THE EXPRESSIONS IN THIS CHAPTER LIKE $(z^2-1)^{\alpha/2}$ FOR $\alpha \in \mathbb{C}$ MUST BE EVALUATED ON THE HP 48 AS $(z-1)^{\alpha/2}(z+1)^{\alpha/2}$ IN ORDER TO OBTAIN THE CORRECT RESULT.

FUNCTIONS $Q_n^m(z)$

In general, the functions $Q_n(z)$ can be written as

$$Q_n(z) = \alpha P_n(z) - W_{n-1}(z), \qquad \alpha = \frac{1}{2}\ln\left(\frac{z+1}{z-1}\right), \qquad W_{n-1}(z) = \sum_{k=0}^{[(n-1)/2]} \frac{(2n-4k-1}{(n-k)(2k+1)} P_{n-2k-1}(z),$$

where $W_{n-1}(z)$ is a polynomial of degree $n - 1$ and $W_{-1}(z) = 0$. If we define $R_n(z) = W_{n-1}(z)$, then $R_n(z)$ satisfies the same recurrence relations as $P_n(z)$ and $Q_n(z)$, which is the easiest way to compute it. The below programs return the algebraic polynomials $P_n(z)$ and $R_n(z)$ from 0 to input v.

```
PSSV: « → v « { z 1 } 1 z → L α β « 2 v FOR k β '(2×k-1)/k×z×β-(k-1)
      /k×α' EVAL EXCO DUP 'L' STO+ 'β' STO 'α' STO NEXT L » » »

WSSV: « → v « { -1 0 } 0 -1 → L α β « 2 v FOR k β '(2×k-1)/k×z×β-(k-1)
      /k×α' EVAL EXCO DUP 'L' STO+ 'β' STO 'α' STO NEXT L » » »
```

Observe that $Q_n^m(z)$ is a transcendental function, not a polynomial, and $Q_n^m(z) \neq 0$ for $m > n$. Some examples are

$$Q_0^0(z) = \alpha = \frac{1}{2}\ln\left(\frac{z+1}{z-1}\right) = \alpha\, P_0^0(z) \qquad Q_1^0(z) = \alpha\, z - 1 = \alpha\, P_1^0(z) - 1$$

$$Q_2^0(z) = \alpha\frac{(3z^2 - 1)}{2} - \frac{3}{2}z = \alpha\, P_2^0(z) - \frac{3}{2}z$$

$$Q_3^0(z) = \alpha\frac{(5z^3 - 3z)}{2} - \frac{5}{2}z^2 + \frac{2}{3} = \alpha\, P_3^0(z) - \frac{5}{2}z^2 + \frac{2}{3}$$

$$Q_0^1(z) = \frac{-1}{\sqrt{z^2-1}} \qquad Q_0^2(z) = \frac{2z}{z^2-1} \qquad Q_0^3(z) = \frac{-6z^2-2}{(z^2-1)^{3/2}}$$

$$Q_1^1(z) = \frac{-z}{\sqrt{z^2-1}} + \alpha\sqrt{z^2-1} \qquad Q_1^2(z) = \frac{2}{z^2-1} \qquad Q_1^3(z) = \frac{-8z}{(z^2-1)^{3/2}}$$

$$Q_2^1(z) = \frac{-3z^2+2}{\sqrt{z^2-1}} + \alpha\, 3z\sqrt{z^2-1} \qquad Q_2^2(z) = \frac{-3z^3+5z}{z^2-1} + \alpha\, 3\,(z^2-1)$$

$$Q_2^3(z) = \frac{-8}{(z^2-1)^{3/2}} \qquad Q_3^1(z) = \frac{-15z^3 + 13z}{2\sqrt{z^2 - 1}} + \alpha\sqrt{z^2-1}\left(\frac{15}{2}z^2 - \frac{3}{2}\right)$$

$$Q_3^2(z) = \frac{-15z^4 + 25z^2 - 8}{z^2 - 1} + \alpha\, 15z\,(z^2 - 1)$$

$$Q_3^3(z) = \frac{-15z^5 + 40z^3 - 33z}{(z^2 - 1)^{3/2}} + \alpha\, 15\,(z^2 - 1)^{3/2}.$$

Observe the explicit poles at $z = \pm 1$ in these equations. These formulas allow you to use the recurrence relations given above.

$P_n^m(x)$ AND $Q_n^m(x)$ ON THE CUT

AMS 55 and Erdelyi state general rules for obtaining explicit formulas for $p_\nu^\mu(x)$ and $q_\nu^\mu(x)$ on the cut $-1 < x < 1$ where $P_\nu^\mu(z)$ and $Q_\nu^\mu(z)$ are, in general, discontinuous:

- replace $z - 1$ by $(1 - x)e^{\pm i\pi}$
- replace $z + 1$ by $(x + 1)$
- replace $(z^2 - 1)$ by $(1 - x^2)e^{\pm i\pi}$

for $z = x \pm i0$ where $\pm i0$ is the limit as ε goes to zero of $i\varepsilon$. $P_\nu^{-\nu}(z)$ and $p_\nu^{-\nu}(x)$ defined in the next section is a counterexample to the rules, so use the definition, not the rules. Some explicit formulas are

$$p_0^0(x) = 1 \qquad p_1^0(x) = x \qquad p_2^0(x) = \tfrac{1}{2}(3x^2 - 1) \qquad p_3^0(x) = \tfrac{1}{2}x[5x^2 - 3]$$

$$p_1^1(x) = -\sqrt{1-x^2} \qquad p_2^1(x) = -3x\sqrt{1-x^2} \qquad p_3^1(x) = -\tfrac{3}{2}[5x^2-1]\sqrt{1-z^2}$$

$$p_2^2(x) = 3[1-x^2] \qquad p_3^2(x) = 15x[1-x^2] \qquad p_3^3(x) = -15[1-x^2]^{3/2}$$

$$q_0^0(x) = \alpha = \tfrac{1}{2}\ln\left(\frac{1+x}{1-x}\right) = \alpha\, p_0^0(x) \qquad q_1^0(x) = \alpha\, x - 1 = \alpha\, p_1^0(x) - 1$$

$$q_2^0(x) = \alpha\frac{(3x^2 - 1)}{2} - \frac{3}{2}x = \alpha\, p_2^0(x) - \frac{3}{2}x$$

$$q_3^0(x) = \alpha\frac{(5x^3-3z)}{2} - \frac{5}{2}x^2 + \frac{2}{3} = \alpha\, p_3^0(x) - \frac{5}{2}x^2 + \frac{2}{3}$$

$$q_0^1(x) = \frac{-1}{\sqrt{1-x^2}} \qquad q_0^2(x) = \frac{2x}{1-x^2} \qquad q_0^3(x) = \frac{-6x^2-2}{(1-x^2)^{3/2}}$$

$$q_1^1(x) = \frac{-x}{\sqrt{1-x^2}} - \alpha\sqrt{1-x^2} \qquad q_1^2(x) = \frac{2}{1-x^2} \qquad q_1^3(x) = \frac{-8x}{(1-x^2)^{3/2}}$$

$$q_2^1(x) = \frac{-3x^2+2}{\sqrt{1-x^2}} - \alpha\, 3x\sqrt{1-x^2} \qquad q_2^2(x) = \frac{-3x^3+5x}{1-x^2} + \alpha\, 3\,(1-x^2)$$

$$q_2^3(x) = \frac{-8}{(1-x^2)^{3/2}} \qquad q_3^1(x) = \frac{-15x^3+13x}{2\sqrt{1-x^2}} - \alpha\sqrt{1-x^2}\left(\frac{15}{2}x^2 - \frac{3}{2}\right)$$

$$q_3^2(x) = \frac{-15x^4+25x^2-8}{1-x^2} + \alpha\, 15x\,(1-x^2)$$

$$q_3^3(x) = \frac{-15x^5+40x^3-33x}{(1-x^2)^{3/2}} - \alpha\, 15\,(1-x^2)^{3/2}.$$

OTHER SPECIAL CASES

We noted above that $P_n^m(x)$ is a particularly easy case. AMS 55 lists several more. Specifically:

$$P_\nu^{1/2}(z) = (z^2 - 1)^{-1/4} \{[z + (z^2 - 1)^{1/2}]^{\nu+1/2} + [z + (z^2 - 1)^{1/2}]^{-\nu-1/2}\}/\sqrt{2\pi}$$

$$P_\nu^{-1/2}(z) = \sqrt{\frac{2}{\pi}} \frac{(z^2 - 1)^{-1/4}}{2\nu + 1} \{[z + (z^2 - 1)^{1/2}]^{\nu+1/2} - [z + (z^2 - 1)^{1/2}]^{-\nu-1/2}\}$$

$$Q_\nu^{1/2}(z) = i\sqrt{\pi/2}\,(z^2 - 1)^{-1/4} [z + (z^2 - 1)^{1/2}]^{-\nu-1/2}$$

$$Q_\nu^{-1/2}(z) = -i\sqrt{2\pi}\,\frac{(z^2 - 1)^{-1/4}}{2\nu + 1} [z + (z^2 - 1)^{1/2}]^{-\nu-1/2}$$

$$P_\nu^{-\nu}(z) = \frac{2^{-\nu}(z^2 - 1)^{\nu/2}}{\Gamma(\nu + 1)}.$$

On the cut, these formulas become

$$p_\nu^{1/2}(x) = (1-x^2)^{-1/4} \{[x+i(1-x^2)^{1/2}]^{\nu+1/2} + [x-i(1-x^2)^{1/2}]^{\nu+1/2}\}/\sqrt{2\pi}$$

$$p_\nu^{-1/2}(x) = -i\sqrt{\frac{2}{\pi}}\,\frac{(1-x^2)^{-1/4}}{2\nu+1} \{[x+i(1-x^2)^{1/2}]^{\nu+1/2} - [x-i(1-x^2)^{1/2}]^{\nu+1/2}\}$$

$$q_\nu^{1/2}(x) = -i\,\tfrac{1}{2}\sqrt{\pi/2}\,(1-x^2)^{-1/4} \{ [x+i(1-x^2)^{1/2}]^{-\nu-1/2} - [x-i(1-x^2)^{1/2}]^{-\nu-1/2}\}$$

$$q_\nu^{-1/2}(x) = \sqrt{\pi/2}\,\frac{(1-x^2)^{-1/4}}{2\nu+1} \{ [x+i(1-x^2)^{1/2}]^{-\nu-1/2} + [x-i(1-x^2)^{1/2}]^{-\nu-1/2}\}$$

$$p_v^{-v}(x) = \frac{2^{-v}(1 - x^2)^{v/2}}{\Gamma(v + 1)}.$$

Observe that $p_v^{-v}(x)$ violates the AMS 55 rules for formulas on the cut, but does satisfy the definition of on the cut. The first four of the above five equations can be nicely expressed using the trigonometric substitution $x = \cos\theta$ as:

$$p_v^{1/2}(\cos\theta) = \sqrt{2/\pi}\,\frac{\cos[(v + 1/2)\theta]}{\sqrt{\sin\theta}} \qquad \theta \in (0,\pi),$$

$$p_v^{-1/2}(\cos\theta) = \frac{2\sqrt{2/\pi}}{2v + 1}\,\frac{\sin[(v + 1/2)\theta]}{\sqrt{\sin\theta}} \qquad \theta \in (0,\pi),$$

$$q_v^{1/2}(\cos\theta) = -\sqrt{\pi/2}\,\frac{\sin[(v + 1/2)\theta]}{\sqrt{\sin\theta}} \qquad \theta \in (0,\pi),$$

$$q_v^{-1/2}(\cos\theta) = \frac{\sqrt{2\pi}}{2v + 1}\,\frac{\cos[(v + 1/2)\theta]}{\sqrt{\sin\theta}} \qquad \theta \in (0,\pi).$$

These equations explicitly show the poles at $x = \pm 1$ when $\mu = \pm 1/2$.

COMPUTATION OF DERIVATIVES

Derivatives of both $P_v^{\mu}(z)$ and $Q_v^{\mu}(z)$ are easily computed from the recurrence relations:

$$(z^2 - 1)\frac{dP_v^{\mu}(z)}{dz} = (v + \mu)(v - \mu + 1)\sqrt{z^2 - 1}\,P_v^{\mu-1}(z) - \mu z\,P_v^{\mu}(z),$$

$$(z^2 - 1)\,\frac{dP_\nu^\mu(z)}{dz} = \nu z\, P_\nu^\mu(z) - (\nu + \mu)\, P_{\nu-1}^\mu(z).$$

RECURRENCE FORMULAS FOR ON THE CUT

With the exception of the recurrence relation implemented by **PQUP** which also works for $p_\nu^\mu(x)$ and $q_\nu^\mu(x)$, the formulas are different on the cut. For x on the cut, the formulas for $p_\nu^\mu(x)$ and $q_\nu^\mu(x)$ are

$$p_\nu^{\mu+2}(x) + \frac{2(\mu+1)x}{\sqrt{1-x^2}}\, p_\nu^{\mu+1}(x) + (\nu - \mu)(\nu + \mu + 1)\, p_\nu^\mu(x) = 0,$$

$$p_{\nu-1}^\mu(x) - p_{\nu+1}^\mu(x) = (2\nu + 1)\sqrt{1 - x^2}\, p_\nu^{\mu-1}(x),$$

$$p_{\nu-1}^\mu(x) - x\, p_\nu^\mu(x) = (\nu - \mu + 1)\sqrt{1 - x^2}\, p_\nu^{\mu-1}(x),$$

$$x\, p_\nu^\mu(x) - p_{\nu+1}^\mu(x) = (\nu + \mu)\sqrt{1 - x^2}\, p_\nu^{\mu-1}(x),$$

$$(\nu - \mu)\, x\, p_\nu^\mu(x) - (\nu + \mu)\, p_{\nu-1}^\mu(x) = \sqrt{1 - x^2}\, p_\nu^{\mu+1}(x),$$

$$(\nu - \mu + 1)\, p_{\nu+1}^\mu(x) - (\nu + \mu + 1)\, x\, p_\nu^\mu(x) = \sqrt{1 - x^2}\, p_\nu^{\mu+1}(x),$$

$$\begin{aligned}(1 - x^2)\,\frac{dp_\nu^\mu(x)}{dx} &= (\nu + 1)\, x\, p_\nu^\mu(x) - (\nu - \mu + 1)\, p_{\nu+1}^\mu(x) \\ &= -\nu x\, p_\nu^\mu(x) + (\nu + \mu)\, p_{\nu-1}^\mu(x).\end{aligned}$$

SURFACE HARMONICS

Solutions to Laplace's and Poisson's equations in spherical coordinates are in the form of even and odd surface harmonics Y.

$$Ye_n^m(\theta, \varphi) = \cos m\varphi \; p_n^m(\cos \theta) \qquad Yo_n^m(\theta, \varphi) = \sin m\varphi \; p_n^m(\cos \theta),$$

where $0 \le \theta \le \pi$ and $0 \le \varphi \le 2\pi$. Using complex notation, this can be written as

$$Y_n^m(\theta, \varphi) = e^{im\varphi} \; p_n^m(\cos \theta) = Ye_n^m(\theta, \varphi) + i \; Yo_n^m(\theta, \varphi).$$

For m = 0, these spherical harmonics are called zonal harmonics, since they only depend on θ, and the nodal lines divide the sphere into zones. The ones for m = n are called sectoral, since the nodal lines divide the sphere into sectors. For $0 < m < n$, they are called tesseral harmonics. These functions are mutually orthogonal eigenfunctions being one–valued and continuous for the two–dimensional surface of the sphere defined by $x^2 + y^2 + z^2 = 1$, where $x = \sin \theta \cos \varphi$, $y = \sin \theta \sin \varphi$, and $z = \cos \theta$. These are most easily evaluated by the formulas for $p_m^m(x)$ and $p_{m+1}^m(x)$ derived above and by use of the recurrence relation **PQUP**.

PROLATE AND OBLATE SPHERICAL AND BISPHERICAL COORDINATES

Solutions to Laplace's and Poisson's equations in prolate and oblate spherical coordinates as well as bispherical coordinates can be expressed in terms of the Legendre functions $p_n^m(x)$ and $q_n^m(x)$. See Morse and Feshbach for details.

TOROIDAL COORDINATES

Solutions to Laplace's and Poisson's equations in toroidal coordinates can be expressed in terms of the Legendre functions $P_{n-1/2}^m(\cosh \eta)$ and $Q_{n-1/2}^m(\cosh \eta)$ for $0 < \eta < \infty$. These are commonly called toroidal or ring functions and can be evaluated with the commands **PμVZ** and **QμVZ**.

CONICAL FUNCTIONS

The conical functions are the solution of the differential equation:

$$(1 - z^2) \frac{d^2w}{dz^2} - 2z \frac{dw}{dz} - [\lambda^2 + 1/4 + (1 - z^2)^{-1} \mu^2] w = 0,$$

where λ is a real parameter. The solutions can be expressed in terms of the Legendre functions

$$P^{\mu}_{-1/2+i\lambda}(z), \qquad Q^{\mu}_{-1/2+i\lambda}(z),$$

and for z equals real x = cos θ on the cut,

$$p^{\mu}_{-1/2+i\lambda}(\cos\theta), \qquad q^{\mu}_{-1/2+i\lambda}(\cos\theta), \qquad \theta \in (0,\pi).$$

GEGENBAUER FUNCTIONS

The Gegenbauer functions $C_v^{(\alpha)}(z)$ and $D_v^{(\alpha)}(z)$, which are solutions to the Gegenbauer differential equation, can also be expressed in terms of the Legendre or Gaussian hypergeometric functions. $C_n^{(\alpha)}(x)$ is one of the orthogonal polynomials discussed in Chapter 16.

LEGENDRE SOFTWARE VERIFICATION

Since the user may easily be confused by the equations and rules which do not work in the literature, numerous explicit examples have been provided. Remember, however, ALL THE EXPRESSIONS IN THIS CHAPTER LIKE $(z^2-1)^{\alpha/2}$ FOR $\alpha \in \mathbb{C}$ MUST BE EVALUATED ON THE HP 48 AS $(z-1)^{\alpha/2}(z+1)^{\alpha/2}$ IN ORDER TO OBTAIN THE CORRECT RESULT. Also, LN((z+1)/(z−1)) must be evaluated as LN(z+1)−LN(z−1) on the HP 48 to avoid incorrect results for some values of z. The general Legendre commands given below do agree with all these special cases throughout the entire complex plane.

STRUVE FUNCTION

The Struve function $\mathbf{H}_v(z)$ is provided by **HVOZ** and defined by the power series expansion

$$H_v(z) = (z/2)^{v+1} \sum_{n=0}^{\infty} \frac{(-1)^n (z/2)^{2n}}{\Gamma(n+3/2)\ \Gamma(n+v+3/2)},$$

where for integer $k \geq 0$, the case $v = -(k+1/2)$ is evaluated as $\mathbf{H}_{-(k+1/2)}(z) = (-1)^k J_{k+1/2}(z)$. For large arguments of $|z|$, the asymptotic expansion

$$H_v(z) \approx Y_v(z) + \frac{1}{\pi} \sum_{k=0}^{m-1} \frac{\Gamma(k+1/2)}{\Gamma(v+1/2-k)\ (z/2)^{2k-v+1}} \qquad |\arg z| < \pi$$

may be used for $|\arg z| < \pi$.

MODIFIED STRUVE FUNCTION

The modified Struve function $\mathbf{L}_v(z)$ is provided by **LVOZ** and defined by the power series expansion

$$L_v(z) = (z/2)^{v+1} \sum_{n=0}^{\infty} \frac{(z/2)^{2n}}{\Gamma(n+3/2)\ \Gamma(n+v+3/2)},$$

where for integer $k \geq 0$ the case $v = -(k+1/2)$ is evaluated as $\mathbf{L}_{-(k+1/2)}(z) = I_{k+1/2}(z)$. For large arguments of $|z|$, the asymptotic expansion

$$L_v(z) \approx I_{-v}(z) + \frac{1}{\pi} \sum_{k=0}^{m-1} \frac{(-1)^{k+1}\ \Gamma(k+1/2)}{\Gamma(v+1/2-k)\ (z/2)^{2k-v+1}} \qquad |\arg z| < \pi/2$$

may be used for $|\arg z| < \pi/2$. Neither **HVOZ** nor **LVOZ** includes asymptotic expansions.

ASSOCIATED LEGENDRE (SPHERICAL HARMONICS) MENU { FTNS LGDR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
$P_\nu^\mu(x)$ ON THE CUT	**PμVX**(μ,ν,x)	μ, ν ∈ ℂ −1 < x < 1	VALUE
PμVX(3,3,0.5) = −9.7427857926		PμVX(0.5 , (1,2) , −0.5) = −28.2756816323	
$Q_\nu^\mu(x)$ ON THE CUT	**QμVX**(μ,ν,x)	μ, ν ∈ ℂ −1 < x < 1	VALUE
QμVX(3,1,0.5) = −6.1584028714		QμVX(0.5 , (1,2) , −0.5) = (0,44.3949135345)	
RECURRENCE UP	**PQUP**(P_1, P_2, μ, ν_1, ν, z)	P_1, P_2, μ ∈ ℂ ν_1, ν, z ∈ ℂ	VALUE

PQUP provides upward recurrence over degree ν for fixed order μ and argument z for all four of the Legendre functions provided in this menu. Since the accuracy of **F2F1** degrades for large values of a and b, **PQUP** provides a means to evaluate the Legendre functions of large degree. AMS 55 states that upward recurrence is not stable for **QμVX** for positive arguments, but is stable for negative arguments. The recurrence formula used by **PQUP** for $P_\nu^\mu(z)$, $Q_\nu^\mu(z)$, $p_\nu^\mu(z)$, and $q_\nu^\mu(z)$ is

$$(\nu-\mu)\,P_\nu^\mu(z) = (2\nu-1)\,z\,P_{\nu-1}^\mu(z) - (\nu+\mu-1)\,P_{\nu-2}^\mu(z);$$

so clearly, if at any step ν = μ, you will get a divide–by–zero error.

ASSOCIATED LEGENDRE (SPHERICAL HARMONICS) MENU { FTNS LGDR }

FUNCTION	COMMAND	INPUTS	OUTPUTS

Let Ξ denote any of the four Legendre functions in this menu. Then the inputs are $P_1 = \Xi_{v1}^{\mu}(z)$, $P_2 = \Xi_{v1+1}^{\mu}(z)$, $\mu \in \mathbb{C}$, $v1 \in \mathbb{C}$, $v \in \mathbb{C}$, and $z \in \mathbb{C}$ where $v - v1$ must be a positive integer greater than 1. The output is $\Xi_v^{\mu}(z)$.

PQUP(PμVZ((0.2,0.4) , (0.1,0.3) , 0.5) , PμVZ((0.2,0.4) , (1.1,0.3) , 0.5) , (0.2,0.4) , (0.1,0.3) , (10.1,0.3) , 0.5) = (–0.810572838932,0.398391225278) which agrees to 8 digits and is more accurate than PμVZ((0.2,0.4) , (10.1,0.3) , 0.5) = (–0.810572835215,0.398391226872)

PQUP(QμVZ((0.2,0.4) , (0.1,0.3) , 0.5) , QμVZ((0.2,0.4) , (1.1,0.3) , 0.5) , (0.2,0.4) , (0.1,0.3) , (10.1,0.3) , 0.5) = (–0.162225425703,0.198658115425) which agrees to 11 digits with
QμVZ((0.2,0.4) , (10.1,0.3) , 0.5) = (–0.162225425704,0.198658115428)

PQUP(QμVX(3 , 2 , +0.5) , QμVX(3 , 3 , +0.5) , 3 , 2 , 10 , +0.5) =
(–244.565823867,–2.77433298954E–9) and
QμVX(3,10,+0.5) = (–244.565823878,–2.E–9)

PQUP(QμVX(3 , 2 , –0.5) , QμVX(3 , 3 , –0.5) , 3 , 2 , 10 , –0.5) =
(–244.565823867,–2.77433298954E–9) and
QμVX(3,10,–0.5) = (–244.565823904,–2.E–9)

PQUP(QμVZ(3 , 2 , (0.5,±1E–499)) , QμVZ(3 , 3 , (0.5,±1E–499)) , 3 , 2 , 10 , (0.5,±1E–499)) = (–407.13092507,∓244.565823867).

Thus, "on the cut" we get with $\mu = 3$:

$$0.5\, e^{-i3\pi} [\, i\, Q_{10}^{3}(0.5 + i\ 1E{-}499) - i\, Q_{10}^{3}(0.5 - i\ 1E{-}499) \,]$$
$$= (-244.565823867,-2.34829958282E{-}9)$$

which is accurate to about 10 digits.

ASSOCIATED LEGENDRE (SPHERICAL HARMONICS) MENU { FTNS LGDR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
$P_v^{\mu}(z)$	**PμVZ**(μ,v,z)	μ, v, z ∈ **C**	VALUE

PμVZ(1,3,(−1,2)) = (87.7653601152,−25.4314458127)

PμVZ((0.2,0.4) , (0.1,0.3) , −0.5) = (0.699644761889,−2.25868375442)

$Q_v^{\mu}(z)$	**QμVZ**(μ,v,z)	μ, v, z ∈ **C**	VALUE

QμVZ(0.5 , (1,2) , (−1,1)) = (4.65272798248,17.69617794)

QμVZ((0.2,0.4) , (0.1,0.3) , −0.5) = (−6.40043495202E−2,−0.605644017674)

UP DIRECTORY	**UPDIR**	NONE	PARENT MENU

For non-negative integers μ and v ≥ μ, the following programs combined with **PQUP** provide very fast computation of $p_v^{\mu}(x)$ and $P_v^{\mu}(z)$. The equations are given above for $p_n^n(x)$, $p_{n+1}^n(x)$, $P_n^n(z)$ and $P_{n+1}^n(z)$.

```
PMMX: « → μ X « IF μ THEN '(-1)^μ×¡¡(2×μ-1)×(1-X^2)^(μ/2)' →NUM
      ELSE 1 END DUP 'X×(2×μ+1)' →NUM × » »

PMMZ: « → μ Z « IF μ THEN '¡¡(2×μ-1)×(Z+1)^(μ/2)×(Z-1)^(μ/2)' →NUM
      ELSE 1 END DUP 'Z×(2×μ+1)' →NUM × » »
```

ASSOCIATED LEGENDRE (SPHERICAL HARMONICS) MENU { FTNS LGDR }

FUNCTION	COMMAND	INPUTS	OUTPUTS

EVALUATION OF $p_n^m(x)$ AND $P_n^m(z)$

For natural numbers m and n with $n \geq m + 2$, the below programs provide fast evaluation of the Legendre functions $p_n^m(x)$ and $P_n^m(z)$:

PMNX: < → m n x < m x PMMX m m n x PQUP > >

PMNZ: < → m n z < m z PMMZ m m n z PQUP > >

For definitions see Chapters 8 of Abramowitz and Stegun, *Handbook of Mathematical Functions*, AMS 55, 1964.

Erdelyi, A., Magnus,W., Oberhettinger, F., and Tricomi, F., *Higher Transcendental Functions*, Volume 1, New York, McGraw–Hill, 1953.

Morse, P., and Feshbach, H., *Methods of Theoretical Physics*, New York, McGraw–Hill, 1953.

STRUVE FUNCTION MENU
{ FTNS STRUV }

FUNCTION	COMMAND	INPUTS	OUTPUTS
$H_v(z)$	**HVOZ**(v,z)	$v \in \mathbf{C}$ $z \in \mathbf{C}$	VALUE
HVOZ((0.1,0.3) , (4,3)) = (–0.316695241869,–2.20192837971)			
$L_v(z)$	**LVOZ**(v,z)	$v \in \mathbf{C}$ $z \in \mathbf{C}$	VALUE
LVOZ((0.1,0.3) , (4,3)) = (–8.93683914434,4.69252342842)			
See page 149 for asymptotic expansions for **HVOZ** and **LVOZ**.			
$Y_v(z)$	**YVOZ**(v,z)	v, $z \in \mathbf{C}$	VALUE
$I_v(z)$	**IVOZ**(v,z)	v, $z \in \mathbf{C}$	VALUE
$\Gamma(z)$	**GAMMA**(z)	$z \in \mathbf{C}$	VALUE
See Chapters 5 and 7 for definitions.			
UP DIRECTORY	**UPDIR**	NONE	PARENT MENU
For definitions see Chapters 12 of Abramowitz and Stegun, *Handbook of Mathematical Functions*, AMS 55, 1964.			

16

ORTHOGONAL POLYNOMIALS

INTRODUCTION

This chapter presents the 15 commands in the orthogonal polynomial menu POLY. The orthogonal polynomials, besides solving various differential equations, are also useful in the approximation of functions. These polynomials can be evaluated in either symbolic form (the equation itself) by using symbolic inputs or in numerical form by using numbers for inputs. The command **PMAT** in the MISC menu of Chapter 17 performs polynomial approximations for all the polynomials in this menu. The **CHEBY** command in the WIND menu provides the discrete orthogonal Chebyshev polynomial. **FMAT**, a faster, less general version of **PMAT**, is also provided. See Chapter 28.

JACOBI: $P_n^{(\alpha,\beta)}(x)$ and $G_n(p,q,x)$

GEGENBAUER ULTRASPHERICAL: $C_n^{(\alpha)}(x)$

CHEBYSHEV: $T_n(x)$, $U_n(x)$, $C_n(x)$, $S_n(x)$, and shifted $T_n^*(x)$, $U_n^*(x)$

LEGENDRE: $P_n(x)$

LAGUERRE: $L_n^{(\alpha)}(x)$ and $L_n(x)$

HERMITE: $H_n(x)$ and $He_n(x)$

BERNOULLI AND EULER POLYNOMIALS

These polynomials are not orthogonal, but are available in the NUMB menu in Chapter 18.

CONTINUOUS CHEBYSHEV POLYNOMIALS

The $T_n(x)$ is also available in coefficient list form as command **TOFXL** in Chapter 28.

DISCRETE CHEBYSHEV POLYNOMIALS

The discrete Chebyshev polynomials are provided by the command **CHEBY** in Chapter 28.

ORTHOGONAL POLYNOMIALS MENU { FTNS POLY }

FUNCTION	COMMAND	INPUTS	OUTPUTS
The below commands provide polynomials in variable v of degree n. The inputs to these commands may be numbers or symbols. In MATHLIB, the degree is generally the last argument for polynomial commands.			
Jacobi $P_n^{(\alpha,\beta)}(x)$	**PαβX**(α,β,v,n)	α β v n	VALUE OR EQUATION
Jacobi $G_n(p, q, x)$	**GPQX**(p,q,v,n)	p q v n	VALUE OR EQUATION
Gegenbauer Ultraspherical $C_n^{(\alpha)}(x)$	**CαOX**(α,v,n)	α v n	VALUE OR EQUATION
Chebyshev $T_n(x)$	**TOFX**(v,n)	v n	VALUE OR EQUATION
Chebyshev $U_n(x)$	**UOFX**(v,n)	v n	VALUE OR EQUATION
Chebyshev $C_n(x)$	**COFX**(v,n)	v n	VALUE OR EQUATION
Chebyshev $S_n(x)$	**SOFX**(v,n)	v n	VALUE OR EQUATION
SHIFTED $T_n^*(x)$	**TSFX**(v,n)	v n	VALUE OR EQUATION
SHIFTED $U_n^*(x)$	**USFX**(v,n)	v n	VALUE OR EQUATION
Legendre $P_n(x)$	**POFX**(v,n)	v n	VALUE OR EQUATION

ORTHOGONAL POLYNOMIALS MENU { FTNS POLY }

FUNCTION	COMMAND	INPUTS	OUTPUTS
Laguerre $L_n^{(\alpha)}(x)$	**LαOX**(α,v,n)	α v n	VALUE OR EQUATION
Laguerre $L_n(x)$	**LOFX**(v,n)	v n	VALUE OR EQUATION
Hermite $H_n(x)$	**HOX**(v,n)	v n	VALUE OR EQUATION
Hermite $He_n(x)$	**HEOX**(v,n)	v n	VALUE OR EQUATION
UP DIRECTORY	**UPDIR**	NONE	PARENT MENU

SOME RECURRENCE RELATIONS

$$T_{n+1}(x) = 2x\,T_n(x) - T_{n-1}(x) \qquad U_{n+1}(x) = 2x\,U_n(x) - U_{n-1}(x)$$

$$S_{n+1}(x) = x\,S_n(x) - S_{n-1}(x) \qquad C_{n+1}(x) = x\,C_n(x) - C_{n-1}(x)$$

$$(n+1)\,P_{n+1}(x) = (2n+1)x\,P_n(x) - n\,P_{n-1}(x)$$

$$(n+1)\,L_{n+1}^{(\alpha)}(x) = [(2n+\alpha+1) - x]\,L_n^{(\alpha)}(x) - (n+\alpha)\,L_{n-1}^{(\alpha)}(x)$$

$$H_{n+1}(x) = 2x\,H_n(x) - 2n\,H_{n-1}(x) \qquad He_{n+1}(x) = x\,He_n(x) - n\,He_{n-1}(x)$$

For definitions see Chapters 22 of Abramowitz and Stegun, *Handbook of Mathematical Functions*, AMS 55, 1964.

17

APPROXIMATION OF FUNCTIONS AND DATA SETS

INTRODUCTION

This chapter presents the 30 commands in the MISC menu. Most of the MISC commands provide random data and deterministic function approximation techniques. They include polynomial, rational, and least squares approximation. The least squares approximation commands are very general and allow complicated polynomial and nonlinear function approximations. MISC is also the home of a number of miscellaneous commands. Related commands are given in the ALGB, LINAG, PROC, FILTR, and WIND menus.

POLYNOMIAL APPROXIMATIONS

PMAT computes the forward transformation matrix required for orthogonal polynomial approximations, including economized minimax ones. A faster but less general version of **PMAT** is the **FMAT** command discussed in Chapter 28. **FEVAL** evaluates polynomial approximations.

RATIONAL APPROXIMATIONS

RATAP provides rational approximations of functions. For the same number of terms, it is also the inverse of the polynomial long divide command, **PLDVD**, given in Chapter 19. **REVAL** evaluates rational approximations.

LEAST SQUARES APPROXIMATIONS WITH ANALYSIS OF VARIANCE

LREG provides multiple linear regression of complex data, including generation of the analysis of variance table, standard errors or singular values, and both T and F statistics for the fit. **LSSOV** extends the **LREG** analysis to combinations of polynomials and nonlinear functions. Commands for least squares filter design are found in the WIND menu discussed in Chapter 28.

TAYLOR AND MACLAURIN SERIES

TALR1 and **TALR2** provide one– and two–dimensional Taylor series expansions. **COEFL** and **COEFV** provide Maclaurin series expansions. These commands are far faster than the HP command **TAYLR**. See also page 450.

SYMBOLIC EQUATIONS

Given a polynomial list of coefficients and a symbol, **PEQN** and **XEQN** convert them into pretty symbolic equations.

GENERATION OF COMPLEX SERIES

CSERS is a very general command that will reduce any HP 48, Math Library, or VAR directory user function into a real or complex time series for processing and plotting.

MISCELLANEOUS FUNCTIONS

Commands include **TIMIT**, the Marcum Q function **MQ**, complex permutations **PERMF**, Pochhammer's symbol **POCH**, and the sorting commands **UNIQE**, **SSORT**, and **ZSORT**.

DISCRETE CHEBYSHEV POLYNOMIAL APPROXIMATIONS

Discrete Chebyshev polynomial approximations are discussed in Chapter 28.

MISCELLANEOUS FUNCTIONS AND APPROXIMATIONS MENU { FTNS MISC }

FUNCTION	COMMAND	INPUTS	OUTPUTS
EVALUATE SERIES	**CSERS**(L,δ)	LIST L δ∈[0,1]	[VALUES]

L = { { DIRECTORY PATH } 'FUNCTION' INDEPENDENT VARIABLE START VALUE FINAL VALUE NUMBER OF POINTS } For example: { {HOME} '**DNUK**$_j$(z,.2)' z 0 7.6 20 } computes the library elliptic function dn(z, .2) versus z at the values z = 0, .4, .8, . . ., 7.6 and stores them in the vector [dn(0, .2) dn(.4, .2) . . ., dn(7.6, .2)]. **CSERS** can be called from any user directory, and the function need not be in the same user directory. HP 48 and MATHLIB functions are all in { HOME }. δ = 0 evaluates the independent variable with a linear scale, while δ = 1 evaluates the independent variable with a logarithmic scale. See Chapter 2 for examples.

TAYLOR SERIES	**TALR1**(F,L)	F AND L	POLYNOMIAL

F is any function and L is the list { INDEPENDENT VARIABLE VALUE TO EXPAND ABOUT NUMBER OF TERMS }.
For example: F = 'EXP(−(X−1)^2)' and L = { X 1 8 } expands the function of X about X_0=1 and computes the first 8 terms =

'1 −(−1+X)^2 +.5(−1+X)^4 −1.66666666667(−1+X)^6 +4.16666666667E−2(−1+X)^8'.

TAYLOR SERIES	**TALR2**(F,L1,L2)	F, L1, L2	POLYNOMIAL

F is any function of two variables, and L1, L2 are the corresponding lists. For example:
'EXP(X+iY)' L1 = { X 0 4 } L2 = { Y '2×π' 3 }
yields a two−dimensional Taylor series expansion of EXP(X + iY).

MISCELLANEOUS FUNCTIONS AND APPROXIMATIONS MENU { FTNS MISC }

FUNCTION	COMMAND	INPUTS	OUTPUTS
POLYNOMIAL EVALUATION	**FEVAL**(L,V)	L AND V	POLYNOMIAL OR NUMBER

Given coefficient list L = { a_0 a_1 a_2 ... a_{n-1} } and independent variable V, FEVAL computes $\Sigma(k=0,n-1,a_k\ V\text{^}k)$. CAREFULLY NOTE THE ORDER OF THE COEFFICIENTS. While **FEVAL** can be used for symbolic values of V, **XEQN** below is preferred. For example:

FEVAL({ 4 5 2 7 } , z) = '4 + (5 + (2 + 7z) z) z' = $4 + 5z + 2z^2 + 7z^3$.

EXPAND COLLECT COMPLETELY	**EXCO**(F)	F IS ANY FUNCTION	FUNCTION

EXCO uses **MULTI** to first algebraically expand F until there is no change and then algebraically collect F until there is no change. See page 570 of the HP 48 owner's manual.

EXECUTION TIME	**TIMIT**(P)	P = PROGRAM OR 'USER COMMAND'	2: RESULT 1: TIME IN SEC

TIMIT is the only MATHLIB command that modifies the HP 48 word size. It sets the word size to 64, which is the default word size. Then it evaluates MEM DROP to start the garbage collector, which improves the time reliability, and uses TICKS to compute the execution time of program P. TIMIT(« 2 SIN ») and TIMIT(« 4 (1,2) JNOZ ») are examples.

MISCELLANEOUS FUNCTIONS AND APPROXIMATIONS MENU { FTNS MISC }

FUNCTION	COMMAND	INPUTS	OUTPUTS
COEFFICIENTS	**COEFL**(F,V,N)	F, V, N	COEFFICIENTS

Reduces a function F in variable V to a coefficient list of length N + 1, which is its Maclaurin series expansion. See **FEVAL** for coefficient order. If F = '$4z^4-2z^2+5z-6$' then COEFL(F,z,4) = { –6 5 –2 0 4 }, COEFL(F,z,6) = { –6 5 –2 0 4 0 0 } and COEFL('EXP(–t)',t,4) = { 1 –1 .5 –1/6 1/24 }.

COEFFICIENT VECTOR	**COEFV**(F,V,N)	F V N	ROW VECTOR

Same as **COEFL** except output is in row vector form for multiplying in polynomial applications.

ORTHOGONAL POLYNOMIAL APPROXIMATION MATRIX	**PMAT**(L,F,N,O)	4: COEF LIST 3: POLYNOMIAL 2: DEGREE N 1: OFFSET O	FORWARD TRANSFORM MATRIX

Given any orthogonal polynomial function F such as found in the { FTNS POLY } menu with coefficient list L, **PMAT** computes the forward transformation matrix that can be used for polynomial approximations. For example, L = { 1 2 } and F = "P$\alpha\beta$X" specifies the Jacobi polynomial $P_n^{(\alpha,\beta)} = P_n^{(1,2)}$, while { } "TOFX" specifies the Chebyshev polynomial $T_n(x)$. Polynomials may also be defined in the current user directory, but must have the same form as those in the Math Library. N + 1 is the number of terms and equals the dimensions of the transformation matrix. The forward matrix expresses each of the orthogonal polynomials in terms of a power series in x^n, where x is some arbitrary variable. The inverse matrix (use **INV** or **MINV**) expresses the x^n in terms of the polynomials.

MISCELLANEOUS FUNCTIONS AND APPROXIMATIONS MENU { FTNS MISC }

FUNCTION	COMMAND	INPUTS	OUTPUTS

See also the **FMAT** command in the WIND menu in Chapter 28 for a faster, less general version of **PMAT**.

For example, consider a Chebyshev polynomial economized minimax approximation of cos(x), which has an error less than 5E–5.

« { } "TOFX" 8 0 PMAT DUP INV 'COS(x)' 'x' 8 COEFV SWAP × DUP →VTR V→L "T" PEQN HALT DROP HALT SWAP × →VTR V→L 'x' XEQN »

The program will halt displaying the cos function in terms of the Chebyshev polynomials T_n.

'1.93762420065E–7×T8 – 4.18526787032E–5×T6 + 4.95334201284E–3×T4 – .229806857642×T2 + .765197753904×T0'.

Push CONT and edit the vector, setting all values to zero after column 5. Push CONT again and observe the approximation:

'3.96267361025E–2 x^4 – .499240451387 x^2 + .999957953559'.

The transformation matrices need only be computed once and then may be stored in VAR. These are the same matrices given in Chapter 22 of AMS 55, but MATHLIB computes any degree.

Discrete Chebyshev polynomial approximations are discussed in Chapter 28.

MISCELLANEOUS FUNCTIONS AND APPROXIMATIONS MENU { FTNS MISC }

FUNCTION	COMMAND	INPUTS	OUTPUTS

Offset O provides the capability of computing the coefficients of an offset variable. For example, **PαβX** gives the Jacobi polynomial in standard form as $\Sigma a_k(x-1)^k$. With O=0, **PMAT** provides the coefficients of $\Sigma b_k x^k$, which is numerically more efficient, but does not allow direct comparison with standard tabulations such as in AMS 55. Offset O solves that problem. To reproduce Table 22.1 in AMS 55 and also the $P_5^{(1,1)}$ example given there, evaluate the following program:

« { 1 1 } "PαβX" 5 1 PMAT DUP INV OVER 1 2 8 48 384 3840 6 →ARRY D→M SWAP × DUP 6 EROW V→L 4 SRND 'Z-1' XEQN 3840 / ».

LIST ROUND	**SRND**(L,N)	L N	ROUNDED LIST

Given list L of numbers and integer N, **SRND** applies « N RND » to each element of list L. See page 148 of the HP 48 owner's manual.

POLYNOMIAL EQUATION	**PEQN**(L,S)	POLYNOMIAL LIST STRING S	PRETTY EQUATION

PEQN({ (-7,-24) (19,8) (-5,0) 1 } , "T") = 'T3 + (-5,0) T2 + (19,8) T1 + (-7,-24) T0'

POLYNOMIAL EQUATION	**XEQN**(L,V)	POLYNOMIAL LIST VARIABLE V	PRETTY EQUATION

XEQN({ (-7,-24) (19,8) (-5,0) 1 } , z) = z^3 + (-5,0) z^2 + (19,8) z + (-7,-24)

MISCELLANEOUS FUNCTIONS AND APPROXIMATIONS MENU { FTNS MISC }

FUNCTION	COMMAND	INPUTS	OUTPUTS
RATIONAL APPROXIMATION	**RATAP**(L,n)	L n	2: P COEF LIST 1: Q COEF LIST

Rational approximations are often more accurate than polynomial approximations. A rational function r(z) of degree N approximating a function f(z) has the form;

$$r(z) = \frac{p(z)}{q(z)} = \frac{p_0 + p_1 z + p_2 z^2 + \ldots, + p_{n-1} z^{n-1}}{1 + q_1 z + q_2 z^2 + \ldots, + q_m z^m},$$

where m + n = N, and without loss in generality we have set $q_0 = 1$. For m = 0, r(z) is just a polynomial in z so polynomial approximations may be regarded as a special case of rational approximations. **RATAP** uses the Pade approximation technique, which chooses the N parameters of r(z) so that the Maclaurin expansions of r(z) and f(z) agree for the first N terms. The input list L is the Maclaurin series expansion of f(z) obtained from **COEFL** and has size N. Parameter n is the number of coefficients in the numerator as shown in the above equation. The user may use N, the size of L, and n to determine the degree of the numerator and denominator of r(z). $n \in [1, N]$.

The inverse process of computing the input coefficient list from the rational approximation can be done with the **PLDVD** command in the ALGB menu. Consider the approximation of e^{-x}:

COEFL(' EXP(–x)' , x , 5) = { 1 –1 1/2 –1/6 1/24 –1/120 }. Then we have the results:

$$\text{RATAP(L,4)} = \begin{cases} \text{2: \{ 1 } & \text{-0.6 } & \text{0.15 } & \text{-1/60 \}} \\ \text{1: \{ 1 } & \text{0.4 } & \text{0.05 \}.} \end{cases}$$

Using **FEVAL** to evaluate the Maclaurin series approximation of e^{-t} represented by L in the range [0.2, 1], we find that the maximum absolute value of error is 1.21E–3, whereas using **REVAL** to evaluate the rational approximation results in a maximum error magnitude of 6.33E–5.

MISCELLANEOUS FUNCTIONS AND APPROXIMATIONS MENU { FTNS MISC }

FUNCTION	COMMAND	INPUTS	OUTPUTS
RATIO EVALUATION	**REVAL**(P,Q,v)	P Q v	VALUE

REVAL is like **FEVAL** except that it evaluates the ratio of two polynomials P/Q such as those output from **RATAP**. For the above example, REVAL(P,Q,1) = .367816091955.

LINEAR REGRESSION	**LREG**(M,θ,ε)	M = MATRIX θ = FLAG ε ≥ 0	3: MATRIX 2: σ VECTOR 1: COEF VECTOR

LREG is a general multiple-linear regression program. For simple single-variable linear regression, the HP 48 has the functions built in along with several transformed fits. Linear, logarithmic, exponential, power, and best are all available. See Chapter 21 of the owner's manual.

LREG and **LSSOV** are designed for more complicated models with multiple parameters. **LREG** also computes a complete analysis of variance table with sums of squares and coefficient standard deviations for performing F and T tests on the estimate. The simplest linear regression model is $y = a + bx$. Given a set of complex x values and a set of corresponding complex y values, we find the values of a and b which minimize the mean square error between the given set of y values and the predicted set of y values using the equation $y = a + bx$. Multiple regression extends this idea to several sets of x values, say $x_1, x_2, \ldots, x_n$. We now choose the coefficients $a, b_1, b_2, \ldots, b_n$ to minimize the mean square error between the given y data and the equation (model) $y = a + b_1x_1 + b_2x_2 + \ldots, + b_nx_n$. Once this basic modeling capability is in place, it can be extended to more complicated models. **LSSOV** on page 175 extends the model to the form:

MISCELLANEOUS FUNCTIONS AND APPROXIMATIONS MENU { FTNS MISC }

FUNCTION	COMMAND	INPUTS	OUTPUTS

$$g(y) = a + b_1 f_1(x_1, x_2, \ldots, x_n) + \ldots, + b_q f_q(x_1, x_2, \ldots, x_n),$$

where g is an arbitrary function of y, and $f_1, f_2, \ldots, f_n$ are arbitrary functions of $x_1, x_2, \ldots, x_n$. Exponentiation of both sides of the linear model gives the equation

$$g'(y) = e^{g(y)} = e^a \times \mathrm{EXP}(b_1 f_1(x_1, x_2, \ldots, x_n)) \times \ldots, \times \mathrm{EXP}(b_q f_q(x_1, x_2, \ldots, x_n)),$$

so the basic model can be transformed into exponential, logarithmic, and power law models. Numerous commands are available in the VSAG and MSAG menus for transforming your data.

General nonlinear functions and data can be locally modeled as multivariate polynomials by using a Taylor or Maclaurin series expansion.

$$y = f(x_1, x_2, \ldots x_n) \approx f(0, 0, \ldots 0) + \frac{\partial f}{\partial x_1} x_1 + \frac{\partial f}{\partial x_2} x_2 + \ldots + \frac{\partial f}{\partial x_n} x_n + \ldots$$
$$\approx a + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n + c_1 x_1 x_2 + \ldots$$

LREG assumes that the input data in matrix M lies vertically with the y data being the last column and the previous columns being the corresponding sets of x data. With the matrix editor and GO↓ set, it is easy to enter your data. However, you may prefer to enter your data horizontally with GO→ set and use **TRNP** to transpose your data into input form. In this case, y is entered as the last row. M can be expressed in terms of X and Y using the commands **CCMB** and **CSPLT**:

M = CCMB(X , Y) CSPLT(M , n) = { 2: X ; 1: Y.

THERE MUST BE AS MUCH DATA AS THE NUMBER OF COEFFICIENTS TO BE COMPUTED. **LREG** and **LSSOV** do not compute underdetermined cases.

MISCELLANEOUS FUNCTIONS AND APPROXIMATIONS MENU { FTNS MISC }

FUNCTION	COMMAND	INPUTS	OUTPUTS

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} \qquad Z = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1n} \\ 1 & x_{21} & x_{22} & \cdots & x_{1n} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} \qquad Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

The M matrix, of size m × (n+1), is input by the user, and the Z matrix, which is the X matrix augmented with a column of 1 values, is computed and used by **LREG** as explained on the next page. There are two distinct cases which must be considered separately since the equations are different. Flag θ = 1 specifies that the user wants a fit with parameter a not constrained to zero, whereas θ = 0 specifies that the user wants a fit through the origin with a = 0. When θ = 0, the X matrix is used for the regression computation, whereas when θ = 1, the Z matrix is used.

$$\hat{y}_k = \hat{a} + \hat{b}_1 x_{k1} + \hat{b}_2 x_{k2} + \ldots + \hat{b}_n x_{kn} \quad \text{where} \quad k=1, 2, \ldots m \quad \hat{a}=0 \quad \text{if} \quad \theta=0$$

$$\bar{y} = \frac{1}{m} \sum_{k=1}^{m} y_k \qquad SSE = \sum_{k=1}^{m} (y_k - \hat{y}_k)^2$$

$$SST = SSR + SSE = \begin{cases} \sum_{k=1}^{m} (y_k - \bar{y})^2 & \theta=1 \\ \sum_{k=1}^{m} y_k^2 & \theta=0 \end{cases} \qquad SSR = \begin{cases} \sum_{k=1}^{m} (\hat{y}_k - \bar{y})^2 & \theta = 1 \\ \sum_{k=1}^{m} \hat{y}_k^2 & \theta = 0 \end{cases}$$

where $\hat{y}_k$ is the least squares estimate corresponding to the kth set (row) of x data, SST is the total sum of squares, SSR is the sum of squares due to regression, and SSE is the sum of squares due to error.

MISCELLANEOUS FUNCTIONS AND APPROXIMATIONS MENU { FTNS MISC }

FUNCTION	COMMAND	INPUTS	OUTPUTS

The solution coefficient vector is given by

$$s = \begin{cases} [\,\hat{b}_1 \;\; \hat{b}_2 \;\; \ldots \;\; \hat{b}_n\,]^T = C\,X^H\,Y \quad C = (X^H X)^{-1} \quad \theta=0 \\ [\,\hat{a} \;\; \hat{b}_1 \;\; \hat{b}_2 \;\; \ldots \;\; \hat{b}_n\,]^T = C\,Z^H\,Y \quad C = (Z^H Z)^{-1} \quad \theta=1 \end{cases} \qquad \mathrm{DET}(C^{-1}) \neq 0,$$

where superscript T denotes transpose, and superscript H denotes Hermitian transpose. When C exists, the solution can be performed by command **OSOVR** in the LINAG menu. The user specifies this means of solution by an $\varepsilon = 0$ input. Even when C^{-1} is either singular or nearly singular, a least squares solution still exists and can be computed via singular value decomposition (SVD) using **LSVDR**. By choosing ε to be a small positive number, the user specifies a SVD solution using ε as the convergence test parameter. See the LINAG menu for more details.

The output of **LREG** is a sum of squares matrix, a vector of coefficient standard deviations, and the solution coefficient vector. The elements of the matrix are

$$\begin{array}{rl} \textbf{Regression :} & \begin{bmatrix} \textbf{DFR} & \textbf{SSR} & \textbf{MSR} \\ \textbf{DFE} & \textbf{SSE} & \textbf{MSE} \\ \textbf{Total} & \textbf{SST} & \textbf{F} \end{bmatrix} \end{array} \quad \textbf{MSR} = \frac{\textbf{SSR}}{\textbf{DFR}} \quad \textbf{MSE} = \frac{\textbf{SSE}}{\textbf{DFE}} \qquad \begin{array}{l} \textbf{DFR} = n \\ \textbf{DFE} = m - n - \theta, \end{array}$$

where m and n are the dimensions of X, and θ is input as 1 or 0.

When $\varepsilon = 0$, the elements of the σ vector returned to level 2 of the stack are the standard errors of the regression coefficients returned to level 1 of the stack and equal $\sigma_k = \sqrt{(\mathrm{MSE}\ C_{kk})}$, corresponding to each coefficient. Defining $b_0 = a$, then $t_k = b_k / \sigma_k$ has the t distribution with DFE degrees of freedom where it is understood that t_0 does not exist for $\theta = 0$. **UτPT** is available to compute the probability, and **IUτPT** is available for computing confidence intervals.

MISCELLANEOUS FUNCTIONS AND APPROXIMATIONS MENU { FTNS MISC }

FUNCTION	COMMAND	INPUTS	OUTPUTS

When $\varepsilon > 0$, the elements of the σ vector are the singular values of the X matrix when $\theta = 0$ and the singular values of the Z matrix when $\theta = 1$. The square of these singular values are the eigenvalues of C^{-1}, but there is no one–to–one relationship to the solution coefficients other than through the unitary SVD matrices used in the computation of the solution (see the discussion in the LINAG menu). The SVD solution is significantly slower, and for $\theta = 0$, the number of X columns must be greater than 1 ($n > 1$). Nevertheless, if C^{-1} is nearly singular, it is the solution of choice.

The F ratio equals MSR/MSE with DFR and DFE degrees of freedom. The probability and confidence interval can be computed using **UτPF** and **IUτPF**.

The coefficient of multiple determination R^2 and its adjusted version R_a^2 are given by:

$$R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST}, \qquad R_a^2 = 1 - \frac{SSE/(m-n-\theta)}{SST/(m-\theta)}.$$

CONSTRAINTS: $m - n - \theta > 0$ for positive degrees of freedom, $n + \theta \geq 2$ for a SVD ($\varepsilon > 0$) solution.

For example, suppose we are given the Y data [(–6,42) (8,33) (–20,41) (29,–1)] and the associated X data [(1,1) (3,4) (7,9) (2,–5)] and [(4,7) (5,2) (1,4) (3,–2)]. Then:

$$\mathbf{M} = \begin{bmatrix} (1,1) & (4,7) & (-6,42) \\ (3,4) & (5,2) & (8,33) \\ (7,9) & (1,4) & (-20,41) \\ (2,-5) & (3,-2) & (29,-1) \end{bmatrix}$$

MISCELLANEOUS FUNCTIONS AND APPROXIMATIONS MENU { FTNS MISC }

FUNCTION	COMMAND	INPUTS	OUTPUTS

$$\text{LREG(M,0,0)} = \begin{cases} 3: \begin{bmatrix} 2 & 5873.0 & 2936.5 \\ 2 & 2.9842 & 1.4921 \\ 4 & 5876 & 1968.0 \end{bmatrix} \\ 2: [\ 0.1147 \quad 0.1405\ ] \\ 1: \{\ (0.9637, 2.033) \quad (3.877, 2.937)\ \} \end{cases}$$

$$\text{LREG(M,0,1E-8)} = \begin{cases} 3: \begin{bmatrix} 2 & 5873.0 & 2936.5 \\ 2 & 2.9842 & 1.4921 \\ 4 & 5876 & 1968.0 \end{bmatrix} \\ 2: [\ 19.50 \quad 9.07\ ] \\ 1: \{\ (0.9637, 2.033) \quad (3.877, 2.937)\ \} \end{cases}$$

$U\tau PF(2,2,1968) = 5.079E{-}4$, $U\tau PT(2,t_1) = 2.590E{-}3$, $U\tau PT(2,t_2) = 8.336E{-}4$

$$\text{LREG(M,1,0)} = \begin{cases} 3: \begin{bmatrix} 2 & 2538.2 & 1269.1 \\ 1 & 1.3196 & 1.3196 \\ 3 & 2539.5 & 961.76 \end{bmatrix} \\ 2: [\ 0.9158 \quad 0.1109 \quad 0.1717\ ] \\ 1: \{\ (-0.8390, -0.5951) \quad (0.9897, 2.021) \quad (4.000, 2.938)\ \} \end{cases}$$

EXCLUDED CASES IN SIMPLE ENGLISH

If M is 1×2, then **LREG** will only find a solution for $\theta = \varepsilon = 0$. If M is 1×3, then there is no **LREG** solution because this is an underdetermined case. If M is 2×2 or 3×2, then **LREG** can find a solution for all cases except $\theta = 0$ and $\varepsilon > 0$. It is possible to construct pathological cases which will terminate with a divide-by-zero error when $\varepsilon = 0$. For these cases, slightly change the data, or use the SVD solution.

MISCELLANEOUS FUNCTIONS AND APPROXIMATIONS MENU { FTNS MISC }

FUNCTION	COMMAND	INPUTS	OUTPUTS

$$\text{LREG(M,1,1E-8)} = \begin{cases} 3: \begin{bmatrix} 2 & 2538.2 & 1269.1 \\ 1 & 1.3196 & 1.3196 \\ 3 & 2539.5 & 961.76 \end{bmatrix} \\ 2: [\ 18.41 \quad 1.430 \quad 8.561\ ] \\ 1: \{\ (-0.8390,-0.5951) \quad (0.9897,2.021) \quad (4.000,2.938)\ \} \end{cases}$$

$U\tau PF(2,1,961.76) = 2.280E{-}2$, $U\tau PT(1,t_0) = 0.4631$, $U\tau PT(1,t_1) = 3.134E{-}2$, $U\tau PT(1,t_2) = 2.201E{-}2$

The F and T statistics can easily be computed by the **FSTAT** and **TSTAT** commands below.

As stated above, the condition for a positive error degrees of freedom is $m - n - \theta > 0$. However, the **LREG** solution is valid when $m - n - \theta = 0$, though the analysis of variance is no longer defined. Consequently, when $m - n - \theta = 0$, **LREG** returns the value of SSE instead of the analysis of variance matrix to level 3 of the output, and in the case where $\varepsilon = 0$, the vector returned to level 2 is $\sqrt{(C_{kk})}$ without multiplication by the undefined MSE. It should be understood that SSE is just the square of the output of **LSERR** defined in the LINAG menu and, thus, is in fact the mean-squared value of the error. Consider the following examples:

$$\text{LREG}\left(\begin{bmatrix} (1,2) & (5,6) \\ (2,3) & (9,7) \end{bmatrix}, 1, 0 \right) = \begin{cases} 3: 0 \\ 2: [\ 3 \quad 1\ ] \\ 1: \{\ (-0.5,2.5) \quad (2.5,-1.5)\ \} \end{cases}$$

MISCELLANEOUS FUNCTIONS AND APPROXIMATIONS MENU { FTNS MISC }

FUNCTION	COMMAND	INPUTS	OUTPUTS

LREG([(1,2) (5,6); (2,3) (9,7)] , 1 , 1E-8) = { 3: 1.1E-20; 2: [0.3170 4.4609]; 1: { (-0.5,2.5) (2.5,-1.5) }

LREG([(1,2) (3,2) (5,6); (2,3) (6,5) (9,7)] , 0 , 0) = { 3: 0; 2: [1.5207 0.7500]; 1: { (4.375,-1.125) (-0.625,-0.125) }

LREG([(1,2) (3,2) (5,6); (2,3) (6,5) (9,7)] , 0 , 1E-8) = { 3: 5.33E-20; 2: [0.5909 9.573]; 1: { (4.375,-1.125) (-0.625,-0.125) }

When using the SVD solution option, do not make ε too small. Often ε = 1E-4 is adequate. The maximum number of iterations per column is set at 20. For more information, see the LINAG menu.

F STATISTICS	**FSTAT**(M,V,L)	**LREG** OR **LSSOV** OUTPUT	4:-2: **LREG** OUTPUT 1: UτPF(DFR, DFE,F)

FSTAT computes the F statistic UτPF(DFR , DFE , F).

MISCELLANEOUS FUNCTIONS AND APPROXIMATIONS MENU { FTNS MISC }

FUNCTION	COMMAND	INPUTS	OUTPUTS
T STATISTICS AND F STATISTICS	**TSTAT**(M,V,L)	**LREG** OR **LSSOV** OUTPUT	5:–3: **LREG** OUTPUT 2: UτPF(DFR, DFE,F) 1: [UτPT VECTOR]
Compute F and T statistics (only when ε = 0). See above examples.			
LEAST SQUARES SOLVE	**LSSOV**(FL,VL,D,θ,ε)	FL VL D θ ε	SAME AS LREG

Given the function list FL (symbolic vector SV as defined in the SYMB menu), the variable list VL, and the data matrix D, **LSSOV** provides the least squares solution and analysis of variance like **LREG**, but it is far more general. Parameters θ and ε are the same as in **LREG,** and the output has the same form as that of **LREG**. The below discussion assumes that the reader is *very familiar* with the material discussed above. An understanding of symbolic vectors (SYMB menu) is helpful.

The data matrix D is just like the M matrix input to **LREG**. The first n columns of D are values for the independent variables x_{jk} for j = 1, 2, . . ., m and k = 1, 2, . . ., n. The n + 1 column of D is the corresponding values of the dependent variable y_j for j = 1, 2, . . ., m. Function list FL contains all the non-constant terms of the model whose linear combination will equal the estimate of y. More specifically, if CL is the coefficient list output from **LSSOV**, then:

$$y = \begin{cases} \text{SDOT(FL , CL)} & \theta = 0 \\ \text{SDOT(\{ 1 \} + FL , CL)} & \theta = 1 \end{cases}$$

MISCELLANEOUS FUNCTIONS AND APPROXIMATIONS MENU { FTNS MISC }

FUNCTION	COMMAND	INPUTS	OUTPUTS

where the **SDOT** command performs a symbolic DOT product of two symbolic vectors that are HP 48 list objects and { 1 } + { f_1 f_2 } = { 1 f_1 f_2 }. By applying the appropriate nonlinear transformation to the Y vector, this model can be extended to more general models, as discussed under the **LREG** command. Again we see that the equations depend on the value of θ. The below examples will make this clearer.

Variable list VL is a list of those variable symbols used in FL as independent variables. The size of list VL is exactly equal to n, and the X entries of the D matrix are values for those variables. ***LSSOV** gives a bad-dimension error if the size of VL is not exactly equal to the number of columns of D minus one.*

As a first simple example, let D = M, where M is defined on page 171.

FL = { x_1 x_2 } = VL = { x_1 x_2 }

FL defines the model to be of the form $y = b_1 x_1 + b_2 x_2$ when θ = 0, and $y = b_0 + b_1 x_1 + b_2 x_2$ when θ = 1. Thus, the constant term b_0 is specified by θ and is never included in FL. With these definitions for FL, VL, and D, the output of **LSSOV** is identical to that of **LREG** given on pages 172 and 173. Thus, the capabilities of **LREG** are a subset of the capabilities of **LSSOV**.

Next consider the simple polynomial model $y = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + b_4 x^4$. Then:

FL = { x 'x^2' 'x^3' 'x^4' }, FV = { x },

and we have the result given on the next page.

THERE MUST BE AS MUCH DATA AS THE NUMBER OF COEFFICIENTS TO BE COMPUTED. **LREG** and **LSSOV** do not compute underdetermined cases.

MISCELLANEOUS FUNCTIONS AND APPROXIMATIONS MENU { FTNS MISC }

FUNCTION	COMMAND	INPUTS	OUTPUTS

LSSOV (FL , VL , [(1,1) (-25,2) ; (3,4.1) (-1096,-2727) ; (7,9) (-26793,-67672) ; (2,-5) (-2980,2673) ; (5.1,3) (-5357,2101)] ,1,0) =

```
3: 2.01E-14
     [ 2.204      1.374
2:     0.310      .04783
     2.799E-3 ]
   { (1.9748,2.5984)
     (2.2146,1.2978)
1:   (0.9471,2.9282)
     (4.0079,1.0111)
     (2.9993,2.9996) }
```

The SVD solution with ε = 1E–6 yields SSE = 3.04E–12 and the singular values: 17044.15, 245.9897, 33.5068, 0.38899, 2.06429.

Consider now the more general model

$$g(y) = a + b_1 f_1(x_1, x_2, \ldots, x_n) + \ldots, + b_q f_q(x_1, x_2, \ldots, x_n)$$

where g(y) is some user–chosen function which is invertible and one-to-one. For example, the inverse of **LN** is **EXP**. If you choose to linearize your problem by taking **LN** of both sides of the equation, then the n + 1 column of the D matrix, which we have denoted as the Y vector, would contain the values $LN(y_k)$ for k = 1, 2, . . ., m. The commands **CCMB**, **CSPLT**, and the commands in the VSAG menu make such transformations trivial. Now the above approximation is specified by:

$$FL = \{ \text{'}f_1(x_1, x_2, \ldots, x_n)\text{'} \ \text{'}f_2(x_1, x_2, \ldots, x_n)\text{'} \ \ldots, \ \text{'}f_q(x_1, x_2, \ldots, x_n)\text{'} \},$$

where we observe that there are n independent variables and either q or q + 1 coefficients required, depending on whether θ = 0 or 1. The n independent variables are specified by the list VL = { x_1 x_2 . . ., x_n }.

MISCELLANEOUS FUNCTIONS AND APPROXIMATIONS MENU { FTNS MISC }

FUNCTION	COMMAND	INPUTS	OUTPUTS

Now the condition for a positive error degrees of freedom is $m - q - \theta > 0$. As in **LREG**, the software does compute the solution for $m - q - \theta = 0$, but SSE is returned instead of the analysis of variance matrix, and when $\varepsilon = 0$, the level 2 output is $\sqrt{(C_{kk})}$ for k = 1, 2, . . ., q. The condition for a SVD solution ($\varepsilon > 0$) is $q + \theta \geq 2$. Consider the following example. Given the data matrix:

$$\mathbf{D} = \begin{bmatrix} (0.1,0.1) & (0,0.1) & (-22.1,52.1) \\ (0,0.1) & (0.1,0) & (-50.6,-19.1) \\ (0,0) & (0.2,0.1) & (-23.8,4.1) \\ (0.1,0.2) & (0.2,0) & (-24.7,-8.3) \\ (0.2,0.3) & (0.2,0.3) & (-12.6,13.2) \end{bmatrix}$$

let us model y (the third column) as the sum of $E_4(w)$, $\psi(z)$, C(w), and $J_{.3}(z)$, where w is the first column and z is the second column of D. The functions are

$$E_n(w) = \text{ENOZ}(n,w) = \text{exponential integral} = \int_1^{\infty} \frac{e^{-zt}}{t^n}\, dt \ ,$$

$\psi(z) = \text{PSI}(z) = d[\ \text{LN}(\ \Gamma(z))\ ]/dz$ where $\Gamma(z)$ is the gamma function,

$$C(w) = \text{COFZ}(w) = \text{Fresnel integral} = \int_0^{w} \cos\left(\frac{\pi}{2}\, t^2\right) dt \ ,$$

$J_v(z) = \text{JVOZ}(v,z) =$ Bessel function of first kind.

Define FV = { 'ENOZ(4,w)' 'PSI(z)' 'COFZ(w)' 'JVOZ(0.3,z)' } and VL = { w z }.

Then we have the results shown on the next page.

MISCELLANEOUS FUNCTIONS AND APPROXIMATIONS MENU { FTNS MISC }

FUNCTION	COMMAND	INPUTS	OUTPUTS

LSSOV(FL,VL,D,0,0) =

```
3: [4  7723.22   1930.80      ]
   [1  7.83E-5   7.83E-5       ]
   [5  7723.22   24654044      ]
2: [ 0.1327  1.70E-3  7.61E-2  7.26E-2 ]
1: { (.77731,2.20243)   (4.9936,2.00112)
     (2.01264,2.81358)  (3.02578,1.91889) }
```

LSSOV(FL,VL,D,1,1E-8) =

```
3: 3.79E-17
2: [ 16.64  1.89  0.305  0.103  7.60E-3 ]
     { (0.10717,-.32398)     (.34078,3.1195)
1:     (4.9932,2.00034)      (1.91711,3.22934)
       (3.08706,1.93301) }
```

While the above example is obviously contrived, it does demonstrate the capability of **LSSOV**. A more practical example might be fitting the sums of orthogonal polynomials, such as those in the POLY menu, to your data.

One of the issues associated with least squares estimation is that of parameter reduction. For example, how much is the above model degraded if one of the four functions in the above model is eliminated. While we give no recommendations on how to reduce your model, **LSSOV** computes all the major parameters, including singular values, to help you in that reduction.

Since the size of the solution matrix C^{-1} is either n×n or (n + 1)×(n + 1), depending on whether $\theta = 0$ or $\theta = 1$, and does not depend on the number of data sets m, even large data sets can be processed with **LREG** and **LSSOV**. While the SVD solution is slower, it always converges, and the zero or very small singular values identify redundant model parameters.

MISCELLANEOUS FUNCTIONS AND APPROXIMATIONS MENU { FTNS MISC }

FUNCTION	COMMAND	INPUTS	OUTPUTS

LSSOV is basically a user setup command for **LREG**. Given inputs FL, VL, and D, **LSSOV** computes the M matrix and calls **LREG** for the solution. As a result, pathological cases can be created which result in one of the cases discussed at the bottom of page 172. If you get a bad-argument or a bad-dimension error, then **LREG** does not like its inputs, and you will have to figure out what needs to be changed. M, as created by **LSSOV**, has m rows (same as D), and SIZE(FL) + 1 columns. Related commands are given in the STAT, PROC, and WIND menus.

WEIGHTED LINEAR REGRESSION AND LEAST SQUARES ESTIMATION

While **LREG** and **LSSOV** do not explicitly handle the case of weighted least squares, it is easy to add. Let W be the positive semidefinite diagonal matrix of appropriate dimension. Then the desired solution is to the linear system $X^H W X s = X^H W Y$ for $\theta = 0$, and $Z^H W Z s = Z^H W Y$ for $\theta = 1$. Thus, the **LREG** and **LSSOV** solutions are special cases with W equal to the identity matrix. Denoting by D the square root of the W matrix, which can be input as a vector, square rooted by **VSRT**, and converted to a diagonal matrix S for multiplying by using **D→M**, we then solve the system with inputs (S X) = X_1 and (S Y) = Y_1.

SYMBOLIC DOT PRODUCT	**SDOT(A,B)**	A, B ∈ SM OR SV OR 𝔸	VALUE

SDOT computes symbolic dot products. See the SYMB menu for definitions.

If α1 = { 1 } + { x y } = { 1 x y } and α2 = { 2.5 4 6.5 }, then

SDOT(α1 , α2) = '2.5 + 4x + 6.5y' .

MISCELLANEOUS FUNCTIONS AND APPROXIMATIONS MENU { FTNS MISC }

FUNCTION	COMMAND	INPUTS	OUTPUTS
LIST REVERSE	**LREV**(L)	LIST	LIST
α = { 1 2 3 4 5 } LREV(α) = { 5 4 3 2 1 }			
PERMUTATIONS	**PERMF**(z,r)	$z \in \mathbb{C}$ $r \in \mathbb{C}$	VALUE
PERMF(z,r) = $\Gamma(z + 1)/\Gamma(z + 1 - r)$ **PERM** is special case of **PERMF**.			
$(z)_n$	**POCH**(z,n)	$z \in \mathbb{C}$ $n \in \mathbb{C}$	VALUE
Pochhammer's symbol: $(a)_n = \Gamma(a + n)/\Gamma(n)$ for $n \in \mathbb{I}$. **POCH** is used by the hypergeometric functions for numerical stability.			
MARCUM Q FUNCTION	**MQ**(α,β)	PARAMETERS $\alpha, \beta \geq 0$	VALUE
$\int_{\beta}^{\infty} x \, EXP(-[x^2 + \alpha^2]/2) \, I_0(\alpha x)dx,$ where I_n is the modified Bessel function of the first kind. This computation can be very slow. Consider setting 7 SCI mode for the integration.			

MISCELLANEOUS FUNCTIONS AND APPROXIMATIONS MENU { FTNS MISC }

FUNCTION	COMMAND	INPUTS	OUTPUTS
DO UNTIL NO CHANGE	**MULTI**(P)	P = PROGRAM	RESULT
See page 569 of the HP 48 owner's manual.			
TEST IF REAL	**TIFRE**(M)	M ∈ SM OR SV OR 𝔸	SM OR SV OR 𝔸
TIFRE tests its argument M to see if every element has an imaginary part whose magnitude is less than 1E–8. If every element does, then **TIFRE** returns the real part of M. M may be a standard vector or matrix object. It can also be a list of numbers or a symbolic vector or matrix.			
COUNT UNIQUE	**UNIQE**(V)	V = VECTOR OR LIST	SYMBOLIC MATRIX
UNIQE returns a symbolic matrix. The first row is the unique elements of V, and the second row is the corresponding number of times they occur in V. **UNIQE** uses the **SAME** command to determine uniqueness, so for numbers, use the vector input which forces 1 to be the same as (1, 0). After appropriately rounding V, **UNIQE** can be used to determine the number of unique roots and the multiplicity of those roots, so the process of computing partial fraction expansions, residues for inverse Laplace transforms, and Jordan forms can be automated. Examples follow.			

MISCELLANEOUS FUNCTIONS AND APPROXIMATIONS MENU { FTNS MISC }

FUNCTION	COMMAND	INPUTS	OUTPUTS

UNIQE([1 2 3 1 4 5 2]) = { { 3 1 4 5 2 } { 1 2 1 1 2 } }

SSIZE of the result is { 2 5 }, so there are 5 unique elements of V. Similarly,

UNIQE({ A B C A BC A }) = { { B C BC A } { 1 1 1 3 } }.

SORT RE V_J > = < 0	**SSORT**(V)	[VECTOR]	LIST OF LISTS

SSORT sorts the elements of the root vector V according to whether the real part is greater than, equal to, or less than 0. This is useful in Laplace transform applications. The output is a list containing three lists, which contain the sorted roots.

SSORT([(–1,2) (0,1) (1,2)]) = { { (1,2) } { (0,1) } { (–1,2) } }

SORT ABS V_J > = < 1	**ZSORT**(V)	[VECTOR]	LIST OF LISTS

ZSORT sorts the elements of the root vector V according to whether the absolute value is greater than, equal to, or less than 1. This is useful in z transform applications. The output is a list containing three lists, which contain the sorted roots.

ZSORT ([.5 1 2]) = { { 2 } { 1 } { .5 } }

MISCELLANEOUS FUNCTIONS AND APPROXIMATIONS MENU { FTNS MISC }

FUNCTION	COMMAND	INPUTS	OUTPUTS
SIGN FUNCTION	**SIGNP**(z)	z ∈ ℂ	SIGN
Same as HP 48 **SIGN** command except that SIGNP(0) = 1.			
UP DIRECTORY	**UPDIR**	NONE	PARENT MENU

Allen, A., *Probability, Statistics, and Queueing Theory*, Boston, Academic Press, 1990.

Anderson, T., *An Introduction to Multivariate Statistical Analysis*, New York, Wiley, 1958.

Burden, R. and Faires, J., *Numerical Analysis*, Boston, PWS–KENT, 1989.

Cramer, H., *Mathematical Methods of Statistics*, Princeton, Princeton Univ. Press, 1971.

Draper, N., and Smith, H., *Applied Regression Analysis*, New York, Wiley, 1966.

Edwards, A., *Multiple Regression and the Analysis of Variance and Covariance*, San Francisco, Freeman, 1979.

Marcum, J., "Statistical Theory of Target Detection by Pulsed Radar," *IRE Transactions on Information Theory*, New York, April 1960.

Rao, C., *Linear Statistical Inference and Its Applications*, New York, Wiley, 1965.

Scheffe, H., *The Analysis of Variance*, New York, Wiley, 1959.

18

NUMBER THEORY CALCULATIONS

INTRODUCTION

This chapter presents the 22 number theory commands in the NUMB menu. The NUMB commands provide greatest common divisor, least common multiple, factoring, and prime number search, in addition to computation of Fibonacci, Bernoulli, Euler, Riemann zeta, and Stirling numbers. Bernoulli and Euler polynomials are also provided.

CALCULATIONS

GCD and **LCM** compute the greatest common divisor and the least common multiple. **FACTR** factors natural numbers into primes, and **FMPY** multiplies those factors. **PRIME** searches for prime numbers in a specified range. 999,999,999,999 is the largest natural number which can be represented on the HP 48. Large integer arithmetic with unlimited precision is available for the HP 48. See Appendix A.

NUMBERS AND POLYNOMIALS

FIBON, **BR**, and **ER** compute the Fibonacci, Bernoulli, and Euler numbers, respectively. **BPOLY**, **BCOEF**, **EPOLY**, and **ECOEF** compute the Bernoulli and Euler polynomials and coefficients. **RZETA** and **CZETA** compute the Riemann zeta and complementary zeta functions. **STRL1** and **STRL2** compute Stirling numbers of the first and second kind. The generalized Riemann zeta function is discussed at the end of the menu.

NUMBER THEORY MENU {NUMB}

FUNCTION	COMMAND	INPUTS	OUTPUTS
GREATEST COMMON DIVISOR	**GCD**(a,b)	a, b ∈ **N**	3: GCD 2: t 1: s
GREATEST COMMON DIVISOR of 426 and 732 is 6 with s = 55 and t = –32. **GCD**(a,b) = s a + t b; 6 = 55 × 426 – 32 × 732.			
LCM	**LCM**(a,b)	a, b ∈ **N**	LCM
LEAST COMMON MULTIPLE = **LCM**(a,b) = ab/**GCD**(a,b); **LCM**(426,732) = 51972.			
FACTOR	**FACTR**(n)	n ∈ **N**	LIST
FACTR(123456789) = { 3 3 3607 3803 }.			
MULTIPLY FACTORS	**FMPY**(L)	L = LIST FROM FACTR	2: LIST 1: n
FMPY({ 3 3 3607 3803 } = { 2: { 3 3 3607 3803 } ; 1: 123456789 }.			
PRIME NUMBERS	**PRIME**(n,m)	n, m ∈ **N** 0 < n < m	LIST
PRIME computes all the prime numbers in the interval [n , m]. **PRIME**(451,491) = { 457 461 463 467 479 487 491 }.			

NUMBER THEORY MENU {NUMB}

FUNCTION	COMMAND	INPUTS	OUTPUTS
FIBONACCI NUMBERS	**FIBON**(r)	r ∈ **N** OR 0 r = ORDER	NUMBER

The Fibonacci numbers may be defined by the recurrence formula:

$F_r = F_{r-1} + F_{r-2}$ where $F_0 = 0$ and $F_1 = 1$.

The closed form expression for them is

$$F_r = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^r - \left(\frac{1-\sqrt{5}}{2}\right)^r\right] = \frac{1}{\sqrt{5}}[\alpha^r - \beta^r],$$

where α and β are called the golden mean numbers.

FIBON(0) = 0 FIBON(1) = 1 FIBON(2) = 1 FIBON(3) = 2 FIBON(4) = 3

BERNOULLI NUMBERS B_r	**BR**(r)	r ∈ **N** OR 0 r = ORDER	NUMBER

BR(0) = 1 BR(1) = −1/2 BR(2) = 1/6 BR(3) = 0 BR(4) = −1/30

BERNOULLI POLYNOMIAL	**BPOLY**(v,r)	r ∈ **N** OR 0 r = ORDER	POLYNOMIAL IN v
BERNOULLI COEFFICIENTS	**BCOEF**(r)	r ∈ **N** OR 0 r = ORDER	COEFFICIENT LIST

$B_4(x)$ = BPOLY(x,4) = $x^4 - 2x^3 + x^2 - 1/30$ BCOEF(4) = { −1/30 0 1 −2 1 }

NUMBER THEORY MENU {NUMB}

FUNCTION	COMMAND	INPUTS	OUTPUTS
EULER NUMBERS E_r	**ER**(r)	r ∈ **N** OR 0 r = ORDER	NUMBER
ER(0) = 1 ER(1) = 0 ER(2) = –1 ER(3) = 0 ER(4) = 5			
EULER POLYNOMIAL	**EPOLY**(v,r)	r ∈ **N** OR 0 r = ORDER	POLYNOMIAL IN v
EULER COEFFICIENTS	**ECOEF**(r)	r ∈ **N** OR 0 r = ORDER	COEFFICIENT LIST
$E_4(x)$ = EPOLY(x, 4) = $x^4 - 2x^3 + x$ ECOEF(4) = { 0 1 0 –2 1 }			
$\zeta(n)$ RIEMANN ZETA FUNCTION	**RZETA**(n)	n ∈ **N** n ≥ 2	NUMBER
COMP RIEMANN ZETA FUNCTION	**CZETA**(n)	n ∈ **N** n ≥ 1	NUMBER
$\zeta(n) = \sum_{k=1}^{\infty} k^{-n}$ $\beta(n) = \sum_{k=0}^{\infty} (-1)^k(2k+1)^{-n}$ RZETA(4) = 1.08232323371 CZETA(2) = .915965594177 = Catalan constant			
$S_n^{(m)}$ STIRLING NUMBER OF FIRST KIND	**STRL1**(m,n)	m, n ∈ **N** 0 ≤ m ≤ n	NUMBER
$(-1)^{n-m} S_n^{(m)}$ is the number of permutations of n symbols which have exactly m cycles.			

NUMBER THEORY MENU {NUMB}

FUNCTION	COMMAND	INPUTS	OUTPUTS

$$x(x-1)\ \ldots\ (x-n+1) = \sum_{m=0}^{n} S_n^{(m)} x^m = \frac{x!}{(x-n)!} = \mathbf{PERM(x,n)} = \mathbf{PERMF(x,n)}$$

$$S_{n+1}^{(m)} = S_n^{(m-1)} - n\, S_n^{(m)},$$

where $S_0^{(0)} = S_n^{(n)} = 1$, $S_n^{(0)} = 0$ for $n > 0$, and $S_n^{(1)} = (-1)^{n-1}(n-1)!$.

STRL1(2,4) = 11

For large arguments, the accuracy of **STRL1** degrades. For full 12–digit integer accuracy (the limit on the HP 48) with large arguments, see the Stirling transform command **STRLT** in the WIND menu of Chapter 28.

$\mathbb{S}_n^{(m)}$ STIRLING NUMBER OF SECOND KIND	**STRL2(m,n)**	m, n ∈ **N** 0 ≤ m ≤ n	NUMBER

$\mathbb{S}_n^{(m)}$ is the number of ways of partitioning a set of n elements into m non-empty subsets.

$$\mathbb{S}_{n+1}^{(m)} = m\, \mathbb{S}_n^{(m)} + \mathbb{S}_n^{(m-1)},$$

where $\mathbb{S}_0^{(0)} = \mathbb{S}_n^{(1)} = \mathbb{S}_n^{(n)} = 1$, $\mathbb{S}_n^{(0)} = 0$ for $n > 0$, and $\mathbb{S}_n^{(n-1)} = \text{COMB}(n,2)$.

STRL2(2,4) = 7

HP COLCT	**COLCT**	EXPRESSION	EXPRESSION
HP EXPAN	**EXPAN**	EXPRESSION	EXPRESSION

NUMBER THEORY MENU {NUMB}

FUNCTION	COMMAND	INPUTS	OUTPUTS
EXPAND COLLECT COMPLETELY	**EXCO(E)**	EXPRESSION	EXPRESSION
VALUE OF i	**I**	NONE	(0,1)
VALUE OF π	**P**	NONE	3.14159265359
UP DIRECTORY	**UPDIR**	NONE	PARENT MENU

SUMS OF RECIPROCAL POWERS

The Riemann zeta function $\zeta(n)$ and the complementary Riemann zeta function $\beta(n)$ are examples of reciprocal powers that are tabulated in AMS 55. Two others are

$$\eta(n) = \sum_{k=1}^{\infty} (-1)^{k-1} k^{-n} = (1 - 2^{1-n})\zeta(n), \qquad \lambda(n) = \sum_{k=0}^{\infty} (2k+1)^{-n} = (1 - 2^{-n})\zeta(n),$$

which can be evaluated using the **RZETA** command.

GENERALIZED RIEMANN ZETA FUNCTION

The generalized Riemann zeta functions $\zeta(s,\alpha)$ and $\zeta(s) = \zeta(s,1)$ are defined by

$$\zeta(s, \alpha) = \sum_{n=0}^{\infty} (\alpha + n)^{-s} \qquad \text{RE } s > 1.$$

$\zeta(s, \alpha)$ is a special case of Lerch's transcendent $\Phi(z, s, \alpha)$ discussed on page 112. $\zeta(s, \alpha)$ is available for RE $\alpha > 0$ and $s \in \mathbb{C}$. See Appendix A.

For definitions see Chapters 23 and 24 of Abramowitz and Stegun, *Handbook of Mathematical Functions*, AMS 55, 1964.

19

SYMBOLIC ALGEBRA AND CALCULUS

INTRODUCTION

This chapter presents the 42 symbolic algebra and calculus commands in the ALGB menu. The ALGB commands provide Taylor series, polynomial arithmetic, polynomial calculus, three complex coefficient root solvers, conversion between symbolic polynomials and polynomial lists, and algebraic simplification. Since most of the commonly used functions are special cases of hypergeometric functions, four commands provide symbolic series expansions of these functions and all their special cases. Related commands are given in the MISC menu for polynomial operations and approximations. The HP 48 solve application can be used with all the MATHLIB functions provided they are treated as real functions of real variables. See page 74 for an example, solving for the zeros of $j_n(x)$, and Chapter 17 of the HP 48 owner's manual.

TAYLOR SERIES EXPANSIONS

TALR1 and **TALR2** provide one– and two–dimensional Taylor series expansion about any specified point. **COEFL** reduces symbolic polynomials to polynomial lists that are lists of the polynomial coefficients. **COEFL** also provides Maclaurin series expansions of algebraic expressions. Both arithmetic and calculus operations are much faster using polynomial lists than performing the same operation with the symbolic polynomial itself. Given a polynomial list, **FEVAL** provides fast polynomial evaluation, while **PEQN** and **XEQN** provide pretty symbolic equations. See also the advanced series approximation material on page 450.

POLYNOMIAL ARITHMETIC AND CALCULUS

PADD, PSUB, PMPY, PDVD, and **PLDVD** for arbitrary degree polynomials provide the basic arithmetic operations for polynomials. **PDERV** and **PINTG** provide differentiation and integration for polynomials. **RDERV** computes the derivative of the quotient of two polynomials.

COMPLEX COEFFICIENT ROOT SOLVER

Given the polynomial list P, **PROOT** creates a companion (Frobenius) matrix whose eigenvalues are the roots of P. It then solves for those eigenvalues. **AROOT** uses Laguerre's method for finding all of the roots. **LROOT,** which seeks a single root given the user's guess, is provided for difficult problems. **DEFLT** automatically determines how many times a root is repeated and deflates the polynomial list accordingly. **CLIST** computes the polynomial from its roots.

HYPERGEOMETRIC FUNCTIONS

H1F1, DH1F1, F2F1, and **DH2F1** provide the hypergeometric functions and their derivatives in symbolic form. The majority of the transcendental functions are special cases of these functions, so this is a fast way of obtaining a power series expansion of these functions. For example, M(a,a,z) = EXP(z) for all values of a.

$$M(a,a,z) = H1F1(a,a,z) = '\Sigma(\eta=0,N,POCH(a,\eta)/POCH(a,\eta)\times z^{\eta}/\eta!)'$$

Canceling the identical numerator and denominator POCH(a,η) factor with the equation editor, we have the Taylor series expansion of EXP(z):

$$EXP(z) = '\Sigma(\eta=0,N,z^{\eta}/\eta!)'.$$

UTILITIES

EVLΣ provides an unconditional multiple ↑**MATCH** command for replacing any or all arguments of algebraic expressions.

SUMΣ provides for easy symbolic evaluation of hypergeometric and other symbolic power series. It explicitly evaluates the terms, including symbolically expanding Pochhammer's symbol $(z)_n$, which is evaluated by POCH(z,n).

TECHNIQUES FOR SOLVING BIG PROBLEMS ON THE HP 48

Ten powerful and useful algebraic manipulation and substitution programs are given. Then techniques for algebraically manipulating hundreds of terms in big algebra and calculus problems on the HP 48 are demonstrated.

ROOT SOLVING AND POLYNOMIAL LIST CONVENTIONS

MATHLIB follows several conventions with respect to polynomial lists and vectors containing the roots of polynomials. The polynomial $P(z) = a_0 + a_1z + a_2z^2 + a_3z^3 + . . ., + a_nz^n$ is represented by the list object { a_0 a_1 a_2 a_3 ... a_n }, which is an order corresponding to increasing powers of z. This order is chosen to maximize the speed of polynomial evaluation with commands like **FEVAL**. The same polynomial can be represented within a scaling factor by its root vector [r_1 r_2 r_3 ... r_n], where the polynomial $P(z) = C (z - r_1) (z - r_2) (z - r_3) . . ., (z - r_n)$, where C is the complex scale factor. Given the polynomial list, **PROOT** and **AROOT** return the corresponding root vector. Given the root vector, **CLIST** returns the corresponding polynomial list scaled so that $a_n = 1$. If $a_0 \neq 0$ and you wish to scale the list so that $a_0 = 1$, command **PSCAL** found in the FILTR menu will rescale the polynomial list.

In order for these two representations of polynomials to work properly, we need the symbol [ ] to represent no roots, but this symbol (an empty vector) is not valid syntax on the HP 48. Consequently, MATHLIB uses { } to represent it. The root vector corresponding to the polynomial lists { } and { 1 } is [ ], which we represent by { }. The polynomial list corresponding to the root vector { } is { 1 }. The command **LZDEL**, which deletes the unnecessary trailing zeros in a polynomial list, and the commands **CLIST**, **PROOT**, and **AROOT** all understand these conventions. The analog and digital filter commands in the FILTR and WIND menus also understand and use these conventions.

CONVERGENCE FAILED is displayed when the root finder cannot converge the roots according to the input arguments. When this happens, push **ATTN** and the software will return what roots it has found. The display will show where the convergence failed so you can count the number of true roots which were converged.

SYMBOLIC HIGHER TRANSCENDENTAL FUNCTIONS

Example programs are given for extending the numerical MATHLIB functions to symbolic ones with defined derivatives. See also Appendix A.

ALGEBRA AND CALCULUS MENU { ALGB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
TAYLOR SERIES	**TALR1**(F,L)	F AND L	POLYNOMIAL

F is any function and L is the list { INDEPENDENT VARIABLE VALUE TO EXPAND ABOUT NUMBER OF TERMS }.
For example: F = 'EXP(−(X−1)^2)' and L = { X 1 8 } expands the function of X about X_0=1 and computes the first 8 terms =
'1 −(−1+X)^2 +.5(−1+X)^4 −1.66666666667(−1+X)^6 +4.16666666667E−2(−1+X)^8'.

TAYLOR SERIES	**TALR2**(F,L1,L2)	F, L1, L2	POLYNOMIAL

F is any function of two variables and L1, L2 are the corresponding lists. For example:
'EXP(X+iY)' L1 = { X 0 4 } L2 = { Y 2π 3 }
yields a two−dimensional Taylor series expansion of EXP(X+iY).

POLYNOMIAL ADD	**PADD**(L1,L2)	COEFFICIENT LISTS L1 & L2	COEFFICIENTS L1 + L2

Adds polynomials of arbitrary degree.
Each coefficient list represents a polynomial. Given the list and an independent variable x, FEVAL below will compute the polynomial. Given a polynomial in x, COEFL below will compute a list of coefficients which represent the polynomial.

{ a_0 a_1 a_2 ... a_{n-1} } represents $\Sigma(k=o,n-1,a_k\ x\text{^}k)$.

CAREFULLY NOTE THE ORDER OF THE COEFFICIENTS

For example: {5 0 3 7 5 9} + {4 7 4 9 7 0 8 2} = { 9 7 7 16 12 9 8 2 }.

ALGEBRA AND CALCULUS MENU { ALGB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
POLYNOMIAL SUBTRACT	**PSUB**(L1,L2)	COEFFICIENT LISTS L1 & L2	COEFFICIENTS L1 – L2

Subtracts polynomials of arbitrary degree.
For example: { 0 2 5 8 1 } – { 5 9 7 2 1 8 0 } = { –5 –7 –2 6 0 –8 }.
Observe that in polynomial arithmetic and calculus, trailing zeros are meaningless and are deleted by the MATHLIB, but leading (left-most) are very significant. Thus:

{ 2 5 8 1 } – { 5 9 7 2 1 8 0 } = { –3 –4 1 –1 –1 –8 }.

POLYNOMIAL MULTIPLY	**PMPY**(L1,L2)	COEFFICIENT LISTS L1 & L2	COEFFICIENTS L1 × L2

Multiplies polynomials of arbitrary degree.

For example: { 2 5 8 1 } × { 5 9 7 2 1 8 0 } = { 10 43 99 116 77 44 50 65 8 }.

POLYNOMIAL DIVIDE	**PDVD**(L1,L2)	COEFFICIENT LISTS L1 & L2	3: QUOTIENT 2: REMAINDER 1: DIVISOR

Divides polynomials of arbitrary degree. See also **PLDVD** in this menu.

L1 / L2 = { 5 9 7 2 1 8 0 } / { 2 5 8 1 0 } = { 466 –63 8 0 }

with remainder { –927 –2195 –3422 }.

Thus, { 2 5 8 1 } × { 466 –63 8 0 } + { –927 –2195 –3422 } = { 5 9 7 2 1 8 }.

ALGEBRA AND CALCULUS MENU { ALGB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

SYMBOLIC POLYNOMIAL ARITHMETIC

The above four commands also work for symbolic arguments. **PDVD** checks the first element of the numerator to see if it is a number. If it is not, then **PDVD** uses **EXCO** to clean up the algebra.

PADD({ A B C D } , { E F }) = { 'A+E' 'B+F' C D }

PSUB({ A B C D } , { E F }) = { 'A–E' 'B–F' C D }

PMPY({ A B C D } , { E F })

= { 'A×E' 'A×F+B×E' 'B×F+C×E' 'C×F+D×E' 'D×F' } = L

PDVD(L , { E F }) =
3: { A B C D }
2: { 0 }
1: { E F }

POLYNOMIAL EVALUATION	**FEVAL**(L,V)	L AND V	POLYNOMIAL OR NUMBER

Given coefficient list L = { a_0 a_1 a_2 ... a_{n-1} } and independent variable V, **FEVAL** computes $\Sigma(k=0,n-1,a_k\ V\text{^}k)$. CAREFULLY NOTE THE ORDER OF THE COEFFICIENTS. While **FEVAL** can be used for symbolic values of V, **XEQN** below is preferred. For example:

FEVAL({ 4 5 2 7 } , z) = '4 + (5 + (2 + 7z) z) z' = $4 + 5z + 2z^2 + 7z^3$.

ALGEBRA AND CALCULUS MENU { ALGB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
COEFFICIENTS	**COEFL**(F,V,N)	F, V, N	COEFFICIENTS
Reduces a function F in variable V to a coefficient list of length N + 1, which is its Maclaurin series expansion. See **FEVAL** for coefficient order. If F = '$4z^4-2z^2+5z-6$', then: COEFL(F,z,4) = { −6 5 −2 0 4 }, COEFL(F,z,6) = { −6 5 −2 0 4 0 0 }, COEFL('EXP(−t)',t,4) = { 1 −1 .5 −1/6 1/24 }.			
LIST REVERSE	**LREV**(L)	L = LIST	LIST
LREV({ 1 2 3 4 5 }) = { 5 4 3 2 1 }			
HP COLCT	**COLCT**	EXPRESSION	EXPRESSION
HP EXPAN	**EXPAN**	EXPRESSION	EXPRESSION
EXPAND COLLECT COMPLETELY	**EXCO**(E)	EXPRESSION	EXPRESSION
HP ↑MATCH	**↑MATCH**	EXPRESSION	EXPRESSION
HP ↓MATCH	**↓MATCH**	EXPRESSION	EXPRESSION
HP | (where)	|	EXPRESSION	EXPRESSION
For the above commands, see Chapter 22 as well as page 570 of the HP 48 owner's manual.			

ALGEBRA AND CALCULUS MENU { ALGB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
EVALUATE COEFFICIENT	**COEFE**(F,V,α)	FUNCTION F VARIABLES V α	F {V α} |
COEFE('SIN(ω×t)×EXP(−α×t)' , t , T) = 'SIN(ω×T)×EXP(−α×T)'			
HP APPLY	**APPLY**	2: symbol list 1: name	name(symbols)
HP QUOTE	**QUOTE**	symbol or object	symbol or object
See the HP 48 Programmer's Reference Manual.			
SYMBOLIC M(a, b, z)	**H1F1**(a,b,z)	a, b, z	EXPRESSION
SYMB NTH DERIVATIVE OF M(a, b, z)	**DH1F1**(a,b,z,n)	a, b, z n ∈ **N**	EXPRESSION
SYMBOLIC F(a, b, c, z)	**H2F1**(a,b,c,z)	a, b, c, z	EXPRESSION
SYMB NTH DERIVATIVE OF F(a, b, c, z)	**DH2F1**(a,b,c,z,n)	a, b, c, z n ∈ **N**	EXPRESSION

H1F1 is the symbolic form of the confluent hypergeometric function M(a, b, z).
H2F1 is the symbolic form of the Gaussian hypergeometric function F(a, b, c, z).
The symbol N is reserved as the number of terms in these four commands, so do not use it as an input variable.

ALGEBRA AND CALCULUS MENU { ALGB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

Numerous functions are special cases of these two functions. For example, e^z, sin(z), cos(z), $J_v(z)$, $I_v(z)$, $\gamma(a,x)$, erf(z), and most of the common polynomials are all special cases, to list a few. Consequently, the above four commands provide series approximations for all these functions and their derivatives, as well as symbolic equations for them. For more details see AMS 55.

The commands **EVLΣ** and **SUMΣ** may help you manipulate these symbolic expressions. **EVLΣ** provides a means of replacing any or all the symbols in an algebraic expression. **SUMΣ** will explicitly evaluate the terms of the above four commands provided the second argument of **POCH** is a number, which it is for **H1F1** and **H2F1**. For **DH1F1** and **DH2F1**, it is when n is a number, n ∈ **N**.

SYMBOLIC EVALUATE SUM	**EVLΣ**(S,L)	EXPRESSION S LIST L	EXPRESSION

EVLΣ replaces symbols in algebraic S. List L of even size lists in pairs the current symbol and its replacement symbol or number. **EVLΣ** provides a special capability like a multiple, unconditional ↑**MATCH** command.

EVLΣ('Σ(n=1,N,POCH(a,n)×z^n/n!)' , {a A z Z}) = 'Σ(n=1,N,POCH(A,n)×Z^n/n!'

COMPUTE SUM	**SUMΣ**(S,N)	SUM S	N ∈ **N**	SUM

Replaces upper symbolic limit of the sum in algebraic S with number N and evaluates the sum (LOWER LIMIT MUST BE A NUMBER).

NOTHING MAY BE LEFT OF THE Σ.

SUMΣ('Σ(n=1,N, z^n / n!') , 4) = '.5×z^2 + .1666667×z^3 + 4.1666667E−2×z^4 + z'

ALGEBRA AND CALCULUS MENU { ALGB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
EVALUATE COMPLETELY	**EVALC**(F)	F = FUNCTION OR EXPRESSION	F
Uses **MULTI** to cause repeated **EVAL** commands until no change.			
HP ISOLATE VARIABLE	**ISOL**	2: $SYMB_1$ 1: GLOBAL	$SYMB_2$
HP QUADRATIC SOLUTION	**QUAD**	2: $SYMB_1$ 1: GLOBAL	$SYMB_2$
See Chapter 22 of the HP48 owner's manual.			
POLYNOMIAL ROOT	**PROOT**(L,ε,MAX)	3: COEF LIST 2: ε > 0 1: MAX TRYS	[VECTOR]

PROOT takes the complex coefficient list L (see **FEVAL** for definition and order) and creates a Frobenius matrix M. **EIGEN** is used to compute the eigenvalues of M, which are the complex roots of L. For example let $f(x) = x^5 - 0.2x^4 + 7x^3 + x^2 - 3.5x + 2 = 0$. Then list L = { 2 –3.5 1 7 –0.2 1 }. With ε = 1E–10, the roots are

(.140377913, ±2.7314490896), (–.908563257, 0), (.413903715, ±.3506447075).

MAX is the maximum number of iterations (say 20) **EIGEN** is allowed to converge each eigenvalue. If the solution fails at some eigenvalue, the diagonal matrix is still returned, but only the last values of the vector (the ones which converged) are actual roots.

ALGEBRA AND CALCULUS MENU { ALGB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

CONVERGENCE FAILED is displayed when the root finders cannot converge the roots according to the input arguments. When this happens, push **ATTN**, and what roots it has found are returned. The display shows where the convergence failed, so you can count the number of true roots.

POLYNOMIAL ROOT	**AROOT**(L,v,ε, MAX)	L v ε MAX	[VECTOR]

AROOT uses Laguerre's method for solving for all the roots of the polynomial defined by the complex coefficient list L. Value v is the initial guess for all of the roots, and ε sets the convergence criterion. MAX (say 100) is the maximum iterations allowed to converge each root. After converging each root, **DEFLT** is used to remove that root from polynomial L.

The most difficult polynomial root solutions occur when there are repeated roots. Consider the example: '$x^{12} + 99x^{11} - 377x^{10} - 26395x^9 + 149080x^8 + 1703048x^7 - 15440048x^6 + 8684864x^5 + 302914240x^4 - 1377763200x^3 + 2718976000x^2 - 2620320000x + 1008000000$' = $(x-2)^5$ (x–5) (x+6) (x–7) (x+10) (x–10) (x+15) (x+100). With v = 0, ε = 1E–8, and MAX = 100, we only get two roots: (1.98607080, 0) and (1.99562237, –1.32705219E–2). Dropping ε = 1E–4 yields the roots: (1.98607080, 0) (1.99563474, –1.32669871E–2) (1.99563459, .0132665015) (2.01132960, 8.27241365E–3) (2.01133027, –8.27192808E–3) (5, 0) (–6, 0) (7, 0) (–10, 0) (10, 0) (–15, 0) (–100, 0).

The average of the first 5 roots is (2, 0) to 11 digits. **LROOT** can be used to verify that (2,0) is the true root, and **DEFLT** can be used to remove it from P, giving the reduced polynomial
{ (–31500000, 0) (3135000, 0) (1619500, 0) (–108650, 0) (–23945, 0) (673, 0) (109,0) 1 }, which is easily solved since all the roots are unique.

ALGEBRA AND CALCULUS MENU { ALGB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
POLYNOMIAL ROOT	**LROOT**(L,v,ε, MAX)	L v ε MAX	ROOT

LROOT uses Laguerre's method for solving for a single root of the polynomial defined by the list L. Value v is the initial guess, and ε sets the convergence criterion. For example, let the polynomial be $z^3 + (-5,0)\ z^2 + (19,8)\ z + (-7,-24)$. The output of **COEFL** is { (–7,–24) (19,8) (–5,0) 1 }. With v = 8 and ε = 1E–10, the output of **LROOT** is (1.00000124,2.00000221). The actual roots are [(1,2) (1,2) (3,–4)].

DEFLATE POLYNOMIAL	**DEFLT**(P,r,ε)	3: POLYNOMIAL 2: ROOT r 1: ε > 0	3: REMAINDER 2: ROOT r 1: NUMBER

DEFLT successively divides polynomial P by divisor D = { –r 1 }, where r is a root of P as long as the absolute value of the remainder is less than ε. P and D are polynomial lists. For example, let P = { (–7,–24) (19,8) (–5,0) 1 }, r = (1,2), and ε = 1E–8. Then D = { (–1,–2) 1 }, and **DEFLT** returns the remainder { (–3,4) 1 }, the root r, and the multiplicity of the root 2. If instead r = (–1,2), then P is returned to level 3, the root to level 2, and 0 to level 1, since (–1,2) is not a root of P. Root r is considered a true root of P if, after division by D, the absolute value of the remainder is less than or equal to ε.

POLYNOMIAL DERIVATIVE	**PDERV**(L)	POLYNOMIAL LIST	POLYNOMIAL LIST

PDERV computes the derivative of the polynomial represented by the list L. The derivative of polynomial { (–7,–24) (19,8) (–5,0) 1 } is { (19,8) (–10,0) (3,0) }, corresponding to the polynomial (3,0) z^2 + (–10,0) z + (19,8).

ALGEBRA AND CALCULUS MENU { ALGB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
POLYNOMIAL DERIVATIVE OF QUOTIENT	**RDERV**(P,Q)	2: NUMERATOR 1: Q = DENOMINATOR	2: NUMERATOR 1: DENOMINATOR

RDERV computes the derivative of the quotient of polynomials P and Q. For example, if P = { 3 –4 1 } and Q = { –48 44 –12 1 }, then the derivative of the quotient P/Q has the numerator { 60 –24 –13 8 –1 } and the denominator { 2304 –4224 3088 –1152 232 –24 1 }.

The commands **PDERV**, **RDERV**, and **PINTG** do allow polynomial lists with symbolic elements.

POLYNOMIAL INTEGRATION	**PINTG**(P,C)	2: POLY LIST P 1: NUMBER OR SYMBOL	INTEGRATED POLYNOMIAL LIST

Given the polynomial list P and integration constant C, **PINTG** computes the polynomial list of the integral of the polynomial corresponding to list L.

PINTG({ (19,8) (–10,0) (3,0) } , 1) = { 1 (19,8) (–5,0) (1,0) }.

POLYNOMIAL EQUATION	**PEQN**(L,S)	2: POLY LIST 1: STRING S	PRETTY EQUATION

PEQN({ (–7,–24) (19,8) (–5,0) 1 } , "T") = 'T3 + (–5,0) T2 + (19,8) T1 + (–7,–24) T0'.

ALGEBRA AND CALCULUS MENU { ALGB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
POLYNOMIAL EQUATION	**XEQN**(L,V)	2: POLY LIST 1: VARIABLE V	PRETTY EQUATION

XEQN({ (–7,–24) (19,8) (–5,0) 1 } , 'z') = z^3 + (–5,0) z^2 + (19,8) z + (–7,–24).

POLYNOMIAL LONG DIVISION	**PLDVD**(L1,L2,n)	COEFFICIENT LISTS L1 & L2 $n \in \mathbb{N}$	3: QUOTIENT 2: REMAINDER 1: DIVISOR

PLDVD is very different than **PDVD**. **PLDVD** begins by computing the first n terms of the reciprocal of the polynomial represented by L2 with padding. Then it multiplies that result by the polynomial represented by L1. The remainder is then computed as that polynomial, when added to the product of the quotient and the divisor (L2), is equal to the polynomial represented by L1.

PLDVD({ 1 } , { 1 A } , 4) =

3: { 1 '–A' 'A^2' '–A^3' }
2: { 0 0 0 0 'A^4' }
1: { 1 A }

As a result, **PLDVD** gives a very different result for the **PDVD** first example.

PLDVD({ 5 9 7 2 1 8 0 } , { 2 5 8 1 0 } , 5) =

3: { 2.5 –1.75 –2.125
12.0625 –20.28125 }
2: { 0 0 0 0 0 15.03125
150.1875 20.28125 }
1: { 2 5 8 1 0 }

While I do not wish you to make a habit of breaking conventions, **PADD**, **PSUB**, **PMPY**, **PDVD**, **PLDVD**, and **LZDEL** do accept [VECTOR] inputs where the nth value in the vector corresponds with the nth value of the normal list input.

ALGEBRA AND CALCULUS MENU { ALGB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

Observe that like all polynomial commands, the trailing zeros in the input lists are ignored. **PLDVD** essentially reduces a rational polynomial to the first n terms of its power series expansion, the quotient, plus higher order terms contained in the remainder. For the same number of terms, **PLDVD** and **RATAP** are inverses of each other. **RATAP** computes rational approximations of power series, and **PLDVD** computes power series approximations of rational functions.

PLDVD({1 -0.6 0.15 -1/60} , {1 0.4 0.05} , 6) =

```
3: { 1  -1  1/2  -1/6
     1/24  -1/120}
2: { 0 0 0 0 0 0
     .00125  1/2400 }
1: { 1  .4  .05 }
```

The magnitude of the remainder is a measure of the goodness of the approximation. Applications of both these commands are discussed in Chapter 27.

COEFFICIENT LIST	**CLIST**(V)	[VECTOR]	LIST

Given vector V containing the roots of some polynomial, **CLIST** computes the corresponding polynomial list.

LIST ROUND	**SRND**(L,N)	L N	ROUNDED LIST

Given list L of numbers and integer N, **SRND** applies ≪ N RND ≫ to each element of L. See page 148 of the HP 48 owner's manual.

ALGEBRA AND CALCULUS MENU { ALGB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
LIST ZERO DELETE	**LZDEL**(L)	LIST	LIST

Removes the contiguous zeros at the right of list L. For example:

LZDEL({ 1 2 3 0 4 0 0 0 }) = { 1 2 3 0 4 },

LZDEL({ A B C D 0 0 }) = { A B C D }.

TEST IF REAL	**TIFRE**(M)	M ∈ SM OR SV OR 𝔸	SM OR SV OR 𝔸

TIFRE tests its argument M to see if every element has an imaginary part whose magnitude is less than 1E–8. If every element does, then **TIFRE** returns the real part of M. M may be a standard vector or matrix object. It can also be a list of numbers or a symbolic vector or matrix.

UP DIRECTORY	**UPDIR**	NONE	PARENT MENU

SPECIAL ALGEBRA IDENTITY AND EXPANSION PROGRAMS

The following ten programs provide various identities and algebraic expansion capabilities. MTHS supplies missing HP 48 identities. PARE, PARED, and POLYE provide parenthesis expansion. TRIGE expands various trigonometric forms. TOEXP expresses trigonometric and hyperbolic functions in terms of exponentials. CEXPE expands complex exponential forms. CNJE evaluates complex conjugates. SREIM is a symbolic equivalent of **C→R** and isolates the symbolic real and imaginary parts of equations. CREIM is like **R→C**, but works with algebraic expressions. These programs may be refined to meet your specific needs. These ten programs are available. See Appendix A.

ALGEBRA AND CALCULUS MENU { ALGB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

SYMBOLIC ALGEBRAIC ANALYTIC FUNCTION IDENTITIES

The program MTHS provides identities for LN(EXP(&)), LOG(ALOG(&)), ASIN(SIN(&)), ACOS(COS(&)), ATAN(TAN(&)), ASINH(SINH(&)), ACOSH(COSH(&)), ATANH(TANH(&)), √(SQ(&)), INV(1/&), EXP(LNP1(&)), EXPM(LN(&)), √(–&) = √(&)i where & denotes any argument. The identities LN(e) = 1 and LN(INV(e)) = –1 are also provided.

```
MTHS: ≪ → F ≪ F " MATH IDENTITIES" 3 DISP { 'LN(EXP(&X))' &X }
↑MATCH DROP { 'LOG(ALOG(&X))' &X } ↑MATCH DROP { 'ASIN(SIN(&X))'
&X } ↑MATCH DROP { 'ACOS(COS(&X))' &X } ↑MATCH DROP
{ 'ATAN(TAN(&X))' &X } ↑MATCH DROP { 'ASINH(SINH(&X))' &X } ↑MATCH
DROP { 'ACOSH(COSH(&X))' &X } ↑MATCH DROP { 'ATANH(TANH(&X))' &X }
↑MATCH DROP { '√SQ(&X)' &X } ↑MATCH DROP { 'INV(1/&X)' &X }
↑MATCH DROP { 'EXP(LNP1(&X))' '&X+1' } ↑MATCH DROP { 'EXPM(LN(&X))'
'&X–1' } ↑MATCH DROP { '√–&X' '√&X×i' } ↑MATCH DROP { 'LN(e)' 1 }
↑MATCH DROP { 'LN(INV(e))' –1 } ↑MATCH DROP ≫ ≫
```

PARENTHESIS EXPANSION UTILITIES

PARE expands forms like a(b±c) and (b±c)a and PARED forms like (b±c)/d.

```
PARE: ≪ → f ≪ "EXPAND A(B±C)" 3 DISP 0 → F ≪ f DO { '&C×(&A+&B)'
'&C×&A+&C×&B' } ↑MATCH 'F' STO { '&C×(&A–&B)' '&C×&A–&C×&B' }
↑MATCH 'F' STO+ F UNTIL 0 == END DO { '(&A+&B)×&C' '&A×&C+&B×&C'
} ↑MATCH 'F' STO { '(&A–&B)×&C' '&A×&C–&B×&C' } ↑MATCH 'F' STO+ F
UNTIL 0 == END ≫ ≫ ≫

PARED: ≪ → f ≪ "EXPAND (B±C)/D" 3 DISP 0 → F ≪ DO { '(&A+&B)/&C'
'&A/&C+&B/&C' } ↑MATCH 'F' STO { '(&A–&B)/&C' '&A/&C–&B/&C' } ↑MATCH
'F' STO+ F UNTIL 0 == END ≫ ≫ ≫
```

ALGEBRA AND CALCULUS MENU { ALGB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

PARENTHESIS EXPANSION UTILITY

Program POLYE expands the binomial forms (a±b)^n and c(a±b)^n for n ∈ **N**. Then it expands the parentheses of the resulting expression.

```
POLYE: « → F « "EXPAND (A±B)^N" 3 DISP F { '&C×(&A+&B)^&N'
  'Σ(K=0,&N,&C×&A^K×&N!/K!×&B^(&N–K)/(&N–K)!' } ↑MATCH DROP
  { '&C×(&A–&B)^&N' 'Σ(K=0,&N,&C×&A^K×&N!/K!×(–&B)^(&N–K)/(&N–K)!' }
↑MATCH DROP{ '(&A+&B)^&N' 'Σ(K=0,&N,&A^K×&N!/K!×&B^(&N–K)/(&N–K)!' }
↑MATCH DROP{ '(&A–&B)^&N' 'Σ(K=0,&N,&A^K×&N!/K!×(–&B)^(&N–K)/(&N–K)!' }
  ↑MATCH DROP EVAL { '&C×(&A+&B)' '&C×&A+&C×&B' } ↑MATCH DROP
    { '&C×(&A–&B)' '&C×&A–&C×&B' } ↑MATCH DROP
  { '(–&A)^&N' '(–1)^&N×&A^&N' } ↑MATCH DROP EVAL { '(&A+&B)×&C'
  '&A×&C+&B×&C' } ↑MATCH DROP { '(&A–&B)×&C' '&A×&C–&B×&C' }
    ↑MATCH DROP { '(–&A)^&N' '(–1)^&N×&A^&N' } ↑MATCH DROP
  { '(&A×&B)^&N' '&A^&N×&B^&N' ↑MATCH DROP EVAL COLCT » »
```

FAST ALGEBRAIC EXPANSION AND COLLECTION

The HP 48 **EXPAN** command can be very slow in expanding certain expressions. For example, expanding $z^{20}(3z^{43}+5z^7)$ with **EXPAN** will expand z^7 to z×z×z×z×z×z×z as part of the effort. Similarly, it will expand z^{20} and z^{43}. Only after fully expanding the expression can you use **COLCT** to yield an expression without parentheses. As more symbols get put on the stack as part of the expansion, the slower **EXPAN** works, and the memory manager may become completely occupied with cleaning up garbage. If the objective is simply to eliminate the parentheses, there is a great deal of wasted effort both in the expansion and the eventual collection.

However, the program « PARE COLCT » will expand $z^{20}(3z^{43}+5z^7)$ to $z^{20}\times(3\times z^{43})+z^{20}\times(5\times z^7)$ and then collect as $5\times z^{27}+3\times z^{63}$ in seconds.

ALGEBRA AND CALCULUS MENU { ALGB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

TRIGONOMETRIC EXPANSIONS

Program TRIGE provides expansions for SIN(&A±&B), COS(&A±&B), TAN(&A±&B), SIN(&A)^n, and COS(&A)^n for n = 2, 3, 4 where &A and &B denote any argument.

```
TRIGE: ≪ → F ≪ "TRIG IDENT EXPAND" 3 DISP F { 'SIN(&A+&B)'
'SIN(&A)×COS(&B)+COS(&A)×SIN(&B)' } ↑MATCH DROP { 'SIN(&A-&B)'
'SIN(&A)×COS(&B)-COS(&A)×SIN(&B)' } ↑MATCH DROP { 'COS(&A+&B)'
'COS(&A)×COS(&B)-SIN(&A)×SIN(&B)' } ↑MATCH DROP { 'COS(&A-&B)'
'COS(&A)×COS(&B)+SIN(&A)×SIN(&B)' } ↑MATCH DROP { 'TAN(&A+&B)'
'(TAN(&A)+TAN(&B))/(1-TAN(&A)×TAN(&B))' } ↑MATCH DROP { 'TAN(&A-&B)'
'(TAN(&A)-TAN(&B))/(1+TAN(&A)×TAN(&B))' } ↑MATCH DROP { 'COS(&A)^2'
'.5+.5×COS(2×&A)' } ↑MATCH DROP { 'COS(&A)^3'
'.75×COS(&A)+.25×COS(3×&A)' } ↑MATCH DROP { 'COS(&A)^4'
'.375+.5×COS(2×&A)+.125×C0S(4×&A)' } ↑MATCH DROP { 'SIN(&A)^2'
'.5-.5×COS(2×&A)' } ↑MATCH DROP { 'SIN(&A)^3'
'.75×SIN(&A)-.25×SIN(3×&A)' } ↑MATCH DROP { 'SIN(&A)^4'
'.375-.5×COS(2×&A)+.125×C0S(4×&A)' } ↑MATCH DROP ≫ ≫
```

SYMBOLIC CONVERSION TO EXPONENTIAL FORM

TOEXP converts SIN, COS, SINH, and COSH to exponential form. This is useful in many integration problems.

```
TOEXP: ≪ → F ≪ " TO EXP FORM" 3 DISP F
{ 'SIN(&)' 'EXP(&×i)/(0,2)-EXP(-&×i)/(0,2)' } ↑MATCH DROP
{ 'COS(&)' 'EXP(&×i)/(0,2)+EXP(-&×i)/(0,2)' } ↑MATCH DROP
{ 'SINH(&)' '.5×EXP(&)-.5×EXP(-&)' } ↑MATCH DROP
{ 'COSH(&)' '.5×EXP(&)+.5×EXP(-&)' } ↑MATCH DROP ≫ ≫
```

ALGEBRA AND CALCULUS MENU { ALGB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

COMPLEX EXPONENTIAL EXPAND

Program CEXPE expands complex exponentials of the form $e^{\pm iz}$, $e^{a\pm ib}$, $e^{a\pm bi}$, e^{ai}, i^a, and a^i.

```
« → F « "EXPAND EXP(iz)" 3 DISP F { 'EXP(&A+i×&B)'
'EXP(&A)×COS(&B)+EXP(&A)×SIN(&B)×i' } ↑MATCH DROP { 'EXP(&A–i×&B)'
'EXP(&A)×COS(&B)–EXP(&A)×SIN(&B)×i' } ↑MATCH DROP { 'EXP(&A+&B×i)'
'EXP(&A)×COS(&B)+EXP(&A)×SIN(&B)×i' } ↑MATCH DROP { 'EXP(&A–&B×i)'
'EXP(&A)×COS(&B)–EXP(&A)×SIN(&B)×i' } ↑MATCH DROP { 'EXP(i×&)'
'COS(&)+SIN(&)×i' } ↑MATCH DROP { 'EXP(–i×&)' 'COS(&)–SIN(&)×i' }
↑MATCH DROP { 'EXP(&×i)' 'COS(&)+SIN(&)×i' } ↑MATCH DROP { 'i^&A'
'COS(π×&A/2)+SIN(π×&A/2)×i' } ↑MATCH DROP
{ '&A^i' 'COS(LN(&A))–SIN(LN(&A))×i' } ↑MATCH DROP { 'LN(EXP(&A))' &A }
↑MATCH DROP EVAL COLCT » »
```

COMPLEX CONJUGATE EVALUATE

Program CNJE attempts to evaluate the CONJ operations in the input expression.

```
CNJE: « → E « E CEXPE "EVAL CONJ" 3 DISP 0 → F « DO
{ 'CONJ(&A+&B)' 'CONJ(&A)+CONJ(&B)' } ↓MATCH 'F' STO { 'CONJ(&A–&B)'
'CONJ(&A)–CONJ(&B)' } ↓MATCH 'F' STO+ F UNTIL 0 == END EVAL
{ 'CONJ(i×&)' '–&×i' } ↑MATCH DROP { 'CONJ(&×i)' '–&×i' } ↑MATCH DROP
{ 'CONJ(–(&×i))' '&×i' } ↓MATCH DROP EVAL { 'CONJ(&)' & } ↑MATCH
DROP EVAL COLCT » » »
```

ALGEBRA AND CALCULUS MENU { ALGB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

SYMBOLIC C→R TO SEPARATE THE REAL AND IMAGINARY PARTS

Program SREIM trys to separate the real and imaginary parts of a symbolic expression. Like the **ISOL** command, it may fail, especially with complicated expressions.

SREIM: « → F « F CNJE CEXPE POLYE PARE PARED COLCT DUP { '&×i' '&×0' } ↓MATCH DROP { 'i' 0 } ↑MATCH DROP { '–i' 0 } ↑MATCH DROP EVAL SWAP '–i' × PARE COLCT { '&×i' '&×0' } ↓MATCH DROP { 'i' 0 } ↑MATCH DROP { '–i' 0 } ↑MATCH DROP EVAL » »

SYMBOLIC R→C TO COMBINE THE REAL AND IMAGINARY PARTS

CREIM: « → R C « R C 'i' × + » »

EXAMPLE – ALGEBRA TECHNIQUES FOR BIG PROBLEMS

It is told that an aerospace engineer was stuck with the hydrodynamic flow problem of evaluating the following particularly difficult integral:

$$\int [\, A + Bt + Ct^3 + \sin(t) \,]^4 \, dt.$$

After two long weeks, he was able to expand the integrand into its 256 terms three times. However, since he obtained three different answers and could not determine which was correct, he gave up. We present our solution on the next few pages. This is a big problem, and you will need 25 K to 30 K bytes of free memory to run it. If you run short, modify EX4 on the next page to purge IEQN and IEQNI.

ALGEBRA AND CALCULUS MENU { ALGB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

The following program will give you the 256 terms in about 35 minutes.

EX1: « '(A + Bz + Cz^3 + SIN(z))^4' POLYE POLYE POLYE PARE »

However, the above 256 terms contain terms with SIN(z)^n for n = 2, 3, and 4. An investment in TRIGE will eliminate all the powers of the SIN function, resulting in even more terms. A second investment in PARE will again eliminate the parentheses, so we are ready to integrate the hundreds of terms. EX2 takes about 31 minutes (times may vary depending on a variety of conditions).

EX2: « TRIGE PARE 'IEQN' STO »,

which stores the result in the variable IEQN. Then the program

EX3: « 0 z IEQN z ∫ EVAL 'IEQNI' STO »

will perform the integration in 6.7 minutes. The output consists of a part which is completely integrated and a part which is not. The following program will isolate these two parts.

EX4: « IEQNI { '∫(0,z,&,z)' 0 } ↓MATCH DROP 'IEQN1' STO IEQNI
{ '&A+∫(0,z,&B,z)' &B } ↓MATCH DROP 'IEQN2' STO »

The completed part of the integral is now stored in IEQN1, and the remaining integrand in IEQN2. Since the HP 48 does not have megabytes of memory for massive integral tables, we will have to give it a little help. From standard integral tables, we have

$$\int z^n \sin az\, dz = -\sum_{k=0}^{n} k! \binom{n}{k} \frac{z^{n-k}}{a^{k+1}} \cos(az + k\pi/2),$$

ALGEBRA AND CALCULUS MENU { ALGB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

$$\int z^n \cos az\, dz = \sum_{k=0}^{n} k! \binom{n}{k} \frac{z^{n-k}}{a^{k+1}} \sin(az + k\pi/2).$$

Observe that the integral depends on the values of n and a in addition to z. In particular, n = 1, 2, . . ., 9 and a = 1, 2, . . ., 4. Using commands from the SYMB menu discussed in Chapter 25, the following program constructs an 8 × 10 symbolic function matrix of coefficients. The first four rows are coefficients of sin kz for k = 1, 2, 3, and 4. The second four rows are coefficients of cos kz for k = 1, 2, 3, and 4. In addition, the columns are coefficients of z^n for n = 0, 1, . . ., 9. By isolating these coefficients, the above integral formulas can be directly applied.

Program EX2 made the integrand linear in sin kz and cos kz for k = 1, 2, 3, and 4. By substituting a variable S1, S2, . . ., S8 for these, we can isolate the coefficients by computing the gradient with respect to each. The result is a column vector of coefficients, each being a polynomial in z. By now expanding these coefficients into a Maclaurin series, we isolate the coefficients of z^n for n = 0, 1, . . ., 9. The result is a symbolic column vector of polynomial coefficient lists, which is also an 8 × 10 symbolic matrix as defined in Chapter 25. Program EX5 computes the matrix in about 22 minutes.

```
EX5: « 1 4 FOR K 'SIN(K×z)' EVAL "S" K + OBJ→ NEXT 1 4 FOR K
'COS(K×z)' EVAL "S" K 4 + + OBJ→ NEXT { 8 2 } →SOB → L « IEQN2
1 8 FOR K " MATCH " K + 3 DISP L K SERW ↑MATCH DROP NEXT "
COMPUTE GRAD" 3 DISP L 2 SECOL SGRD SEXCO
« z 9 COEFL » L1F1 'IEQN3' STO » »
```

ALGEBRA AND CALCULUS MENU { ALGB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

Now substitution of the above two integral formulas is simple, since the row number corresponds to the value of a and the column number corresponds to value of n. EX6 provides the integral formulas as a function of row number and column number. It is called by EX7 below.

```
EX6: « → r c « -2 SF IF r 4 ≤ THEN
'-Σ(k=0,c-1,k!×COMB(c-1,k)×z^(c-1-k)/r^(k+1)×COS(r×z+k×π/2))'
ELSE r 4 - 'r' STO
'Σ(k=0,c-1,k!×COMB(c-1,k)×z^(c-1-k)/r^(k+1)×SIN(r×z+k×π/2))'
END EVAL -2 CF » »
```

Program EX7 completes the integration and stores the final result in IEQN1.

```
EX7: « 1 8 FOR R 1 10 FOR C IEQN3 R C 2 →LIST SGET DUP IF 0
SAME NOT THEN R C EX6 × END 'IEQN1' STO+ NEXT NEXT » »
```

This entire example requires a little over 1.6 hours to compute, and the final equation for the integral consists of hundreds of terms. Use the | (where) command to evaluate it.

SYMBOLIC ALGEBRA AND CALCULUS WITH LIBRARY FUNCTIONS

The HP 48 commands **APPLY** and **QUOTE** were not designed to work with library objects. Consequently, the program « A GAMMA » will evaluate to an error. The easiest way to include the MATHLIB higher transcendental functions in your symbolic algebra and calculus computations is to create a number of symbolic functions which call the library commands when all the arguments are real or complex numbers. A number of example programs are given below with derivative definitions. Over 200 are available, as discussed, in Appendix A. They supply full symbolic capability, including derivatives, for the functions in Chapters 3 through 15.

ALGEBRA AND CALCULUS MENU { ALGB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

SYMBOLIC FUNCTION AND DERIVATIVE DEFINITION EXAMPLES

Many HP 48 calculators have a bug in the **APPLY** command so it does not work properly. The first rule is never use the **APPLY** command to create a symbolic function definition, because for some arguments, it will not differentiate properly.

For example, the program « → z « 'APPLY(YYS,z)' EVAL » » stored in variable YYS defines the symbolic function YYS. If YYS is evaluated with argument Z, then the resulting function 'YYS(Z)' properly differentiates to 'derYYS(Z,1)'. However, if YYS is evaluated with argument '4×Z', the resulting function 'YYS(4×Z)' differentiates to 'derYYS(4×Z,0)' instead of the correct function 'derYYS(4×Z,4)'.

The below program sequences combined with special command **EVLΣ** create symbolic functions which will differentiate properly.

GAMMAS: « → z « IF z TYPE 1 ≤ THEN z GAMMA ELSE 'GAMMAS(α)' α z 2 →LIST EVLΣ END » »

PSIS: « → z « IF z TYPE 1 ≤ THEN z PSI ELSE 'PSIS(α)' α z 2 →LIST EVLΣ END » »

DNPSIS: « → n z « IF n TYPE 1 ≤ z TYPE 1 ≤ AND THEN n z DNPSI ELSE 'DNPSIS(α,β)' α n β z 4 →LIST EVLΣ END » »

der■!: « → A dA « IF dA 0 SAME THEN 0 ELSE 'GAMMAS(A+1)×PSIS(A+1)×dA' EVAL COLCT END » »

derGAMMAS: « → A dA « IF dA 0 SAME THEN 0 ELSE 'GAMMAS(A)×PSIS(A)×dA' EVAL COLCT END » »

derPSIS: « → A dA « IF dA 0 SAME THEN 0 ELSE 'DNPSIS(1,A)×dA' EVAL COLCT END » »

ALGEBRA AND CALCULUS MENU { ALGB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

FIRST DERIVATIVE EXAMPLE

Compute the derivative of $4^n/n!$.

DEX1: « '4^n/n!' n ∂ »

= '1.38629436112×4^n/n! – 4^n×(PSIS(1+n)×GAMMAS(1+n))/n!^2'.

Since GAMMAS(1+n) = n!, this reduces to $4^n/n![1.38629436112-\psi(1+n)]$.

FIRST DERIVATIVE EXAMPLE

Compute the derivative of $4^n/\Gamma(n+1)$.

DEX2: « '4^n/GAMMAS(n+1)' n ∂ »

= '–(PSIS(1+n)/GAMMAS(1+n)×4^n) + 1.38629436112/GAMMAS(1+n)×4^n'.

SECOND DERIVATIVE EXAMPLE

Evaluate the second derivative of $z^3 \cos(2z)/\Gamma(z)$ at z = 1.3.

DEX3: « 'z^3×COS(2×z)/GAMMAS(z)' z ∂ EXCO z ∂ EXCO DUP z 1.3 2 →LIST | »

The result is –10.8782061908.

20

COMPLEX LINEAR ALGEBRA

INTRODUCTION

This chapter presents the 66 complex linear algebra commands in the LINAG menu. The LINAG commands include linear solutions, least squares solutions, singular value decomposition (SVD), tests for special matrices, Gaussian decomposition of matrices, Householder reflections and decompositions, Givens rotations, bidiagonal and tridiagonal decompositions, upper Hessenberg decomposition, Schur decompositions for the Hermitian and general cases, Cholesky factorization, as well as characteristic polynomial computation, eigenvalues, and eigenvectors. Utilities provided include matrix reordering based on pivot vectors, normal and Hermitian transposition, determinants, various matrix norms, matrix trace and minor operations.

Numerous related commands are given in the MATR, VECTR, VSAG, MSAG, SYMB, and PROC menus. Commands for manipulating parts of matrices and vectors are given in the MATR and VECTR menus. Matrix and vector scalar algebra commands are given in the MSAG and VSAG menus. Symbolic array (matrix and vector function) operations are given with the calculus and differential equation commands in the SYMB menu. Toeplitz solution commands are given in the PROC menu.

DEFINITIONS

Many of the commands available in the LINAG menu are advanced topics in linear algebra. The following definitions are given to make the command explanations clear. Let matrix $M \in \mathbb{C}$ have m rows and n columns. Then M is of order $m \times n$.

VECTOR: A vector V is an ordered collection of numbers or functions where the ordering is one-dimensional and the elements are denoted by V_j for j = 1, 2, . . ., N where N is the dimension or SIZE of the vector. This menu is limited to vectors containing real or complex numbers only. The HP 48 supports three kinds of vectors: The basic vector is denoted by [V_1 V_2 ... V_N], the column vector is denoted by [[V_1] [V_2] ... [V_N]], and the row vector is denoted by [[V_1 V_2 ... V_N]]. The column and row vectors are special cases of matrices and share all the matrix commands.

MATRIX: A matrix is an ordered collection of numbers or functions where the ordering is two-dimensional and the elements are denoted by M_{jk} where j = 1, 2, . . ., m and k = 1, 2, . . ., n. M_{j1} is a column vector and M_{1k} is a row vector. This menu is limited to matrices containing real or complex numbers only. The HP 48 denotes matrices by [[M_{11} ... M_{1n}] [M_{21} ... M_{2n}] ... [M_{m1} ... M_{mn}]].

ADDITION AND SUBTRACTION: Matrix addition and subtraction are defined as simply the sum or difference between corresponding elements. A + B implies $A_{jk} + B_{jk}$ for all j and k.

INNER OR DOT PRODUCT: The DOT product is defined for vectors and equals the sum of the products of pairs of corresponding vector elements. $A \bullet B = \Sigma(j=1,N,A_j \times B_j)$. The square root of A • A is the Frobenius (Euclidean) norm of vector A. This is computed by the HP command **ABS**. When A and B are complex, then the dot product is A • CONJ(B), where CONJ computes the complex conjugate of B.

MATRIX PRODUCT: The matrix product A × B is the computation of the m × n inner products of the rows of A and the columns of B. Thus, if C = A × B, then $C_{jk} = \Sigma(\ell=1,m,A_{j\ell} \times B_{\ell k})$.

DETERMINANT: The determinant of a square matrix is a special permuted product of the elements; it is provided by the HP command **DET**.

RANK: The rank of matrix M is the dimension of the range of M. If M has rank r, then: M contains exactly r linearly independent rows and r linearly independent columns, there is an r × r submatrix of M which has a nonzero determinant, and $r \leq \min\{m, n\}$. A matrix of rank $r = \min\{m, n\}$ is said to have or be of full rank.

CONSISTENT: If there is at least one solution, then a linear system is consistent. The linear system Mx = b is consistent if and only if rank [Mb] = rank M, where [Mb] is the augmented matrix with additional column b. The system $x_1 + 2x_2 = 1$ and $x_1 + 2x_2 = 2$ is obviously inconsistent and has no solution. If square matrix M has full rank, then the linear system is consistent for all b.

NONSINGULAR: A matrix is nonsingular if m = n and it has full rank. If M is nonsingular, then: inverse M^{-1} exists and $M\ M^{-1} = I$, the identity matrix; Mx = b is consistent for all b; and Mx = b is a FULLY DETERMINED linear system with solution $x = M^{-1}b$. Similarly, the linear system xM = b is fully determined if M is nonsingular and $x = bM^{-1}$.

UNDERDETERMINED: Underdetermined systems are ones which have more than one feasible solution. The linear system Mx = b is underdetermined if m < n since there are fewer equations than unknowns. If M is square but not of full rank, then the system is also underdetermined. The usual approach is to choose the solution for which Mx – b = 0 and the norm of x is minimized. Then $x = M^H(MM^H)^{-1}b$, where the superscript H denotes Hermitian transpose.

OVERDETERMINED: Overdetermined systems are ones which have more equations than unknowns. The linear system Mx = b is overdetermined if m > n and M has full rank. The system is thus inconsistent. The usual approach is to choose the solution for which the norm of Mx – b is a minimum that is a least squares solution. Then $x = (M^HM)^{-1}M^Hb$. The inverses used in the underdetermined and overdetermined cases are Moore–Penrose pseudoinverses, which generally work when M has full rank. When M is rank–deficient, then either MM^H or M^HM may be singular, and the Moore–Penrose technique fails with a divide–by–zero error.

SINGULAR VALUE DECOMPOSITION LEAST SQUARES: All of the above linear system solution cases are special cases of the singular value decomposition (SVD) approach to linear solutions. The SVD least squares solution commands, while having longer execution time, will always converge to a minimum norm solution, which is exact in the fully determined case. The nonzero singular values of a matrix are invariant under conjugation, transposition, and the addition of zero element rows and columns to make the matrix larger.

TRANSPOSE: The transpose of matrix M, denoted by superscript T, swaps the row and column indices. If $C = M^T$ then $C_{jk} = M_{kj}$. The Hermitian transpose, denoted by superscript H, also conjugates the elements. $C = M^H$ implies $C_{jk} = CONJ(M_{kj})$.

SYMMETRIC: A matrix $M \in \mathbf{C}$ is symmetric if $M^T = M$.

HERMITIAN: A matrix $M \in \mathbf{C}$ is Hermitian if $M^H = M$.

ORTHOGONAL: Two vectors are orthogonal if their dot product is zero. A matrix $M \in \mathbf{R}$ is orthogonal if each pair of column vectors is orthogonal. Thus, if M is orthogonal, then $MM^T = M^TM = I$.

UNITARY: A matrix $M \in \mathbb{C}$ is unitary if $MM^H = M^HM = I$.

NORMAL: A matrix $M \in \mathbb{C}$ is normal if $MM^H = M^HM$.

TRACE: The trace of a square matrix is the sum of its diagonal elements.

MINOR: The determinant of a square submatrix. If the submatrix is a principal submatrix, then the minor is a principal minor. The signed minor $[(-1)^{i+j}$ **DET** M] is called a cofactor.

EIGENVALUES AND EIGENVECTORS: For square matrix $M \in \mathbb{C}$, if there exists a linear solution to $Mx - \lambda x = 0$ for vector $x \neq 0$ and scalar λ, the value of λ is an eigenvalue (characteristic value) of M, and x is an eigenvector (characteristic vector) of M. More rigorously, x is a right eigenvector of matrix M.

SCHUR VECTORS: The column vectors of the unitary matrix associated with the Schur decomposition. Only when M is normal are the Schur vectors also the eigenvectors of M.

SINGULAR VALUES AND SINGULAR VECTORS: For rectangular matrix $M \in \mathbb{C}$, the singular value decomposition (SVD) of $M = U\ D\ V^H$, where U and V are unitary and D is real and diagonal. The values of D are the singular values of M. Also, $D = U^H M\ V$. The columns of U are the left singular vectors, and the columns of V are the right singular vectors of matrix M. The square of the nonzero singular values of M equals the nonzero eigenvalues of $M^H M$ and those of $M\ M^H$.

CHARACTERISTIC POLYNOMIAL: The characteristic polynomial of square matrix M is the equation **DET**$\{\lambda I - M\} = 0$. The roots of this polynomial are eigenvalues of M.

CONDITION NUMBER: Ratio of the largest and the smallest singular values.

LINEAR AND LEAST SQUARES SOLUTIONS

Linear and least squares solutions are given in pairs for the unknown vector on the right $Mx = b$ which is normally the case, as well as the unknown vector on the left, $xM = b$. **LSOVL** and **LSOVR** handle the fully determined cases, **USOVL** and **USOVR** handle the underdetermined cases, and **OSOVL** and **OSOVR** handle the overdetermined cases. The last four commands compute least squares solutions using the Moore–Penrose pseudoinverse technique. Commands for least squares data fitting are in the MISC menu.

SINGULAR VALUE DECOMPOSITION (SVD)

When the matrix MM^H or M^HM does not have full rank, **LSVDL** and **LSVDR** are available for least squares solutions. The matrix of singular values is also returned. The diagonal elements are the singular values, and the off–diagonal elements are a measure of the decomposition accuracy. These commands use **SVD**, which performs a full singular value decomposition of M. **B←M** will remove the singular values of the matrix and place them in a vector, while commands **SVDMI** and **SVDDI** are available for computing the inverse of the SVD matrix. The unitary matrices associated with the SVD are also returned if the flag requesting them is set.

LU AND LDLT DECOMPOSITION

The most common type of matrix decomposition uses Gaussian reduction. **GLUD** provides the LU decomposition of M into the product of a lower triangular and an upper triangular matrix. Row pivoting is used to improve the accuracy of the decomposition. When M is nonsingular, this LU decomposition can be continued, resulting in M decomposed into the product of a lower triangular, a diagonal, and an upper triangular matrix (transposed lower triangular matrix). This command is performed by **LDLTD**. The commands **GAUSS** and **AGAUS** provide access to the internal decomposition programs used for the Gaussian elimination.

SCHUR DECOMPOSITION AND THE EIGEN PROBLEM

The Schur decomposition is the standard means by which the dense eigen problem is solved. There are two cases which must be considered. When matrix M is normal, then the Schur decomposition $M = QDQ^H$ provides the eigenvalues as the elements of the diagonal matrix D and the eigenvectors as the columns of the unitary matrix Q. **SCRSD** performs this decomposition by first using **HTRDD** to reduce M to a tridiagonal matrix and second using **EIGNS** to iteratively converge the off–diagonal elements to zero. Double implicit Wilkinson shifting is employed to speed the convergence.

The general eigen problem is solved similarly with the command **SCHRD**. **UHESD** is used to decompose M into upper Hessenberg form. Then **EIGEN** iteratively converges the elements below the diagonal to zero. The resulting Schur decomposition thus takes the form $M = QTQ^H$, where the eigenvalues of M are the diagonal elements of the upper triangular matrix T, and the associated Schur vectors are the column vectors of Q. **EIGEN** employs double implicit complex shifting to speed convergence.

Since computing the eigenvectors in the general case is quite tricky, I suggest that you study the techniques in the literature. A program for one technique, the inverse power method, is given at the end of this chapter.

Commands **WSQR** and **GSQR** provide user access to the internal eigen iteration software. They perform a single iteration on any part of a matrix, and loops can be built around them. **WSQR** is used with Hermitian matrices, and **GSQR** is used with arbitrary matrices.

When only the eigenvalues are required, **CPOLY** may be used to compute the coefficients of the characteristic polynomial associated with a matrix M. **AROOT** may then be used to solve for the roots whose values are the eigenvalues of M. However, since the **CPOLY** algorithm computes various powers of matrix M, this approach has poor accuracy when the matrix has a large range of values.

CHOLESKY DECOMPOSITION

CHOLD provides Cholesky factorization of a Hermitian positive definite matrix M = GG^H into the product of a matrix G with its Hermitian transpose. When M is real, then G is its square root.

COMPUTATION OF RANK AND QR DECOMPOSITION

HRQR uses Householder reflections to compute the rank of matrix M and the QR decomposition M = QR where Q is unitary and R is upper triangular. Column pivoting is used to preserve accuracy. The program returns matrix R, the rank, the pivot vector, and a list containing the reduction history. This list can be input to the command **HBDU** to construct the unitary matrix Q.

BIDIAGONAL DECOMPOSITION

HBDD computes a bidiagonal decomposition of the form M = UBV^H, where U and V are unitary and bidiagonal B has zero elements except for those on the diagonal and the first superdiagonal. **HBDD** returns matrix B and two lists containing the reduction history. The commands **HBDU** and **HBDV** convert these lists into the U and V matrices.

HBDDR also performs the bidiagonal decomposition $M = UBV^H$, but in addition rotates all the values of B onto the positive real axis. It can be proven that this rotation of the bidiagonal elements with unitary matrices is unique. The U and V unitary matrices are also returned on the stack.

HOUSEHOLDER REFLECTIONS

Unitary decompositions are the ones of choice for eigen and SVD analyses. Householder reflections are used to introduce zeros into rows and columns, while Givens (Jacobi) rotations are used to zero individual matrix elements. **HTRDD** uses symmetric Householder reflections to decompose a Hermitian matrix $M = VTV^H$, where V is unitary and T is tridiagonal with all elements equal to zero except those on and just above or below the diagonal. **UHESD** uses Householder reflections to decompose an arbitrary matrix $M = VHV^H$ to upper Hessenberg form, where V is unitary and H is like upper triangular form, but it also has nonzero elements just below the diagonal.

Access to the internal Householder reflection software is provided. **HOUSE** and **VHOUS** construct the Householder reduction vector, **RHOUS** and **CHOUS** apply it to the rows and columns of the matrix being reduced without spending the computer time to actually construct the associated unitary reflection matrix, and **PHOUS** constructs the unitary reflection matrix given the Householder vector.

GIVENS AND JACOBI ROTATIONS

Access to the internal Givens rotation software is provided. **GIVEN** and **GROT** construct in compact 2 × 2 matrix form the elements of the complex Givens rotation matrix. Commands **RROT** and **CROT** then expand the compact rotation matrix and apply it to the specified rows and columns.

CHARACTERISTIC POLYNOMIAL

The characteristic polynomial of a matrix can be constructed several ways. **CPOLY** constructs it directly from the traces of the powers of matrix M. This is fast but can be inaccurate. Alteratively, **SCRSD** and **SCHRD** can be used to solve for the eigenvalues of M, and then **CLIST** will construct the polynomial list. Given the polynomial list, **XEQN** constructs the algebraic symbolic polynomial.

MATRIX NORMS

RABS and **CABS** provide the Frobenius (Euclidean) norm for each row and column of a matrix. A vector is returned. HP commands **RNRM** and **CNRM** return the maximum value of the vector elements, and the commands **RABS2** and **CABS2** return the square of the values returned by **RABS** and **CABS** in a vector. **RABS2** and **CABS2** are in the MSAG menu.

RANK AND CONDITION NUMBER

HRQR is the fastest way of determining the rank of a matrix in MATHLIB. It uses column pivoting and generally will be sufficient. The most reliable method uses the singular value decomposition command **SVD** and is thus slower. The condition number may be computed from the output of **SVD** as the ratio of the largest singular value and the smallest singular value.

POWER METHOD FOR COMPUTING EIGENVALUES AND EIGENVECTORS

The Schur decomposition approach is generally more accurate than the power method, but requires computation of all the eigenvalues. The power method is useful when only a few of the largest or a few of the smallest eigenvalues are required. Programs are given at the end of Chapter 26 for computing the largest and smallest eigenvalues of Hermitian Toeplitz matrices along with their associated eigenvectors. These programs can be easily generalized to the general complex matrix case. By the use of deflation, as many eigenvalues and eigenvectors as required can be computed.

A program using the inverse power method for computing the eigenvectors of matrices in the general complex case is given at the end of this chapter. It uses **SCHRD** with flag F = 2 to compute the upper Hessenberg decomposition, its associated unitary matrix, and the eigenvalues. Then it iterates to compute each eigenvector.

TESTING EIGENVALUES AND EIGENVECTORS

The commands **λ?** and **λV?** provide tests of whether a given number or vector is an eigenvalue or eigenvector of a matrix. They may also be used to determine the accuracy of convergence. Command **EVS?** tests whether two eigenvectors are the same except for normalization. These commands are given in the MATR menu discussed in Chapter 21.

LIMITATIONS OF THE SOFTWARE

The matrix commands in this menu are intended for matrices of at least size 2 × 2. Arrays of dimension 1 × n or m × 1 are vectors. Given a random real 40 × 40 matrix and a dimension 40 random real column vector, **LSOVR** will solve the system in under 4 minutes. Given a random real 10 × 10 matrix with ε = 1E–8 and F = 0, **SVD** will compute the singular values in under 23 minutes. Given the same matrix with ε = 1E–8 and F = 0, **SCHRD** will compute the eigenvalues in 35 minutes. While **CPOLY** combined with **AROOT** for ε = 1E–6 will compute the eigenvalues in a little over 3 minutes, they may be inaccurate for some matrices. Given the same 10 × 10 matrix with ε = 1E–8, program EVSOV will compute all the eigenvalues and eigenvectors in about 53 minutes. The program « 0 1 1 S MRDN », where S equals { 40 40 } or { 10 10 }, will compute the random matrix for you. Smaller matrices are faster; complex matrices are slower.

CONVERGENCE OF SOLUTIONS

The convergence of eigenvalues and singular values can require some experimentation. If ε is too small, convergence may never occur due to numerical errors. Matrices with repeated eigenvalues are very difficult to converge and ε = 1E–4 may be required. If there is still no convergence, consider spoiling the matrix by adding a small random matrix to it to slightly shift the eigenvalues. Random matrix commands are given in the MSAG menu. **CONVERGENCE FAILED** is displayed when eigen and SVD computations fail to converge according to the user's input arguments.

There is always a small probability that at some point in the decomposition, a pivot element may become exactly zero. If you are this unlucky, add a small number to your matrix with **MADD** or add a small random matrix to your matrix. The HP 48 has a numerical range of over ±1E±499. Adding a small number to your matrix will not change the final result, but it will avoid pathological numerical problems with pivot elements perfectly equaling zero.

Internal to MATHLIB decompositions, a zero mean, standard deviation less than 1E–25, uniformly distributed "random" matrix is added to the input matrix. The seed is fixed for repeatability, and it will have no effect on results, except for pathological cases such as decomposing an all–zero matrix and observing that the algorithm pivoted the all–zero data.

LINEAR ALGEBRA MENU {LINAG}

FUNCTION	COMMAND	INPUTS	OUTPUTS
LINEAR SOLVE	**LSOVL**(B,A)	B ∈ℂ A ∈ℂ	SOLUTION
LINEAR SOLVE	**LSOVR**(B,A)	B ∈ℂ A ∈ℂ	SOLUTION

LSOVL provides the XA = B solution and **LSOVR** provides the AX = B solution in fully determined cases. They use **RSD** once to improve the accuracy of the solution. The **RSD** command is discussed on page 361 of the HP 48 owner's manual.

LINEAR SOLVE	**USOVL**(B,A)	B ∈ℂ A ∈ℂ	SOLUTION
LINEAR SOLVE	**USOVR**(B,A)	B ∈ℂ A ∈ℂ	SOLUTION

USOVL provides XA = B solution and **USOVR** provides AX = B solution in underdetermined cases. They use the Moore–Penrose technique and **RSD** once to improve the accuracy of the solution.

LINEAR SOLVE	**OSOVL**(B,A)	B ∈ℂ A ∈ℂ	SOLUTION
LINEAR SOLVE	**OSOVR**(B,A)	B ∈ℂ A ∈ℂ	SOLUTION

OSOVL provides XA = B solution and **OSOVR** provides AX = B solution in overdetermined cases. They use the Moore–Penrose technique and **RSD** once to improve the accuracy of the solution.

NOTES ON DIMENSIONALITY

Let matrix A have m rows and n columns. Then A has order m×n. The left solution is XA = B and the right solution is AX = B. Let X have order $R_x \times C_x$ and B have order $R_b \times C_b$. Note that B is either a row vector, a column vector, or a matrix of correct dimension. B ≠ [VECTOR]. Then we have the dimensionality relationships given on the next page. The right solution is the common one.

LINEAR ALGEBRA MENU {LINAG}

FUNCTION	COMMAND	INPUTS	OUTPUTS

FULLY DETERMINED LEFT: $C_x = C_b$ and A has order n×n with rank A = n
FULLY DETERMINED RIGHT: $R_x = R_b$ and A has order m×m with rank A = m

UNDERDETERMINED LEFT: $C_x > C_b$ and A has order m×n with m>n and rank A=n
UNDERDETERMINED RIGHT: $R_x > R_b$ and A is order m×n with n>m and rank A=m

OVERDETERMINED LEFT: $C_x < C_b$ and A has order m×n with n>m and rank A = m
OVERDETERMINED RIGHT: $R_x < R_b$ and A has order m×n with m>n and rank A = n

When the system is consistent but A is rank-deficient, **LSOVL** and **LSOVR** give you a correct solution, but do not tell you that it is not unique. The M matrix on page 234 with B = [[2][2][2]] has solutions of the form [[−1][0][1]] + α [[−1][2][−1]] for arbitrary $\alpha \in \mathbb{C}$, but **LSOVR** only gives you the α = .448053766453 solution. When you doubt that A is full rank, use **LSVDL** and **LSVDR**. They give the unique minimum Euclidean norm solution and also return the singular values. If the above six commands abort with a divide–by–zero error, A is probably rank–deficient.

LEAST SQUARES SVD SOLVE	**LSVDL**(B,A,ε, MAX)	$B \in \mathbb{C}$ $A \in \mathbb{C}$ $\varepsilon > 0$	2: SVD MATRIX 1: SOLUTION
LEAST SQUARES SVD SOLVE	**LSVDR**(B,A,ε, MAX)	$B \in \mathbb{C}$ $A \in \mathbb{C}$ $\varepsilon > 0$	2: SVD MATRIX 1: SOLUTION

LSVDL provides XA = B solution and **LSVDR** provides AX = B solution. The above six commands are special cases of these two commands. These two commands require considerably more execution time, but do provide least squares solutions in the rank–deficient case where A^HA in the right solution or AA^H in the left solution is singular. Parameter ε sets the accuracy of the SVD, and MAX is the maximum number of allowed iterations per column to converge each singular value. If the MAX value (say 20) is reached, then the current estimate of the SVD matrix probably having nonzero superdiagonal elements is returned with a wrong solution. **SVDMI** computes the inverse SVD matrix using 10ε for its threshold. When m < n, the matrix is internally decomposed by **SVD** in transposed form.

LINEAR ALGEBRA MENU {LINAG}

FUNCTION	COMMAND	INPUTS	OUTPUTS
The column and iteration numbers are displayed during execution. **CONVERGENCE FAILED** is displayed when **LSVDL** or **LSVDR** fails to converge according to the user's input arguments. Push **ATTN**. The column number tells where the convergence failed.			
LEAST SQUARES ERROR	**LSERL**(B,A,X)	B ∈ℂ A ∈ℂ X ∈ℂ	VALUE
LEAST SQUARES ERROR	**LSERR**(B,A,X)	B ∈ℂ A ∈ℂ X ∈ℂ	VALUE

These commands return the root mean square error of XA – B and AX – B.

EXAMPLE LINEAR AND LEAST SQUARES SOLUTIONS

In this first example, the A matrix is square and has full rank. Therefore, all four right hand solver commands given above compute the same solution.

$$\mathbf{A} = \begin{bmatrix} (12,0) & (2,-3) & (-1,4) \\ (5,2) & (5,3) & (-2,1) \\ (1,7) & (7,4) & (6,3) \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} (4,2) & (3,1) \\ (6,3) & (9,2) \\ (6,-2) & (-5,3) \end{bmatrix}$$

$$\mathbf{X} = \begin{bmatrix} (.1757,.4159) & (.4006,.8958) \\ (.7636,-.5312) & (.9828,-.7436) \\ (-.03,-.482) & (-1.1596,.6755) \end{bmatrix}$$

The error for the first three linear solvers is 3.317E–11 while that for **LSVDR** is 9.523E–11 using ε = 1E–11. The singular values are 17.3368, 4.1064, 9.1964, and the SVD matrix for this example is

LINEAR ALGEBRA MENU {LINAG}

FUNCTION	COMMAND	INPUTS	OUTPUTS

$$\begin{bmatrix} 17.3368 & 0 & 0 \\ 0 & 4.1064 & 0 \\ 0 & 0 & 9.1964 \end{bmatrix}.$$

Taking the Hermitian transpose of the above B vector, this example can be repeated for the four left–hand solution commands. For the same matrix A, the result is

$$\mathbf{B} = \begin{bmatrix} (4,-2) & (6,-3) & (6,2) \\ (3,-1) & (9,-2) & (-5,-3) \end{bmatrix},$$

$$\mathbf{X} = \begin{bmatrix} (.0728,-.2545) & (-.0601,-.5677) & (.6075,-.2406) \\ (-.7478,.435) & (.1.5866,-1.1284) & (-.4895,-.3248) \end{bmatrix}.$$

Now that the above eight linear solvers have been demonstrated in the general complex case, let us focus on some simple real cases to better understand the capabilities of this software. Consider the underdetermined case using **USOVR**:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 5 & 7 \\ 1 & 5 & 6 & 7 \\ 1 & 8 & 9 & 10 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 1 \\ 5 \\ 4 \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} 9.2 \\ 2.1333 \\ -2.4 \\ -.0667 \end{bmatrix}.$$

The mean squared error is 3.6729E–11. An example overdetermined case using **OSOVR** is

LINEAR ALGEBRA MENU {LINAG}

FUNCTION	COMMAND	INPUTS	OUTPUTS

$$A = \begin{bmatrix} 1 & 5 & 6 \\ 1 & 0 & 5 \\ 5 & 2 & 3 \\ 8 & 4 & 3 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \quad X = \begin{bmatrix} .52109 \\ -.26155 \\ .29746 \end{bmatrix},$$

where the mean squared error is 0.03057. If B^T = [5 2 3 4], then the solution is X^T = [.11842 .5032 .39146] and the error is 0.27514. With ε = 1E−11 and MAX = 10, **LSVDR** gives the identical result with the singular values 13.0780, 5.9392, 2.9480. Consider next the rank-deficient example where A has a rank of 2. Using **USOVR** we obtain

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 5 & 6 & 7 \\ 1 & 8 & 9 & 10 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad X = \begin{bmatrix} 0 \\ 1/9 \\ 1/9 \\ 1/9 \end{bmatrix}.$$

LSVDR gives the same result with the singular values 19.6403, 1.12246, 0.

CONVERGENCE FAILED is displayed when **LSVDL**, **LSVDR**, and **SVD** fail to converge according to the user's input arguments. Push **ATTN**. The column number tells where convergence failed.

SINGULAR VALUE DECOMPOSITION	**SVD**(A,ε,F,MAX) ε > 0	A ∈ ℂ	3: U IF F=1 ONLY 2: V IF F=1 ONLY 1: D

Given arbitrary m × n matrix A, **SVD** computes the singular value decomposition D = U^H A V, where U and V are unitary and U D V^H = A. D has the same dimensions as A, and the magnitude of the major diagonal elements are the singular values.

LINEAR ALGEBRA MENU {LINAG}

FUNCTION	COMMAND	INPUTS	OUTPUTS

In the special case where A is symmetric, U = V, and the symmetric Schur decomposition **SCRSD** can be used to compute the same result.

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix} \qquad \varepsilon = 1E\text{-}11 \qquad \mathbf{D} = \begin{bmatrix} 25.46241 & 0 & 0 \\ 0 & 1.29066 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{U} = \begin{bmatrix} .1409 & .8247 & -.0115 & -.5476 \\ .3439 & .4263 & .4235 & .7216 \\ .547 & .0278 & -.8125 & .1997 \\ .7501 & -.3706 & .4005 & -.3736 \end{bmatrix} \qquad \mathbf{V} = \begin{bmatrix} .5045 & -.7608 & .4082 \\ .5745 & -.0571 & -.8165 \\ .6445 & .6465 & .4082 \end{bmatrix}.$$

The algorithm uses **HBDDR** to decompose complex A into a bidiagonal, non-negative real matrix, which is then diagonalized using Givens rotations and double implicit Wilkinson shifting relative to A^HA to speed the convergence. For the above example, ABS(UDV^H – A) = 6.565E–10. Consider next the example

$$\mathbf{A} = \begin{bmatrix} (1,2) & (2,2) & (3,2) \\ (4,1) & (5,3) & (6,1) \\ (7,2) & (8,3) & (9,3) \end{bmatrix} \qquad \mathbf{D} = \begin{bmatrix} 18.08731 & 0 & 0 \\ 0 & 1.04124 & 0 \\ 0 & 0 & 1.32851 \end{bmatrix}$$

using ε = 1E–11. The error is ABS(UDV^H – A) = 3.315E–10. Flag F specifies whether the unitary matrices U and V are computed, and parameter MAX sets the maximum number of iterations allowed to converge each column. If MAX value (say 20) is reached, then the current estimate of the SVD matrix probably having nonzero superdiagonal elements is returned with the associated unitary matrices if requested. When m < n, the matrix is internally decomposed in transposed form to minimize the execution time. The column and iteration numbers are displayed during execution.

LINEAR ALGEBRA MENU {LINAG}

FUNCTION	COMMAND	INPUTS	OUTPUTS
EXTRACT DIAGONAL	**B←M**(M,O)	M ∈ℂ O ∈𝕀	[VECTOR]

B←M extracts any set of diagonal elements from an m × n matrix M. If offset O equals zero, then the main diagonal is extracted. When O > 0, then the Oth superdiagonal is extracted. When O < 0, then the Oth subdiagonal is extracted.

$$\mathbf{M} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$$

B←M(M,0) = [1 6 11]
B←B(M,2) = [3 8]
B←M(M,-2) = [9]

TRANSPOSE	**TRNP**(A)	A ∈ℂ	MATRIX
HERMITIAN TRANSPOSE	**TRNH**(A)	A ∈ℂ	MATRIX

The Hermitian transpose of A is the transpose of the complex conjugate of A.

TEST IF SYMMETRIC	**SYM?**(A,ε)	A ∈ℂ ε > 0	1 IF TRUE, ELSE 0
TEST IF HERMITIAN	**HRM?**(A,ε)	A ∈ℂ ε > 0	1 IF TRUE, ELSE 0
TEST IF ORTHOGONAL	**ORTH?**(A,ε)	A ∈ℂ ε > 0	1 IF TRUE, ELSE 0
TEST IF UNITARY	**UNIT?**(A,ε)	A ∈ℂ ε > 0	1 IF TRUE, ELSE 0

LINEAR ALGEBRA MENU {LINAG}

FUNCTION	COMMAND	INPUTS	OUTPUTS
In the above four tests, $\varepsilon > 0$ is used to set the numerical precision. For example, **SYM?** computes A minus its transpose and evaluates the maximum of the absolute value of that difference. If it is less than ε, a 1 is returned, and if it is not, a 0 is returned.			
HP CONSTANT MATRIX	**CON**	2: SIZE OR MATRIX 1: VALUE	MATRIX
HP IDENTITY MATRIX	**IDN**	NUMBER OF ROWS	MATRIX
HP REDIMENSION	**RDM**	2: OLD MATRIX 1: SIZE	NEW MATRIX
See page 359 of the HP 48 owner's manual for complete definitions.			
ROW REORDER	**RORDR**(M,V)	$M \in \mathbb{C}$ $V \in \mathbb{N}$	MATRIX M
Given an m × n matrix M and an integer vector V of size m with elements 1, 2, . . ., m in some order, **RORDR** interchanges the order of the rows of M according to the content of V. An example is given below and is continued on the next page.			

$$\mathbf{M} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad \mathbf{V} = [3\ 1\ 2] \quad \Rightarrow \quad \begin{bmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \\ 1 & 2 & 3 \end{bmatrix}$$

LINEAR ALGEBRA MENU {LINAG}

FUNCTION	COMMAND	INPUTS	OUTPUTS

The interpretation of the elements of V is to regard it as a "where I want that row to be" specification. [3 1 2] says put the first row in the third row, the second row in the first row, and finally put the third row in the second row. V can be created by **SRTI**, by several of the matrix decomposition commands in this menu, or by hand. Now the inverse transformation is "where did I come from," which is in this example [2 3 1]. A way to do these permutations is to construct a permutation matrix P by creating an m × m identity matrix using **IDN** and then using **RORDR** to order it. Then pre-multiplication (P × M) by P reorders M. Since $P \times P^{-1} = I$ equals the identity matrix I, the inverse row permutation is produced by pre-multiplication by P^{-1}.

COL REORDER	**CORDR**(M,V)	M ∈ **C** V ∈ **N**	MATRIX M

Given an m × n matrix M and an integer vector V of size n with elements 1, 2, . . ., n in some order, **CORDR** interchanges the order of the columns of M according to the content of V.

$$\mathbf{M} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad \mathbf{V} = [3\ 1\ 2] \quad \Rightarrow \quad \begin{bmatrix} 2 & 3 & 1 \\ 5 & 6 & 4 \\ 8 & 9 & 7 \end{bmatrix}$$

Column permutations are very similar to row permutations. V is the "where do I want the columns to be" vector. If P is a column permutation matrix, then post-multiply by P (M × P) to permute.

INVERSE ORDER	**IORDR**(V)	V = [VECTOR] ∈ **N**	[VECTOR]

Given any valid row or column permutation vector V such as used in **RORDR** and **CORDR**, this command creates the required vector V to put the rows or columns back where they were.

LINEAR ALGEBRA MENU {LINAG}

FUNCTION	COMMAND	INPUTS	OUTPUTS
MATRIX INVERSE	**MINV**(M)	MATRIX M	INVERSE MATRIX
MINV is slower than **INV**, but more accurate. It solves MX = I for X with **LSOVR**.			
HILBERT MATRIX	**HILBT**(N)	N	MATRIX
Creates an N × N Hilbert test matrix.			
HP ROW NORM	**RNRM**	MATRIX	NORM
HP COLUMN NORM	**CNRM**	MATRIX	NORM
For the above HP commands, see page 359 of the HP 48 owner's manual.			
ROW ABS	**RABS**(A)	MATRIX $A \in \mathbb{C}$	[VECTOR]
Computes the square root of the sum of the squares of the absolute values of the elements in each row and returns the values in a vector. This is the Frobenius (Euclidean) norm of each row.			
COLUMN ABS	**CABS**(A)	MATRIX $A \in \mathbb{C}$	[VECTOR]
Computes the square root of the sum of the squares of the absolute values of the elements in each column and returns the values in a vector. This is the Frobenius (Euclidean) norm of each column.			

LINEAR ALGEBRA MENU {LINAG}

FUNCTION	COMMAND	INPUTS	OUTPUTS
LU DECOMPOSITION	**GLUD**(A)	SQUARE MATRIX $A \in \mathbb{C}$	3: PIVOT VECTOR 2: L MATRIX 1: U MATRIX

Employs Gaussian transformations to decompose matrix A into the product of a lower triangular matrix L times an upper triangular matrix U. Row pivoting is employed to improve the accuracy.

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 5 & 6 \\ 2 & 7 & 9 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 \\ .5 & 1 & 0 \\ .5 & -1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 7 & 9 \\ 0 & 1.5 & 1.5 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 7 & 9 \\ 1 & 5 & 6 \\ 1 & 2 & 3 \end{bmatrix} = \mathbf{PA}$$

The pivot vector [3 2 1] is returned to stack 3, the L matrix to stack 2, and the U matrix to stack 1. Observe that the product is the row-pivoted version of A. The Gaussian decomposition matrix which computed U from A is the product $L^{-1} \times P$, where P is the permutation matrix created by applying the pivot vector to an identity matrix I. Note that A is singular, giving U a row of 0s.

$$\mathbf{P} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad \mathbf{L^{-1}} = \begin{bmatrix} 1 & 0 & 0 \\ -.5 & 1 & 0 \\ -1 & 1 & 1 \end{bmatrix} \quad \mathbf{L^{-1}P} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & -.5 \\ 1 & 1 & -1 \end{bmatrix} \quad \mathbf{L^{-1}PA} = \mathbf{U}$$

[3 2 1] is an example of a pivot vector that equals its own inverse.

LOWER × DIAGONAL × UPPER DECOMPOSITION	**LDLTD**(M)	NONSINGULAR SQUARE MATRIX $M \in \mathbb{C}$	4: PIVOT VECTOR 3: L MATRIX 2: DIAG VECTOR 1: U MATRIX

This decomposition of a nonsingular matrix uses Gaussian transformations to express the row-pivoted matrix M as the product of L × D × U. For example:

LINEAR ALGEBRA MENU {LINAG}

FUNCTION	COMMAND	INPUTS	OUTPUTS

$$M = \begin{bmatrix} 2 & 4 & 3 & 2 \\ 3 & 6 & 5 & 2 \\ 2 & 5 & 2 & -3 \\ 4 & 5 & 14 & 14 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ .5 & 1 & 0 & 0 \\ .75 & .9 & 1 & 0 \\ .5 & .6 & 1 & 1 \end{bmatrix} \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 2.5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & .5 \end{bmatrix} \begin{bmatrix} 1 & 1.25 & 3.5 & 3.5 \\ 0 & 1 & -2 & -4 \\ 0 & 0 & 1 & -.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 5 & 14 & 14 \\ 2 & 5 & 2 & -3 \\ 3 & 6 & 5 & 2 \\ 2 & 4 & 3 & 2 \end{bmatrix},$$

where the pivot vector is [4 3 2 1], LDU = PM, and P is the permutation matrix. Note that the diagonal matrix is returned as a vector. The command **D→M** discussed below will convert that vector into a diagonal matrix for multiplying.

APPLY GAUSS TRANSFORM	**AGAUS**(A,V,R,C)	A ∈ℂ V ∈ℂ R C	REDUCED MATRIX
COMPUTE TRANSFORM	**GAUSS**(A,C,RS, RF)	A ∈ℂ C RS RF	VECTOR V

AGAUS and **GAUSS** are the two internal programs which perform the LU decomposition. Given matrix A, the column C to be processed, the first row of that column RS, and the final row of that column RF, **GAUSS** computes the vector V, which **AGAUS** requires to transform the elements of column C from RS + 1 to RF to zero. Given the original matrix A and the vector V, plus the starting row R and starting column C, **AGAUS** processes matrix A and returns the result.

An example is given on the next page.

LINEAR ALGEBRA MENU {LINAG}

FUNCTION	COMMAND	INPUTS	OUTPUTS

$$A = \begin{bmatrix} 2 & 7 & 9 \\ 1 & 5 & 6 \\ 1 & 2 & 3 \end{bmatrix} \quad \text{GAUSS(A,1,1,3)} = [\ 1\ .5\ .5\ ] = V$$

$$\text{AGAUS(A,V,1,1)} = \begin{bmatrix} 2 & 7 & 9 \\ 0 & 1.5 & 1.5 \\ 0 & -1.5 & -1.5 \end{bmatrix}$$

HOUSEHOLDER RANK AND QR DECOMPOSITION	**HRQR**(A,ε)	MATRIX A ∈ ℂ m ≥ n ε > 0	4: LIST 3: PIVOT VECTOR 2: RANK 1: R MATRIX

HRQR decomposes m × n matrix A into the product of a unitary matrix Q and an upper triangular matrix R. The number of rows cannot be less than the number of columns. Column pivoting is used to preserve accuracy. This command computes matrix R, and the **HBDU** command computes Q from the output list. For ε = 1E–8:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 5 & 6 \\ 1 & 8 & 9 \\ 1 & 11 & 12 \end{bmatrix} \qquad R = \begin{bmatrix} -16.432 & -1.826 & -14.606 \\ 0 & -.8165 & .8165 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The rank of this matrix is 2, and the pivot vector equals [2 3 1]. The list that is output allows reconstruction of the unitary (orthogonal since A is real) Q matrix. For this example, it equals

LINEAR ALGEBRA MENU {LINAG}

FUNCTION	COMMAND	INPUTS	OUTPUTS

{ [[1] [.309] [.463] [.618]] [[1] [–.327] [–.789]] },

which is a list containing two column vectors since the rank of A is two.

The SVD is the most reliable rank determination.

HOUSEHOLDER QR Q COMPUTATION	**HBDU**(L)	L = LIST FROM **HRQR** OR **HBDD**	UNITARY Q OR U MATRIX

HBDU computes the unitary (orthogonal) Q matrix from the list output of **HRQR** and **HBDD**. Since Q is unitary (orthogonal), $R = Q^H \times A \times \Pi$ as shown in the example:

$$\mathbf{Q} = \begin{bmatrix} -.183 & -.816 & -.400 & -.374 \\ -.365 & -.408 & .255 & .797 \\ -.548 & 0 & .691 & -.472 \\ -.730 & .408 & -.546 & .0488 \end{bmatrix} \quad \Pi = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \quad \mathbf{Q\,R} = \begin{bmatrix} 3 & 1 & 2 \\ 6 & 1 & 5 \\ 9 & 1 & 8 \\ 12 & 1 & 11 \end{bmatrix} = \mathbf{A}\,\Pi$$

where Π denotes the column pivot matrix associated with the **HRQR** pivot vector.

HOUSEHOLDER BIDIAGONAL DECOMPOSITION	**HBDD**(A)	MATRIX $A \in \mathbb{C}$ $m \geq n \geq 2$	3: U LIST 2: V LIST 1: B MATRIX

HBDD computes the bidiagonal decomposition of matrix A. $U \times B \times V^H = A$ and $B = U^H \times A \times V$ where matrices U and V are unitary (orthogonal if A real). The number of columns cannot exceed the number of rows. An example is given on the next page.

LINEAR ALGEBRA MENU {LINAG}

FUNCTION	COMMAND	INPUTS	OUTPUTS

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix} \qquad B = \begin{bmatrix} -12.884 & 21.876 & 0 \\ 0 & 2.246 & -.613 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The U LIST = {[[1] [.288] [.504] [.720]] [[1] [–.289] [–.753]] [[1] [–.434]]} and the V LIST = { [[1] [.447]] }. The U unitary matrix is evaluated from the U list using **HBDU**, while the V unitary matrix is evaluated from the V list using **HBDV** below.

$$U = \begin{bmatrix} -.078 & -.833 & -.011 & -.548 \\ -.310 & -.451 & .423 & .722 \\ -.543 & -.069 & -.812 & .200 \\ -.776 & .312 & .400 & -.374 \end{bmatrix} \qquad V = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -.667 & -.745 \\ 0 & -.745 & .667 \end{bmatrix}$$

HOUSEHOLDER COMPUTATION OF V MATRIX	**HBDV**(L)	OUTPUT LIST FROM **HBDD**	UNITARY MATRIX V

HBDV constructs the V matrix from the **HBDD** list. In the case where the decomposed matrix has only two columns, **HBDD** returns the list { }. Nevertheless, **HBDV**({ }) properly constructs V.

LINEAR ALGEBRA MENU {LINAG}

FUNCTION	COMMAND	INPUTS	OUTPUTS
HOUSEHOLDER TRIDIAGONAL DECOMPOSITION	HTRDD(A,F)	SYMMETRIC OR HERMITIAN MATRIX	2: V IF F = 1 1: T MATRIX

HTRDD computes the tridiagonal decomposition of a Hermitian (symmetric if real) matrix A. $V \times T \times V^H = A$ and $V^H \times A \times V = T$, where matrix V is unitary (orthogonal if A real) and T is real even when A is complex. If F = 0, then only the tridiagonal matrix is returned. For example:

$$\mathbf{A} = \begin{bmatrix} 1 & 3 & 4 \\ 3 & 2 & 8 \\ 4 & 8 & 3 \end{bmatrix} \quad \mathbf{T} = \begin{bmatrix} 1 & -5 & 0 \\ -5 & 10.32 & 1.76 \\ 0 & 1.76 & -5.32 \end{bmatrix} \quad \mathbf{V} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -.6 & -.8 \\ 0 & -.8 & .6 \end{bmatrix}.$$

HOUSEHOLDER UPPER HESSENBERG DECOMPOSITION	UHESD(A,F)	ARBITRARY SQUARE COMPLEX MATRIX	2: V IF F = 1 1: H MATRIX

Given the square matrix $A \in \mathbb{C}$, **UHESD** will perform the decomposition $V^H \times A \times V = H$, where H is upper Hessenberg and unitary V is the product of Householder matrices. For example:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad \mathbf{V} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -.496 & -.868 \\ 0 & -.868 & .496 \end{bmatrix} \quad \mathbf{H} = \begin{bmatrix} 1 & -3.597 & -.248 \\ -8.062 & 14.046 & 2.831 \\ 0 & .8308 & -.046 \end{bmatrix}$$

LINEAR ALGEBRA MENU {LINAG}

FUNCTION	COMMAND	INPUTS	OUTPUTS
HERMITIAN SCHUR DECOMPOSITION	**SCRSD**(A,ε,F, MAX)	SQUARE $A \in \mathbb{C}$ $\varepsilon > 0$	2: Q IF F = 1 ONLY 1: D

Cascading **HTRDD** with **EIGNS** to produce the Schur decomposition is the standard means by which the dense symmetric and Hermitian eigen problem is solved. **EIGNS** employs double implicit Wilkinson shifting to speed the convergence of the eigenvalues and eigenvectors. $Q^H A Q = D$ and $Q D Q^H = A$, where Q is unitary and D is diagonal. The eigenvalues are the diagonal values of D, and the columns of Q are the eigenvectors. For ε = 1E−10 and F = 1:

$$A = \begin{bmatrix} (2,0) & (1,-3) & (2,4) \\ (1,3) & (5,0) & (2,1) \\ (2,-4) & (2,-1) & (3,0) \end{bmatrix}, \quad D = \begin{bmatrix} (-3.2826,0) & (0,0) & (0,0) \\ (0,0) & (8.7638,0) & (0,0) \\ (0,0) & (0,0) & (4.5188,0) \end{bmatrix},$$

$$Q = \begin{bmatrix} (-.7075,0) & (-.5714,0) & (-.186,-.3721) \\ (-.0258,.3737) & (-.4711,-.3865) & (.4027,.5711) \\ (.1739,-.5735) & (-.4288,.3441) & (-.3185,.4869) \end{bmatrix}.$$

F = 1 requests that the unitary matrix Q be returned, and ε is used by **EIGNS**.

MAX is the maximum number of iterations allowed to converge each eigenvalue.

CONVERGENCE FAILED is displayed when **SCRSD**, **SCHRD**, **EIGNS**, and **EIGEN** fail to converge according to the user's input arguments. Push **ATTN**. The column number tells where convergence failed.

LINEAR ALGEBRA MENU {LINAG}

FUNCTION	COMMAND	INPUTS	OUTPUTS
GENERAL COMPLEX SCHUR DECOMPOSITION	**SCHRD**(A,ε,F, MAX)	SQUARE $A \in \mathbb{C}$ $\varepsilon > 0$	3: V IF F = 2 2: Q or H IF F ≠ 0 1: T

Cascading **UHESD** and **EIGEN** to produce the Schur decomposition is the standard means by which the dense general eigen problem is solved. **EIGEN** employs double implicit real and complex shifting to speed the convergence of the eigenvalues and Schur vectors. $Q^H A Q = T$ and $Q T Q^H = A$, where Q is unitary and T is upper triangular. The eigenvalues are the diagonal values of T, and the columns of Q are the Schur vectors. For example with ε = 1E–10:

$$\mathbf{A} = \begin{bmatrix} (1,2) & (2,1) & (3,1) \\ (1,4) & (4,2) & (5,2) \\ (1,3) & (7,5) & (9,2) \end{bmatrix}$$

$$\mathbf{T} = \begin{bmatrix} (13.2157,6.5589) & (1.8464,3.7257) & (-3.3742,1.4993) \\ (0,0) & (-.1466,.1489) & (-.178,2.2271) \\ (0,0) & (0,0) & (.9309,-.7079) \end{bmatrix}$$

$$\mathbf{Q} = \begin{bmatrix} (-.0684,-.274) & (-.2294,-.5065) & (-.2547,-.7391) \\ (-.0867,-.4927) & (-.5374,-.3223) & (.4059,.4385) \\ (-.2304,-.7855) & (.4344,.3308) & (-.1781,.0123) \end{bmatrix}$$

ε is used by **EIGEN**. F = 1 requests that the unitary matrix Q associated with the Schur decomposition be returned. For F = 2, matrices V and H associated with the **UHESD** output are returned to levels 3 and 2 of the stack, while the T output of **EIGEN** is returned to level 1. MAX is the maximum number of iterations allowed to converge each eigenvalue. A program for computing the eigenvectors of A from the **SCHRD** output with F = 2 is given at the end of this chapter.

LINEAR ALGEBRA MENU {LINAG}

FUNCTION	COMMAND	INPUTS	OUTPUTS
SYMMETRIC EIGEN MATRIX DECOMPOSITION	**EIGNS**(A,ε,F,MAX)	SQUARE A ∈ **C** ε > 0	2: Q IF F = 1 ONLY 1: D

Given tridiagonal matrix A, **EIGNS** performs unitary transformations to reduce it to a diagonal matrix whose diagonal values are the eigenvalues of A. U^H A U = D and U D U^H = A. Wilkinson double implicit shifting is employed to speed convergence of the eigenvalues and eigenvectors. The algorithm iterates on each diagonal value $a_{k,k}$ until the absolute value of $a_{k,k-1} < \varepsilon$. The column and iteration numbers are displayed during execution.

$$\mathbf{A} = \begin{bmatrix} 1 & 3 & 0 & 0 \\ 3 & 5 & 2 & 0 \\ 0 & 2 & 6 & 4 \\ 0 & 0 & 4 & 8 \end{bmatrix} \qquad \varepsilon = 1E\text{-}10$$

$$\mathbf{D} = \begin{bmatrix} -.80074 & 0 & 0 & 0 \\ 0 & 2.56635 & 0 & 0 \\ 0 & 0 & 6.82303 & 0 \\ 0 & 0 & 0 & 11.41135 \end{bmatrix}$$

$$\mathbf{U} = \begin{bmatrix} .83614 & .34704 & -.41969 & -.0655 \\ -.50189 & .1812 & -.81461 & -.22733 \\ .20145 & -.74104 & -.11301 & -.63048 \\ -.09156 & .54552 & .38406 & -.73927 \end{bmatrix}$$

LINEAR ALGEBRA MENU {LINAG}

FUNCTION	COMMAND	INPUTS	OUTPUTS
Flag F specifies whether the unitary matrix U is computed, and parameter MAX sets the maximum number of iterations allowed to converge each column. If MAX value (say 20) is reached, then the current estimate of the eigenvalue matrix probably having nonzero superdiagonal elements is returned with the associated unitary matrices if requested.			
GENERAL EIGEN MATRIX DECOMPOSITION	**EIGEN**(A,ε,F,MAX)	SQUARE A $\in \mathbb{C}$ $\varepsilon > 0$	2: Q IF F = 1 ONLY 1: T

Given upper Hessenberg matrix A, **EIGEN** performs unitary transformations to reduce it to an upper diagonal matrix whose diagonal values are the eigenvalues of A. $U^H A U = T$ and $U T U^H = A$. Both real and complex double implicit shifting are employed to speed convergence of the eigenvalues and Schur vectors. The algorithm iterates on each value $a_{k,k}$ until the absolute value of $a_{k,k-1} < \varepsilon$.
For ε = 1E−10, we have

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 0 & 9 & 10 & 11 \\ 0 & 0 & 12 & 13 \end{bmatrix}$$

$$T = \begin{bmatrix} (-1.5432,0) & (-2.9487,4.341) & (-1.1712,1.8399) & (-.8931,.3359) \\ (0,0) & (2.4856,-1.5511) & (3.5722,2.9824) & (1.9384,2.898) \\ (0,0) & (0,0) & (2.4856,1.5511) & (5.1333,.0921) \\ (0,0) & (0,0) & (0,0) & (26.5721,0) \end{bmatrix}$$

LINEAR ALGEBRA MENU {LINAG}

FUNCTION	COMMAND	INPUTS	OUTPUTS

$$U = \begin{bmatrix} (-.2286,-.086) & (-.4133,-.0631) & (.8389,.2421) & (.0561,0) \\ (.1877,.0706) & (.6623,-.5069) & (.382,-.1893) & (.287,0) \\ (-.685,-.2576) & (.0713,.032) & (-.2045,-.0486) & (.6436,0) \\ (.5652,.2126) & (-.3008,.1816) & (-.0355,.1019) & (.7073,0) \end{bmatrix}$$

Flag F specifies whether the unitary matrix U is computed, and parameter MAX sets the maximum number of iterations allowed to converge each column. If MAX value (say 20) is reached, then the current estimate of the eigenvalue matrix probably having nonzero superdiagonal elements is returned with the associated unitary matrices if requested.

SYMMETRIC MATRIX EIGEN STEP	**WSQR**(A,F,L)	SQUARE A $\in \mathbb{C}$ F = FIRST ROW L = LAST ROW	MATRIX A

Performs a single eigen iteration on A from row (column) F to row L. This program gives the user access to the internal symmetric (Hermitian) eigen software, and loops can be built around it. Matrix A must be tridiagonal. The user may iterate over any portion of the matrix.

$$A = \begin{bmatrix} 1 & 3 & 0 & 0 \\ 3 & 5 & 2 & 0 \\ 0 & 2 & 6 & 4 \\ 0 & 0 & 4 & 8 \end{bmatrix} \qquad \text{WSQR(A,1,4)} = \begin{bmatrix} -.3116 & 1.535 & 0 & 0 \\ 1.535 & 4.95 & 2.1072 & 0 \\ 0 & 2.1072 & 3.9614 & -.2738 \\ 0 & 0 & -.2738 & 11.4002 \end{bmatrix}$$

A second application of WSQR(A,1,4) will converge $A_{4,4}$ to the eigenvalue 11.41135.

LINEAR ALGEBRA MENU {LINAG}

FUNCTION	COMMAND	INPUTS	OUTPUTS
GENERAL MATRIX EIGEN STEP	**GSQR**(A,F,L)	SQUARE A $\in \mathbb{C}$ F = FIRST ROW L = LAST ROW	MATRIX A

Performs a single eigen iteration on A from row (column) F to row L. This program gives the user access to the internal general complex eigen software, and loops can be built around it. Matrix A must be upper Hessenberg. The user may iterate over any portion of the matrix.

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 0 & 9 & 10 & 11 \\ 0 & 0 & 12 & 13 \end{bmatrix} \qquad \mathbf{GSQR(A,1,4)} = \begin{bmatrix} -.2637 & -.1607 & -.9768 & 2.4129 \\ 4.0963 & 1.0068 & -.5293 & 3.3281 \\ 0 & 6.9457 & 2.8476 & -1.7658 \\ 0 & 0 & -5.1049 & 26.4092 \end{bmatrix}$$

Two more applications of GSQR(A,1,4) converge $A_{4,4}$ to the eigenvalue 26.5721.

TRACE	**TRACE**(M)	MATRIX M	VALUE
MATRIX MINOR	**MINOR**(M,R,C)	M ROW# COL#	MATRIX

Performs first step in computing a minor or cofactor, that of extracting the submatrix where row R and column C have been deleted. To complete, call **DET** and adjust the sign.

HOUSE	**HOUSE**(A,C,FR, LR)	A C FR LR	COLUMN VECTOR

Extracts the Cth column subset from A from the first row FR to the last row LR and calls **VHOUS** to compute the appropriate Householder vector for reduction of that column. By transposing A first and properly adjusting parameters C, FR, and LR, **HOUSE** can also be used for row reduction.

LINEAR ALGEBRA MENU {LINAG}

FUNCTION	COMMAND	INPUTS	OUTPUTS
ROW HOUSE	**RHOUS**(A,X,R,C)	A X R C	REDUCED A

Applies Householder matrix in a pre–multiplication sense to introduce zeros into a specified portion of a column. A is the matrix, X is the Householder vector obtained from **HOUSE**, R is the first affected row, and C is the first affected column. Only the lower right part of A is reduced.

COLUMN HOUSE	**CHOUS**(A,X,R,C)	A X R C	REDUCED A

Applies Householder matrix in a post–multiplication sense to introduce zeros into a specified portion of a row. A is the matrix, X is the Householder vector obtained from **HOUSE**, R is the first affected row, and C is the first affected column. Only the upper right part of A is reduced.

HOUSEHOLDER VECTOR	**VHOUS**(V)	V = [VECTOR]	COLUMN VECTOR

Computes Householder vector required to zero out all but the first element of V. For example, let V = [4 2 5 3]. Then the output column vector X is

[[1] [.1762] [.4406] [.2644]].

HOUSEHOLDER MATRIX	**PHOUS**(X)	X = COLUMN VECTOR	MATRIX

Given the Householder vector column vector X, **PHOUS** computes the unitary matrix P such that PV = [a 0 0 ... 0]^T. For the above example, PV = [[–7.348] [0] [0][0]]. When X is real, then P is orthogonal.

LINEAR ALGEBRA MENU {LINAG}

FUNCTION	COMMAND	INPUTS	OUTPUTS
GIVENS ROW (COLUMN) ROTATION	**GIVEN**(A,R1,R2,C)	A R1 R2 C $A \in \mathbb{C}$ $m \geq n$	θ MATRIX
For the complex values of matrix A located at { R1 C } and { R2 C }, **GIVEN** extracts these values a and b, respectively, and calls **GROT** to compute the appropriate θ matrix to zero { R2 C } with **RROT**. Similarly, **GIVEN**(TRNH(A),C1,C2,R) will compute θ to zero { R C2 } with **CROT**. The compressed 2×2 θ matrix is expanded and applied to the appropriate elements by **RROT** and **CROT**.			
GIVENS (JACOBI) ROW ROTATION	**RROT**(θ,A,R1,R2,C)	θ A R1 R2 C $A \in \mathbb{C}$ $m \geq n$	ROTATED A
RROT applies a Givens rotation defined by matrix θ to matrix A starting with column C. The affected rows are R1 and R2. The appropriate value of θ is computed by either **GIVEN** or **GROT**.			
GIVENS COLUMN ROTATION	**CROT**(θ,A,C1,C2,R)	θ A C1 C2 R $A \in \mathbb{C}$ $m \geq n$	ROTATED A
CROT applies a Givens rotation defined by matrix θ to matrix A starting with row R. The affected columns are C1 and C2. The appropriate value of θ is computed by either **GIVEN** or **GROT**.			
GIVENS ROW (COLUMN) ROTATION	**GROT**(a,b)	$a, b \in \mathbb{C}$	θ MATRIX
For complex a and b, **GROT** computes matrix θ so that:			

LINEAR ALGEBRA MENU {LINAG}

FUNCTION	COMMAND	INPUTS	OUTPUTS
$\theta \times \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} z \\ 0 \end{bmatrix}$ OR $[a\ b] \times \theta^H = [z\ 0]$ $z \in \mathbb{C}$			
TO DIAGONAL MATRIX	**D→M**(V)	V = [VECTOR]	DIAGONAL MATRIX
TO DIAGONAL VECTOR	**D←M**(M)	DIAGONAL MATRIX	[VECTOR]

D→M places the elements of vector V into the diagonal of square zero matrix M. Given square matrix M, **D←M** removes the diagonal elements and places them into a vector.

CHOLESKY DECOMPOSITION	**CHOLD**(A,ε,MAX)	$A \in \mathbb{C}$ ε MAX	MATRIX G $A = G\ G^H$

CHOLD computes the Cholesky decomposition of a positive definite Hermitian (symmetric) matrix. $\varepsilon > 0$ sets the convergence accuracy, and MAX is the maximum number of iterations per column.

$$\mathbf{A} = \begin{bmatrix} (12,0) & (2,-3) & (-1,4) \\ (2,3) & (5,0) & (-2,1) \\ (-1,-4) & (-2,-1) & (3,0) \end{bmatrix}$$

LINEAR ALGEBRA MENU {LINAG}

FUNCTION	COMMAND	INPUTS	OUTPUTS

$$G = \begin{bmatrix} (3.34666,0) & (-.85276,0) & (.13667,.23233) \\ (.8419,1.20781) & (1.04667,1.27659) & (-.00759,.32732) \\ (-.31087,-1.35357) & (.06349,-.33287) & (-.60907,.76513) \end{bmatrix}$$

CHOLD uses **SCRSD**. If the real part of any eigenvalue is negative or if the absolute value of the imaginary part of any eigenvalue is greater than 5ε, then the NOT POSITIVE DEFINITE error is returned.

HOUSEHOLDER BIDIAGONAL DECOMPOSITION	**HBDDR**(A)	MATRIX $A \in \mathbb{C}$ $m \geq n \geq 2$	3: U 2: V 1: B

HBDDR uses **HBDD** to bidiagonalize A. Then it computes the unique unitary transformation to rotate the decomposition onto the positive real axis.
$U \times B \times V^H = A$ and $B = U^H \times A \times V$.

INVERT SVD MATRIX	**SVDMI**(D,ε)	ARBITRARY $D \in \mathbb{C}$	SVD INV OF D

SVDMI is an internal command used by the SVD linear solvers. Input D is the matrix of singular values that is output by **SVD**.

The pseudoinverse matrix used in SVD and least squares problems is obtained by inverting the nonzero singular values of matrix D. For those singular values of D for which $|D_{jj}| > \varepsilon$, **SVDDI** replaces those values with their reciprocal values. All others are set to zero. If D is $m \times n$, then the returned inverse is $n \times m$.

LINEAR ALGEBRA MENU {LINAG}

FUNCTION	COMMAND	INPUTS	OUTPUTS
INVERT SVD DIAGONAL VECTOR	**SVDDI**(V,ε)	ARBITRARY $V \in \mathbb{C}$ $\varepsilon > 0$	SVD INVERSE OF V

When computing the SVD of matrices, the singular values can be extracted using the **B←M** command. **SVDDI** performs the same operation as **SVDMI**, except that it performs it on the extracted vector V. **B→M** can be used to convert V back to a square or rectangular diagonal matrix. **SVDDI** is a MATHLIB internal command which is used by **SVDMI** to compute the pseudoinverse matrix.

CHARACTERISTIC POLYNOMIAL	**CPOLY**(M)	SQUARE MATRIX M	POLYNOMIAL LIST

Given square matrix M, **CPOLY** computes its characteristic polynomial DET(M – λI). For example:

$$\mathbf{M} = \begin{bmatrix} 1 & 5 & 2 \\ 4 & 7 & 1 \\ 0 & 5 & 8 \end{bmatrix} \quad \Rightarrow \quad \{\ 69\ \ 46\ \ -16\ \ 1\ \}.$$

The roots of the characteristic polynomial are the eigenvalues (characteristic values) of M. Using **XEQN**(LIST,λ) we obtain the characteristic equation

$$\lambda^3 - 16\lambda^2 + 46\lambda + 69.$$

Using **AROOT**(LIST, 0, 1E–9,100), we compute the eigenvalues:

[–1.07282527025 5.61165926221 11.461166008].

LINEAR ALGEBRA MENU {LINAG}

FUNCTION	COMMAND	INPUTS	OUTPUTS
POLYNOMIAL ROOT	**AROOT**(L,v,ε, MAX)	L v ε MAX	[VECTOR]

AROOT uses Laguerre's method for solving for all the roots of the polynomial defined by the complex coefficient list L. Value v is the initial guess for all of the roots, and ε sets the convergence criterion. MAX (say 100) is the maximum iterations allowed to converge each root. After converging each root, **DEFLT** is used to remove that root from polynomial L.

The most difficult polynomial root solutions occur when there are repeated roots. Consider the example: '$x^{12} + 99x^{11} - 377x^{10} - 26395x^9 + 149080x^8 + 1703048x^7 - 15440048x^6 + 8684864x^5 + 302914240x^4 - 1377763200x^3 + 2718976000x^2 - 2620320000x + 1008000000$' = $(x-2)^5$ (x–5) (x+6) (x–7) (x+10) (x–10) (x+15) (x+100). With v = 0, ε = 1E–8, and MAX = 100, we only get two roots: (1.98607080, 0) and (1.99562237, –1.32705219E–2). Dropping ε = 1E–4 yields the roots: (1.98607080, 0) (1.99563474, –1.32669871E–2) (1.99563459, .0132665015) (2.01132960, 8.27241365E–3) (2.01133027, –8.27192808E–3) (5, 0) (–6, 0) (7, 0) (–10, 0) (10, 0) (–15, 0) (–100, 0). The average of the first 5 roots is (2, 0) to 11 digits. **LROOT** can be used to verify that (2, 0) is the true root and **DEFLT** can be used to remove it from P giving the reduced polynomial { (–31500000, 0) (3135000, 0) (1619500, 0) (–108650, 0) (–23945, 0) (673, 0) (109, 0) 1 } which is easily solved since all the roots are unique.

CONVERGENCE FAILED is displayed if, for the input parameters, **AROOT** cannot converge all the roots. Push **ATTN**.

EIGENVECTORS OF COMPLEX MATRICES – JORDAN CANONICAL FORM

The EVSOV program on the next page solves for the eigenvectors of a general complex square matrix A with distinct eigenvalues, using inverse power iterations. The output is a vector of eigenvalues on level 1 and a nonunitary similarity transformation matrix Z whose columns are the eigenvectors of input matrix A. $D = Z^{-1} A Z$ and $Z D Z^{-1} = A$, where D is diagonal with eigenvalue elements.

LINEAR ALGEBRA MENU {LINAG}

FUNCTION	COMMAND	INPUTS	OUTPUTS

Since **SCHRD** only computes approximate eigenvalues, the software will not detect repeated eigenvalue cases which require block techniques. The resulting D matrix for distinct eigenvalues is the Jordan canonical form and Z the associated similarity transformation. If the eigenvalues are too good, **MINV** may fail, so you must make them not so good. With H – λI almost singular, convergence is very fast. See Chapter 7 of Golub and Van Loan for the algorithms and theory.

EVSOV: ≪ → A ε M ≪ A ε 2 M SCHRD D←M DUP SIZE EVAL DUP IDN { } → V H D N I L ≪ 1 N FOR K "SOLVE VECTOR " K + 3 DISP D K GET H I 3 PICK × – MINV D (1,0) CON → λ Q U ≪ DO Q U × U U CONJ DOT / 'U' STO UNTIL H λ U ε 10 × λV? END 'L' U V→L STO+ ≫ NEXT V L V←L N N 2 →LIST RDM TRNP × D ≫ ≫ ≫

Chapter 27 discusses other approaches to computing Jordan canonical form.

UP DIRECTORY	UPDIR	NONE	PARENT MENU

Bronson, R., *Matrix Methods*, Boston, Academic Press, 1991.
Golub, G. and Van Loan, C., *Matrix Computations*, Baltimore, John Hopkins Univ Press, 1989.
Herstein, I., and Winter, D., *Matrix Theory and Linear Algebra*, New York, Macmillan, 1988.
Horn, R., and Johnson, C., *Matrix Analysis*, New York, Cambridge University Press, 1990.
Press, W., Flannery, B., Teukolsky, S., and Vetterling, W., *Numerical Recipes*, New York, Cambridge Univ. Press, 1989.
Searle, S., *Matrix Algebra Useful for Statistics*, New York, Wiley, 1982.
Wilkinson, J., *The Algebraic Eigenvalue Problem*, Oxford, Clarendon Press, 1965.
Wilkinson, J., and Reinsch, C., *Linear Algebra*, New York, Springer–Verlag, 1971.

21

SPECIAL MATRIX OPERATIONS

INTRODUCTION

This chapter presents the 58 special matrix operations in the MATR menu. The MATR commands include object conversion, sorting, and rearrangement, as well as numerous primitives for extracting and replacing subsets of elements. They allow the user to manipulate the data in matrices without putting the elements on the stack. Commands for creating diagonal matrices, companion matrices, and Hermitian Toeplitz matrices, and testing eigenvalues and eigenvectors are also provided.

OBJECT TYPE CONVERSION

The commands →**ROW** and →**COL** provide conversions from vector objects and symbolic vector objects (lists) to row vectors and column vectors. →**VTR** converts both column and row vectors (matrix objects) into vector objects. **M→RL** and **M←RL** convert between standard matrix objects and a list of vectors, each of which contains the elements of a row of the matrix. Similarly, **M→CL** and **M←CL** convert between a matrix and a list of vectors, where each vector is a column of the matrix. To roll (rotate) the rows or columns of matrix M, see the **VROT** command in Chapter 26.

RROLL: « → M α « M M→RL α VROT M←RL » »

CROLL: « → M α « M M→CL α VROT M←CL » »

SORTING AND REVERSING

RSORT and **RSRTI** sort the rows of a matrix according to the values in a specified column into ascending or descending order. **CSORT** and **CSRTI** similarly sort the columns of a matrix according to the values in a specified row. **REV→** performs a right–left reversal of the columns of a matrix, while **REV↑** performs an up–down reversal of the rows of a matrix.

TESTING EIGENVALUES AND EIGENVECTORS

λ? and **λV?** test the accuracy of eigenvalues and eigenvectors. **EVS?** compares the equivalence of two eigenvectors in the general complex case. A program for testing if a matrix is diagonal is also given.

SPECIAL MATRICES

DIAG and **COMP** create diagonal and companion matrices. **HTOEP** creates Hermitian Toeplitz matrices from their first column.

ELEMENT MANIPULATIONS

Forty-one additional commands provide for value and subset inserting, deleting, swapping, copying, interleaving, de-interleaving, and moving, as well as matrix splitting and splicing.

RANDOM MATRICES

MRDU and **MRDN** provide random matrices with uniform and normal statistics. **MRDC** provides two jointly normal matrices with a user-specified correlation. **MRDC** can also be used to create random complex matrices. **MRDS** and **MRDH** create random symmetric and Hermitian matrices. These five commands are in the MSAG menu discussed in Chapter 31.

SOFTWARE LIMITATIONS

Matrix commands assume a minimum 2 × 2 size, and not row or column vectors.

MATRIX OPERATIONS MENU
{ MATR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
VECTOR TO ROW VECTOR	→**ROW**(V)	[VECTOR]	[[ROW VECTOR]]
VECTOR TO COLUMN VTR	→**COL**(V)	[VECTOR]	[[COL VECTOR]]

→ROW([1 2 3]) = [[1 2 3]] →COL([1 2 3]) = [[1][2][3]]

The inverse command →**VTR** is given below in this menu.

MATRIX ROW SORT ↑ BASED ON A COLUMN	**RSORT**(M,C)	M = MATRIX C = SORT COL	SORTED MATRIX

SORT ↑ = increasing value with increasing column number.

MATRIX ROW SORT ↓ BASED ON A COLUMN	**RSRTI**(M,C)	M = MATRIX C = SORT COL	SORTED MATRIX

$$\mathbf{RSORT}\left(\begin{bmatrix}1 & 8 & 2\\ 9 & 5 & 3\\ 4 & 6 & 7\end{bmatrix}, 1\right) = \begin{bmatrix}1 & 8 & 2\\ 4 & 6 & 7\\ 9 & 5 & 3\end{bmatrix} \qquad \mathbf{RSRTI}\left(\begin{bmatrix}1 & 8 & 2\\ 9 & 5 & 3\\ 4 & 6 & 7\end{bmatrix}, 3\right) = \begin{bmatrix}4 & 6 & 7\\ 9 & 5 & 3\\ 1 & 8 & 2\end{bmatrix}$$

MATRIX OPERATIONS MENU
{ MATR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
MATRIX COL SORT ↑ BASED ON A ROW	**CSORT**(M,R)	M = MATRIX R = SORT ROW	SORTED MATRIX
SORT ↑ = increasing value with increasing row number.			
MATRIX COL SORT ↓ BASED ON A ROW	**CSRTI**(M,R)	M = MATRIX R = SORT ROW	SORTED MATRIX
CSORT $\left(\begin{bmatrix}1&8&2\\9&5&3\\4&6&7\end{bmatrix}, 2\right) = \begin{bmatrix}2&8&1\\3&5&9\\7&6&4\end{bmatrix}$		**CSRTI** $\left(\begin{bmatrix}1&8&2\\9&5&3\\4&6&7\end{bmatrix}, 1\right) = \begin{bmatrix}8&2&1\\5&3&9\\6&7&4\end{bmatrix}$	
EXTRACT ROW SUBSET	**ERWS**(M,R,F,L)	M = MATRIX R = EXTRT ROW F = FIRST COL L = LAST COL	EXTRACTED VECTOR
EXTRACT COLUMN SUBSET	**ECOLS**(M,C,F,L)	M = MATRIX C = EXTRT COL F = FIRST ROW L = LAST ROW	EXTRACTED VECTOR
Examples of these commands are given on the next page.			

MATRIX OPERATIONS MENU
{ MATR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
$\mathbf{ERWS}\left(\begin{bmatrix}1&8&2\\9&5&3\\4&6&7\end{bmatrix},2,1,2\right) = [\,9\;5\,]$ $\mathbf{ECOLS}\left(\begin{bmatrix}1&8&2\\9&5&3\\4&6&7\end{bmatrix},3,1,3\right) = [\,2\;3\;7\,]$			
REPLACE ROW SUBSET	**RRWS**(M,V,R,F)	M = MATRIX V = REPLC VTR R = REPLC ROW F = FIRST COL	MATRIX WITH PART OF ONE ROW REPLACED
REPLACE COLUMN SUBSET	**RCOLS**(M,V,C,F)	M = MATRIX V = REPLC VTR C = REPLC COL F = FIRST ROW	MATRIX WITH PART OF ONE COLUMN REPLACED
$\mathbf{RRWS}\left(\begin{bmatrix}1&8&2\\9&5&3\\4&6&7\end{bmatrix},[\,0\;1\,],3,2\right) = \begin{bmatrix}1&8&2\\9&5&3\\4&0&1\end{bmatrix}$ $\mathbf{RCOLS}\left(\begin{bmatrix}1&8&2\\9&5&3\\4&6&7\end{bmatrix},[\,0\;1\,],2,1\right) = \begin{bmatrix}1&0&2\\9&1&3\\4&6&7\end{bmatrix}$			
EXTRACT PARTIAL SUBMATRIX	**EPSM**(M,R,C,r,c)	M = MATRIX R C r c	SUBMATRIX

MATRIX OPERATIONS MENU { MATR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
REPLACE PARTIAL SUBMATRIX	**RPSM**(M,A,R,C)	MATRICES M & A R C	MATRIX

$$\mathbf{EPSM}\left(\begin{bmatrix}1 & 8 & 2\\ 9 & 5 & 3\\ 4 & 6 & 7\end{bmatrix}, 1, 2, 3, 3\right) = \begin{bmatrix}8 & 2\\ 5 & 3\\ 6 & 7\end{bmatrix}$$

$$\mathbf{RPSM}\left(\begin{bmatrix}1 & 8 & 2\\ 9 & 5 & 3\\ 4 & 6 & 7\end{bmatrix}, \begin{bmatrix}0 & 1\\ 0 & 2\end{bmatrix}, 2, 1\right) = \begin{bmatrix}1 & 8 & 2\\ 0 & 1 & 3\\ 0 & 2 & 7\end{bmatrix}$$

R = First row, C = First column, r = Last column, c = Last column

INSERT ROW	**IROW**(M,V,R)	M = MATRIX V = INSERT VTR R = INSERT ROW	MATRIX WITH AN ADDITIONAL ROW
INSERT COLUMN	**ICOL**(M,V,C)	M = MATRIX V = INSERT VTR C = INSERT COL	MATRIX WITH AN ADDITIONAL COLUMN

$$\mathbf{IROW}\left(\begin{bmatrix}1 & 8 & 2\\ 9 & 5 & 3\\ 4 & 6 & 7\end{bmatrix}, [\,2\ 3\ 6\,], 4\right) = \begin{bmatrix}1 & 8 & 2\\ 9 & 5 & 3\\ 4 & 6 & 7\\ 2 & 3 & 6\end{bmatrix}$$

MATRIX OPERATIONS MENU
{ MATR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
$\text{ICOL}\left(\begin{bmatrix}1&8&2\\9&5&3\\4&6&7\end{bmatrix}, [\,2\,3\,6\,], 2\right) = \begin{bmatrix}1&2&8&2\\9&3&5&3\\4&6&6&7\end{bmatrix}$			
DELETE ROW	**DROW**(M,R)	M = MATRIX R = DEL ROW	MATRIX WITH ONE LESS ROW
DELETE COLUMN	**DCOL**(M,C)	M = MATRIX C = DEL COL	MATRIX WITH ONE LESS COL
$\text{DROW}\left(\begin{bmatrix}1&8&2\\9&5&3\\4&6&7\end{bmatrix}, 2\right) = \begin{bmatrix}1&8&2\\4&6&7\end{bmatrix}$ $\quad$ $\text{DCOL}\left(\begin{bmatrix}1&8&2\\9&5&3\\4&6&7\end{bmatrix}, 3\right) = \begin{bmatrix}1&8\\9&5\\4&6\end{bmatrix}$ Deleting the only row (column) results in an →ARRY error.			
SWAP ROWS	**SROW**(M,R1,R2) M = MATRIX	R1 & R2 ROW NUMBERS	MATRIX WITH ROWS SWAPPED
SWAP COLUMNS	**SCOL**(M,C1,C2) M = MATRIX	C1 & C2 COL NUMBERS	MATRIX WITH COLS SWAPPED
Examples are given on the next page.			

MATRIX OPERATIONS MENU
{ MATR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
SROW$\left(\begin{bmatrix}1&8&2\\9&5&3\\4&6&7\end{bmatrix},1,3\right)=\begin{bmatrix}4&6&7\\9&5&3\\1&8&2\end{bmatrix}$		**SCOL**$\left(\begin{bmatrix}1&8&2\\9&5&3\\4&6&7\end{bmatrix},2,3\right)=\begin{bmatrix}1&2&8\\9&3&5\\4&7&6\end{bmatrix}$	
MOVE ROW	**MROW**(M,F,T) M = MATRIX	F = FROM ROW T = TO ROW	M WITH MOVED ROW
MOVE COLUMN	**MCOL**(M,F,T) M = MATRIX	F = FROM COL T = TO COL	M WITH MOVED COLUMN
MROW$\left(\begin{bmatrix}1&8&2\\9&5&3\\4&6&7\end{bmatrix},1,3\right)=\begin{bmatrix}9&5&3\\4&6&7\\1&8&2\end{bmatrix}$		**MCOL**$\left(\begin{bmatrix}1&8&2\\9&5&3\\4&6&7\end{bmatrix},1,3\right)=\begin{bmatrix}8&2&1\\5&3&9\\6&7&4\end{bmatrix}$	
COPY ROW	**CROW**(M,F,T)	F = FROM ROW T = TO ROW	M WITH ROW F COPIED TO T
COPY COLUMN	**CCOL**(M,F,T)	F = FROM COL T = TO COL	M WITH COL F COPIED TO T
CROW$\left(\begin{bmatrix}1&8&2\\9&5&3\\4&6&7\end{bmatrix},1,3\right)=\begin{bmatrix}1&8&2\\9&5&3\\1&8&2\end{bmatrix}$		**CCOL**$\left(\begin{bmatrix}1&8&2\\9&5&3\\4&6&7\end{bmatrix},1,3\right)=\begin{bmatrix}1&8&1\\9&5&9\\4&6&4\end{bmatrix}$	

MATRIX OPERATIONS MENU
{ MATR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
TRANSPOSE	**TRNP**(M)	M = MATRIX	TRANSPOSED M
TO VECTOR	**→VTR**	COLUMN OR ROW VECTOR	[VECTOR]
→VTR([[1][2][3]]) = [1 2 3] →VTR([[1 2 3]]) = [1 2 3]			
EXTRACT ROW	**EROW**(M,R)	Like ERWS but entire row R	
EXTRACT COL	**ECOL**(M,C)	Like ECOLS but entire column C	
REPLACE ROW	**RROW**(M,V,R)	Like RRWS but entire row R	
REPLACE COL	**RCOL**(M,V,C)	Like RCOLS but entire column C	
EXTRACT LOWER RIGHT HAND SIDE OF MATRIX	**ESBM**(M,R,C)	M = MATRIX R = FIRST ROW C = FIRST COL	SUBMATRIX R=C=1 RETURNS ENTIRE MATRIX
REPLACE LOWER RIGHT HAND SIDE OF MATRIX	**RSBM**(M,A)	M = MATRIX A = SUBMATRIX	MATRIX M AND A MAY BE THE SAME SIZE

The above six commands allow row, column, and submatrix manipulations with fewer dimensional specifications than their above equivalents.

$$\textbf{ESBM}\left(\begin{bmatrix}1&8&2\\9&5&3\\4&6&7\end{bmatrix},1,2\right)=\begin{bmatrix}8&2\\5&3\\6&7\end{bmatrix}$$

$$\textbf{RSBM}\left(\begin{bmatrix}1&8&2\\9&5&3\\4&6&7\end{bmatrix},\begin{bmatrix}0&1\\0&2\end{bmatrix}\right)=\begin{bmatrix}1&8&2\\9&0&1\\4&0&2\end{bmatrix}$$

MATRIX OPERATIONS MENU
{ MATR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
TO ROW LIST	**M→RL**(M)	M = MATRIX	{[ROW1]...}
TO MATRIX	**M←RL**(L)	L = ROW LIST	MATRIX

M→RL([[1 2 3][4 5 6]]) = { [1 2 3] [4 5 6] }

M←RL({ [1 2 3] [4 5 6] }) = [[1 2 3][4 5 6]]

TO COL LIST	**M→CL**(M)	M = MATRIX	{[COL1]...}
TO MATRIX	**M←CL**(L)	L = COL LIST	MATRIX

M→CL([[1 2 3][4 5 6]]) = { [1 4] [2 5] [3 6] }

M←CL({ [1 4] [2 5] [3 6] }) = [[1 2 3][4 5 6]]

ROW INTERLEAVE	**RNLV**(M)	M = MATRIX	MATRIX

Given matrix M with an even number of rows, this command interleaves the second half of the matrix with the first half. For example:

$$\mathbf{M} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix}, \qquad \mathbf{RNLV(M)} = \begin{bmatrix} 1 & 2 & 3 \\ 7 & 8 & 9 \\ 4 & 5 & 6 \\ 10 & 11 & 12 \end{bmatrix}.$$

RNLV can also be used to interleave the values of column vectors.

MATRIX OPERATIONS MENU
{ MATR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
COLUMN INTERLEAVE	**CNLV**(M)	M = MATRIX	MATRIX

Given matrix M with an even number of columns, this command interleaves the right half of the matrix with the left half.

$$\mathbf{M} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix} \qquad \mathbf{CNLV(M)} = \begin{bmatrix} 1 & 3 & 2 & 4 \\ 5 & 7 & 6 & 8 \\ 9 & 11 & 10 & 12 \end{bmatrix}$$

CNLV can also be used to interleave the values of row vectors.

ROW DE–INTERLEAVE	**RDLV**(M)	M = MATRIX	MATRIX
COLUMN DE–INTERLEAVE	**CDLV**(M)	M = MATRIX	MATRIX

Given matrix M with an even number of rows (columns), these commands de-interleave each pair of two rows (columns), thus providing the inverse transformations corresponding to **RNLV** and **CNLV**.

ROW SPLIT	**RSPLT**(M,R)	M = MATRIX	R	TWO MATRICES
COLUMN SPLIT	**CSPLT**(M,C)	M = MATRIX	C	TWO MATRICES

Given matrix M, these commands split M into two matrices, the first having the first R rows (C columns) of the original matrix, and the second having the remainder. Examples are given on the next page.

MATRIX OPERATIONS MENU { MATR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
$M = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$		**RSPLT(M,2)** = $\begin{cases} 2: \begin{bmatrix} 1 & 3 & 2 & 4 \\ 5 & 7 & 6 & 8 \end{bmatrix} \\ 1: [[9 \ \ 11 \ \ 10 \ \ 12]] \end{cases}$	
ROW COMBINE	**RCMB**(M1,M2)	TWO MATRICES	MATRIX
COLUMN COMBINE	**CCMB**(M1,M2)	TWO MATRICES	MATRIX
Given matrices M1 and M2, these commands perform the inverse transformations corresponding to **RSPLT** and **CSPLT** of combining two matrices into one.			
MATRIX REVERSE →	**REV→**(M)	MATRIX M	LEFT RIGHT REV
MATRIX REVERSE ↑	**REV↑**(M)	MATRIX M	UP DOWN REV
REV→$\left(\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}\right) = \begin{bmatrix} 3 & 2 & 1 \\ 6 & 5 & 4 \\ 9 & 8 & 7 \end{bmatrix}$		REV↑$\left(\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}\right) = \begin{bmatrix} 7 & 8 & 9 \\ 4 & 5 & 6 \\ 1 & 2 & 3 \end{bmatrix}$	
ROW GET	**RGET**(M,R,C)	M R C	2: $M_{R,C}$ 1: $M_{R,C+1}$

MATRIX OPERATIONS MENU { MATR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
COLUMN GET	**CGET**(M,R,C)	M R C	2: $M_{R,C}$ 1: $M_{R+1,C}$

$$\mathbf{RGET}\left(\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}, 2, 2\right) = \begin{cases}\mathbf{2:}\ 5\\ \mathbf{1:}\ 6\end{cases}$$

REPLACE DIAGONAL	**B**→**M**(M,V,O)	M, V ∈ ℂ O ∈ 𝕀	MATRIX

B→**M** replaces the offset diagonal elements of M with V. If O = 0, then the main diagonal is replaced. When O > 0, then the Oth superdiagonal is replaced. When O < 0, then the Oth subdiagonal is replaced. Matrix M may be m × n and V must be a [VECTOR] of correct size.

$$\mathrm{B}\rightarrow\mathrm{M}\left(\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}, [\ 10\ 11\ ], -1\right) = \begin{bmatrix}1&2&3\\10&5&6\\7&11&9\end{bmatrix}$$

EXTRACT DIAGONAL	**B**←**M**(M,O)	M ∈ ℂ O ∈ 𝕀	[VECTOR]

B←**M** extracts any set of diagonal elements from an m × n matrix M. If offset O = 0, then the main diagonal is extracted. When O > 0, then the Oth superdiagonal is extracted. When O < 0, then the Oth subdiagonal is extracted. An example is given on the next page.

MATRIX OPERATIONS MENU { MATR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
$M = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$		B←M(M,0) = [1 6 11] B←B(M,2) = [3 8] B←M(M,-2) = [9]	
HP REDIMENSION	**RDM**	2: MATRIX 1: LIST	MATRIX
HP IDENTITY MATRIX	**IDN**	NUMBER OF ROWS	MATRIX
HP CONSTANT MATRIX	**CON**	2: LIST OR MATRIX 1: VALUE	MATRIX
See page 359 of the HP 48 owner's manual for complete descriptions.			
DIAGONAL MATRIX	**DIAG**(N,O)	$N \in \mathbb{N}$ $O \in \mathbb{I}$	MATRIX

DIAG creates an N × N matrix with ones on the Oth diagonal where $|O| \leq N - 1$.

$$\textbf{DIAG}(\ 4\ ,\ -1\ ) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

MATRIX OPERATIONS MENU { MATR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
COMPANION MATRIX	**COMP**(L)	L = { LIST }	MATRIX

COMP creates a companion matrix corresponding to list L of size not less than 3.

$$\textbf{COMP}(\{\ 8\ \ 4\ \ -6\ \ 2\ \}) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -4 & -2 & 3 \end{bmatrix}$$

Commands **TRNP**, **REV**↑, and **REV**→ will create the other companion matrix forms.

CREATE HERMITIAN TOEPLITZ MATRIX	**HTOEP**(V)	V = ANY HP REAL OR COMPLEX VECTOR	MATRIX

Given the first column V, **HTOEP** computes the corresponding Hermitian Toeplitz matrix. The first value of V must be real, and the size of V must be ≥ 2.

TEST IF λ IS EIGENVALUE	**λ?**(M,λ,ε) ε > 0	M = MATRIX λ = EIGENVALUE	1 IF TRUE 0 IF FALSE

λ? computes |DET(M – λI)| and compares the result with ε. Since ε is an input, **λ?** can be used to measure the accuracy of convergence of an eigenvalue calculation.

MATRIX OPERATIONS MENU
{ MATR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
TEST IF V IS EIGENVECTOR	**λV?**(M,λ,V,ε) ε > 0	M = MATRIX λ V ε	1 IF TRUE 0 IF FALSE

λV? computes ABS([M – λI]V) and compares the result with ε. Clearly, if λ is not an accurately computed eigenvalue, this test is meaningless. Use the **λ?** test first.

TEST IF VECTORS ARE THE SAME	**EVS?**(U,V,ε)	VECTORS U AND V ε > 0	1 IF TRUE 0 IF FALSE

Two vectors are the same if they have the same dimension and direction. **EVS?** normalizes U and V by the same method. It computes

$$| 1 - [U \bullet V^*]/ABS(U)/ABS(V)/SIGNP(U \bullet V^*) |$$

and compares the result with ε > 0. **EVS?** is useful in comparing eigenvector computations by different algorithms that result in different normalizations and rotations in the complex plane. **ESV?** can also be used to test the accuracy of convergence of an iterative vector calculation. U and V may be any HP 48 vector of the same length.

TEST IF MATRIX M IS DIAGONAL WITHIN ε > 0

DIAG?: « → M ε « M SIZE EVAL MIN → N « M N ZERO 0 B→M MABS MMAX ε < » »

UP DIRECTORY	**UPDIR**	NONE	PARENT MENU

22

STATISTICAL OPERATIONS AND TESTS

INTRODUCTION

This chapter presents the 48 statistical operation and test commands in the STAT menu. The STAT commands compute standard statistical parameters, such as the mean, and perform numerous statistical tests. Many other statistics related commands are given in the PROB, IPROB, BIVN, QUE, MISC, LINAG, MATR, VECTR, PROC, WIND, VSAG, MSAG menus.

DESCRIPTIVE STATISTICS

μ, **σ**, **ADEV**, **SKEW**, **KURT**, and **VARM** compute mean, sample standard deviation, absolute deviation, skewness, and kurtosis. **ROWμ**, **COLμ**, **ROWσ**, and **COLσ** perform mean and standard deviation matrix calculations.

MEASURES OF ASSOCIATION

TTSV, **TTDV**, and **TTPS** perform T tests for different means. **FTDV** performs an F test for different variances. **CDT1** and **CDT2** perform χ^2 tests for distributions. **KST1** and **KST2** perform Kolmogorov–Smirnov tests for distributions. **LCNT** computes the linear correlation coefficient. **SRCTT** performs the Spearman rank correlation T test.

CONTINGENCY TABLE ANALYSIS

CTA2D analyzes contingency tables and computes the degrees of freedom, contingency coefficient, χ^2, and the χ^2 probability of association.

ANALYSIS OF VARIANCE

AVAR1, **AVAR2**, and **ACOVR** perform one– and two–way analysis of variance and one–way analysis of covariance. These are used for testing of data. Analysis of variance commands associated with linear regression, curve and data fitting, and least squares approximation are given in the MISC menu.

MEDIAN AND MODE ESTIMATION

MEDIN and **MODE** estimate the median and mode of a data set.

HISTOGRAMS AND CUMULATIVE DISTRIBUTIONS

HIST computes a data histogram vector from the input data. It is normalized so that it is an estimate of the probability density function. **CUMΣ** will convert that histogram to a cumulative distribution function by computing the cumulative sum.

SORTING

SRT↑, **SRT↓**, and **SRTI** provide vector sorting. Matrix sorting commands are in the MATR menu.

RANDOM VECTORS

RNDU and **RNDN** provide random vectors with uniform and normal amplitude statistics. For random matrices, see the MSAG menu.

DATA SMOOTHING

MAVE provides moving averages of data.

UTILITIES

Numerous data manipulation commands are also provided.

COMPLEX ARGUMENTS

The **μ**, **σ**, **ADEV**, **VARM**, **COVAR**, **LCNT**, **CUMΣ**, **MAVE**, **VADD**, **VSUB**, **RNUM**, **CNUM**, **ROWμ**, **COLμ**, **ROWσ**, and **COLσ** commands do support complex arguments.

DATA ENTRY COMMENTS

Several of the commands take matrices as inputs. Generally a data set lies across a row. With the matrix editor and GO→ set, it is easy to enter your data. However, you may prefer to enter your data vertically with GO↓ set and use **TRNP** to transpose your data into input form. In this case, y is entered as the last column.

NONPARAMETRIC STATISTICS WITH RANKS

There are far too many tests to include them all on the ROM. The tricky part of tests involving ranks is computing the midranks. Program RANK at the end of the menu will do this for you. An example Mann–Whitney test is then presented as an example of how to use the **SRTI** and RANK outputs to perform various tests.

ADDITIONAL STATISTICAL TESTS

Several additional statistical test programs are given in Appendix A. They use commands from various menus on the ROM and involve some advanced programming techniques. They include the Wilcoxon signed rank test, the Kruskal–Wallis test, and the generalization of command **SRCTT**, Spearman's rank correlation test with repeated data and midranking.

SOFTWARE LIMITATIONS

Matrix commands generally assume a matrix with a minimum 2 × 2 size, and not row vectors of size 1 × n nor column vectors of size m × 1.

STATISTICAL OPERATIONS MENU { STAT }

FUNCTION	COMMAND	INPUTS	OUTPUTS
Note that *none* of the commands in this menu makes use of the ΣDAT variable or the numerous associated HP 48 commands. All of that capability can be used in *parallel* with the commands given below. See your HP 48 manual for details.			
MEAN	μ(V)	V = [VECTOR] ∈ ℂ	VALUE
$\mu = \frac{1}{N}\sum_{n=1}^{N} v_n \qquad N = SIZE(V) \geq 2$ μ([1 3 2 4 5]) = 3			
STANDARD DEVIATION	σ(V)	V = [VECTOR] ∈ ℂ	VALUE
$\sigma^2 = \frac{1}{N-1}\sum_{n=1}^{N} \lvert v_n - \mu \rvert^2 \qquad N = SIZE(V) \geq 2$ σ([1 3 2 4 5]) = 1.5811			
ABSOLUTE DEVIATION	ADEV(V)	V = [VECTOR] ∈ ℂ	VALUE
$ADEV = \frac{1}{N}\sum_{n=1}^{N} \lvert v_n - \mu \rvert \qquad N = SIZE(V) \geq 2$			

STATISTICAL OPERATIONS MENU { STAT }

FUNCTION	COMMAND	INPUTS	OUTPUTS
ADEV([1 3 2 4 5]) = 1.5			
SKEWNESS	**SKEW**(V)	V = [VECTOR] ∈ **R**	VALUE
$\mathbf{SKEW} = \frac{1}{N}\sum_{n=1}^{N}\left[\frac{v_n - \mu}{\sigma}\right]^3$ **N = SIZE(V) ≥ 2** SKEW([1 3 2 4 5]) = .634632			
KURTOSIS	**KURT**(V)	V = [VECTOR] ∈ **R**	VALUE
$\mathbf{KURT} = \left\{\frac{1}{N}\sum_{n=1}^{N}\left[\frac{v_n - \mu}{\sigma}\right]^4\right\} - 3$ **N = SIZE(V) ≥ 2** KURT([1 3 2 4 5]) = – 2.17388			
σ^2 AND μ	**VARM**(V)	V = [VECTOR] ∈ **C**	2: VARIANCE 1: MEAN
MEDIAN	**MEDIN**(V)	V = [VECTOR] ∈ **R**	VALUE

The MEDIAN is the argument x for which the cumulative distribution F(x) = ½. Input vector V is first sorted. The MEDIAN is then taken to be the middle value. If the sorted vector is V = [V_1 V_2 V_3 ... V_N], then the median is

STATISTICAL OPERATIONS MENU { STAT }

FUNCTION	COMMAND	INPUTS	OUTPUTS

$$V_{MED} = \begin{cases} V_{(N+1)/2} & N \text{ ODD} \\ \frac{1}{2}(V_{N/2}+V_{(N/2)+1}) & N \text{ EVEN} \end{cases}$$

MEDIN([1 5 4 2 2 4 3 3 3 3]) = MEDIN([1 2 2 3 3 3 3 4 4 5]) = 3

MODE	**MODE**(V,NB)	V = [VECTOR] ∈ **R**	2: [HISTOGRAM] 1: LIST

The MODE is the argument x for which the probability density function is a maximum. **MODE** uses **HIST** to sort and histogram the data based on the number of bins NB specified. That histogram is an estimate of the probability density function for the data. **MODE** then locates the position of the (first) maximum in that histogram L. In addition to the data histogram, **MODE** returns a list:

$$\left\{ L \quad \frac{MAX-MIN}{NB} \quad \frac{MAX-MIN}{NB} L+MIN \quad MIN \quad MAX \right\} = \{ L\ S\ LU\ MIN\ MAX \},$$

where L is the MODE, S is the data value step size which determines the location of the thresholds, MAX is the maximum value and MIN is the minimum value in the data. The range of data values corresponding to the estimated mode is defined by the interval: (LL = LU – S , LU].

For example, let V = [1 5 4 2 2 4 3 3 3 3] and NB = 5. Then the histogram is [.1 .2 .4 .2 .1] and the list is { 3 .8 3.4 1 5 }, so the MODE is 3 and the range of data values corresponding to that mode is (2.6 , 3.4]. Thresholds were set every .8 over the range 1 to 5.

STATISTICAL OPERATIONS MENU { STAT }

FUNCTION	COMMAND	INPUTS	OUTPUTS
COVARIANCE	**COVAR**(V1,V2)	[VECTORS] V1 and V2 ∈ ℂ	VALUE

$$\mathbf{COVAR} = \frac{1}{N-1}\sum_{n=1}^{N}[V_{1n} - \mu(V1)][V_{2n} - \mu(V2)]^* \qquad \mathbf{N = MIN(N_1, N_2)}$$

COVAR([1 3 4 5 6 8] , [2 3 4 5 7 9 10]) = 6.2

T TEST SAME VARIANCE	**TTSV**(V1,V2)	[VECTORS] V1 and V2 ∈ ℝ	2: t 1: UτPT(DF,t)

$$t = \frac{\mu(V1) - \mu(V2)}{\sqrt{\frac{(N_1-1)\sigma^2(V1) + (N_2-1)\sigma^2(V2)}{N_1 + N_2 - 2}\left(\frac{1}{N_1} + \frac{1}{N_2}\right)}} \qquad \mathbf{DF = N_1 + N_2 - 2}$$

Student's t–test for significantly different means when the two distributions are thought to have the same variance. Small values of probability indicate that V1 and V2 have different means.

TTSV([3 5 6 7 8 10] , [6 7 8 9 11 13 14]) = **2: -2.079**
1: 6.178E-2

T TEST DIFFERENT VAR	**TTDV**(V1,V2)	[VECTORS] V1 and V2 ∈ ℝ	2: t 1: UτPT(DF,t)

This command is discussed on the next page.

STATISTICAL OPERATIONS MENU { STAT }

FUNCTION	COMMAND	INPUTS	OUTPUTS

$$t = \frac{\mu(V1) - \mu(V2)}{\sqrt{\frac{\sigma^2(V1)}{N_1} + \frac{\sigma^2(V2)}{N_2}}} \qquad DF = \frac{\left[\frac{\sigma^2(V1)}{N_1} + \frac{\sigma^2(V2)}{N_2}\right]^2}{\frac{[\sigma^2(V1)/N_1]^2}{N_1 - 1} + \frac{[\sigma^2(V2)/N_2]^2}{N_2 - 1}}$$

Student's t–test for different means when the two distributions are thought to have significantly different variances. A small value of probability indicates that V1 and V2 have different means.

TTDV([3 5 6 7 8 10] , [6 7 8 9 11 13 14]) = { 2: -2.118 ; 1: 5.784E-2

T TEST PAIRED SAMPLES	**TTPS**(V1,V2)	[VECTORS] V1 and V2 ∈ **R**	2: t 1: UτPT(DF,t)

$$t = \frac{\mu(V1) - \mu(V2)}{\sqrt{\frac{\sigma^2(V1) + \sigma^2(V2) - 2COVAR(V1, V2)}{N}}} \qquad DF = N - 1$$

Student's t–test in the case of N paired samples, where N is the size of V1 and V2. A small value of probability indicates that V1 and V2 have different means. An infinite result error means the test is invalid for the data set.

TTPS([3 5 6 7 8 10] , [1 5 4 5 6 8]) = { 2: 5.000 ; 1: 4.105E-3

STATISTICAL OPERATIONS MENU { STAT }

FUNCTION	COMMAND	INPUTS	OUTPUTS
F TEST DIFFERENT VAR	**FTDV**(V1,V2)	[VECTORS] V1 and V2 $\in$ **R**	2: F 1: UτPF(DF_N,DF_D,F)

$DF_N = N_1 - 1$ $DF_D = N_2 - 1$ $F = \sigma^2(V1)/\sigma^2(V2)$ $\sigma(V1) \geq \sigma(V2)$

$DF_N = N_2 - 1$ $DF_D = N_1 - 1$ $F = \sigma^2(V2)/\sigma^2(V1)$ $\sigma(V1) \leq \sigma(V2)$

F test for significantly different variances. A small value of probability indicates that V1 and V2 have different variances.

FTDV([3 5 6 7 8 10] , [1 5 4 5 6 8]) = **2: 1.099**
1: 0.920

CHI-SQUARE DISTRIBUTION TEST	**CDT1**(V1,V,DF)	[VECTORS] V1 and V $\in$ **R** DF	2: χ^2 1: UτPC(DF,χ^2)

$$\chi^2 = \sum_{n=1}^{N} \frac{(V1_n - V_n)^2}{V_n} \qquad \textbf{DF = N - 1 if zero constraints}$$

Chi-square distribution test where $V1_n$ is the number of events observed in the nth bin, and V_n is the number expected according to some known distribution. N is the size of V1 and V. A small value of probability indicates that the distributions are different.

CDT1([3 5 6 7 8 10] , [1 5 4 5 6 8] , 5) = **2: 2.967**
1: 0.705

STATISTICAL OPERATIONS MENU { STAT }

FUNCTION	COMMAND	INPUTS	OUTPUTS
CHI-SQUARE DISTRIBUTION TEST	**CDT2**(V1,V2,DF)	[VECTORS] V1 and V2 $\in$ **R** DF	2: χ^2 1: UτPC(DF,χ^2)

$$\chi^2 = \sum_{n=1}^{N} \frac{(V1_n - V2_n)^2}{V1_n + V2_n}$$ **DF = N - 1 if zero constraints.**

Chi-square distribution test for comparing two data sets. N is the size of V1 and V2. A small value of probability indicates that the distributions are different.

CDT2([3 5 6 7 8 10] , [1 5 4 5 6 8] , 5) = **2: 1.241** **1: 0.941**

KOLMOGOROV–SMIRNOV TEST	**KST1**(V1,V)	[VECTORS] V1 and V $\in$ **R**	2: D 1: UTKS(√N D)

V1 is the estimated cumulative distribution function, and V is samples from the *a priori* cumulative distribution. N is the size of V1 and V. **HIST** and **CUMΣ** can be used to create these vectors.

$$D = MAX\ |V1_n - V_n| \quad n \in [1,N]$$

Small values of probability indicate that the cumulative distributions are different.

KST1([0 .1 .3 .5 .7 .9 1] , [0 .2 .4 .6 .8 .9 1]) = **2: 0.100** **1: 1.000**

STATISTICAL OPERATIONS MENU { STAT }

FUNCTION	COMMAND	INPUTS	OUTPUTS
KOLMOGOROV–SMIRNOV TEST	**KST2**(V1,V2)	[VECTORS] V1 and V2 ∈ **ℝ**	2: D 1: UTKS(√(N/2)D)

V1 and V2 are the estimated cumulative distribution function distributions. N is the size of V1 and V2. **HIST** and **CUMΣ** can be used to create these vectors.

$$D = \text{MAX}\ |V1_n - V2_n| \quad n \in [1,N]$$

Small values of probability indicate that the cumulative distributions are different.

KST2([0 .1 .3 .5 .7 .9 1] , [0 .01 .02 .03 .04 .1 1]) = **2: 0.800**, **1: 2.267E-2**

LINEAR CORRELATION	**LCNT**(V1,V2)	[VECTORS] V1 and V2 ∈ **ℂ**	CORRELATION COEFFICIENT r

$$r = \frac{\text{COVAR(V1, V2)}}{\sigma(V1)\,\sigma(V2)}$$

For normally distributed vectors V1 and V2, the distribution of r was first derived by Fisher in 1915, but is not well behaved and is difficult to compute. Letting N denote the minimum SIZE of V1 and V2, for N < 10 the approximate T statistic $T = r\sqrt{((N-2)/(1-r^2))}$ with N – 2 degrees of freedom is often used. Recognizing the problems with r, Fisher created the Fisher z–transform:

STATISTICAL OPERATIONS MENU { STAT }

FUNCTION	COMMAND	INPUTS	OUTPUTS

$$z = \frac{1}{2}\,LN\left(\frac{1+r}{1-r}\right) \qquad \bar{z} = \frac{1}{2}\left[LN\left(\frac{1+r_{true}}{1-r_{true}}\right) + \frac{r_{true}}{N-1}\right] \qquad \sigma(z) = \frac{1}{\sqrt{N-3}},$$

which is useful for $N \geq 10$. The significance level for which r differs from r_{true} and for which two measured values of r differ is given by the equations:

$$\text{erfc}\left(\frac{|z-\bar{z}|\sqrt{N-3}}{\sqrt{2}}\right) \qquad \text{erfc}\left(\frac{|z_1 - z_2|}{\sqrt{2}\sqrt{\frac{1}{N_1-3}+\frac{1}{N_2-3}}}\right),$$

where the complementary error function is available in the { FTNS ERROR } menu.

HISTOGRAM	**HIST**(V,NB)	V = [VECTOR] ∈ **R** NB = # OF BINS	2: [HISTOGRAM] 1: {MAX MIN}

HIST computes a histogram vector of length NB of the data in vector V. The histogram is normalized so that the sum of all the values is 1. In addition to the histogram, the maximum and minimum values are returned in a list. Use **CUMΣ** to create cumulative distribution.

HIST([1 5 4 2 2 4 3 3 3 3] , 5) = { 2: [.1 .2 .4 .2 .1]
1: { 5 1 }

SORT ↑	**SRT↑**(V)	V = [VECTOR] ∈ **R**	[VECTOR]

↑ = increasing value with increasing position [1 2 3].

SORT ↓	**SRT↓**(V)	V = [VECTOR] ∈ **R**	[VECTOR]

STATISTICAL OPERATIONS MENU { STAT }

FUNCTION	COMMAND	INPUTS	OUTPUTS
SORT ↑ WITH SORTED INDEX	**SRTI**(V)	V = [VECTOR] ∈ **R**	2: [VECTOR] 1: [INDEX]
SRT↑([2 5 3 7]) = [2 3 5 7] SRT↓([2 5 3 7]) = [7 5 3 2] SRTI([2 5 3 7]) = 2: [2 3 5 7] 1: [1 3 2 4] Suppose vector V originated as either a row or column in a matrix M. Index vector I is the required vector for sorting matrix M according to row (column) V by using **RORDR** or **CORDR**.			
RANDOM VECTOR WITH UNIFORM AMPLITUDE STATISTICS	**RNDU**(μ,σ,s,N)	μ = MEAN σ = STAND DEV s = SEED N = SIZE	RANDOM [VECTOR]
RANDOM VECTOR WITH NORMAL AMPLITUDE STATISTICS	**RNDN**(μ,σ,s,N)	μ = MEAN σ = STAND DEV s = SEED N = SIZE	RANDOM [VECTOR]
RNDN(1,2,11,8) creates an eight dimensional vector with elements that are normally distributed with mean value of 1 and standard deviation of 2.			
CUMULATIVE SUM OF VALUES	**CUM**Σ(V)	V = [VECTOR] ∈ **C** SIZE = N	V = [VECTOR] SIZE = N + 1
CUMΣ([.1 .2 .4 .2 .1]) = [0 .1 .3 .7 .9 1.]			

STATISTICAL OPERATIONS MENU { STAT }

FUNCTION	COMMAND	INPUTS	OUTPUTS
MOVING AVERAGE	**MAVE**(V,n)	V = [VECTOR] $\in \mathbb{C}$	[VECTOR]
Performs a moving average on the data using n points in each average. If less than n points are available, then it uses the points that are available. Use for data smoothing. MAVE([1 3 1 3 1 3 1 3 1 3] , 2) = [1 2 2 2 2 2 2 2 2 2]			
VECTOR ADD SCALAR	**VADD**(V,s)	V = [VECTOR] $\in \mathbb{C}$ s = SCALAR	[VECTOR] $V_j + s$
VADD([1 2 3 4] , 2) = [3 4 5 6] VSUB([3 4 5 6] , 2) = [1 2 3 4]			
VECTOR SUB SCALAR	**VSUB**(V,s)	V = [VECTOR] $\in \mathbb{C}$ s = SCALAR	[VECTOR] $V_j - s$
CONTINGENCY TABLE ANALYSIS OF TWO DISTRIBUTIONS	**CTA2D**(M)	CONTINGENCY MATRIX M $\in \mathbb{R}$	5: N 4: DF 3: χ^2 2: UτPC(DF,χ^2) 1: C

CTA2D first eliminates any rows and columns for which the row (column) sum is zero. Denoting by R_j and C_k these sums, and letting N equal the sum over j of R_j (or over k of C_k) and defining $n_{jk} = R_j C_k / N$:

$$\chi^2 = \sum_j \sum_k \frac{(M_{jk} - n_{jk})^2}{n_{jk}} \qquad C = \sqrt{\frac{\chi^2}{\chi^2 + N}} \qquad DF = (J-1)(K-1),$$

STATISTICAL OPERATIONS MENU { STAT }

FUNCTION	COMMAND	INPUTS	OUTPUTS

where C is the contingency coefficient. DF is the degrees of freedom. J and K are the row and column dimensions of M after deletion of all rows and columns whose sums are zero. A small value of probability **UτPC** indicates a significant association. Consider the following contingency table represented by the matrix

$$M = \begin{bmatrix} 40 & 10 & 2 & 60 \\ 8 & 35 & 50 & 5 \\ 10 & 15 & 8 & 14 \end{bmatrix}$$

Then N = 257, DF = 6, χ^2 = 130.7099416, UτPC = 9.10987391862E–26, and C = .580631832281.

SPEARMAN RANK CORRELATION T TEST	**SRCTT**(V1,V2)	[VECTORS] V1 AND V2 ∈ **R**	4: r 3: DF 2: t 1: UτPT(DF,t)

Each of the input vectors is sorted and ranked. The index output of **SRTI** is used for that rank, and midranks are not used. See Appendix A for a more general version of this test for data sets with repeated values where midranking is used. r is the correlation coefficient of those ranks computed by **LCNT**, DF is the degrees of freedom, and t is the t statistic. A small value of probability **UτPT** indicates significant correlation. $t = r\sqrt{(DF/(1 - r^2))}$ and DF = N – 2, where N is the size of V1 and V2. For example, let:

V1 = [82 67 91 98 74 52 86 95 79 78 84 80 69 81 73]
V2 = [81 75 85 90 80 60 94 78 83 76 84 69 72 88 61].

Then r = .7607142857, DF = 13, t = 4.225619798, UτPT = 9.91128612E–4.

STATISTICAL OPERATIONS MENU { STAT }

FUNCTION	COMMAND	INPUTS	OUTPUTS
ANALYSIS OF VARIANCE (ONE–WAY)	**AVAR1**(M)	M = MATRIX $\in$ **R** SIZE(M) $\geq$ {2 2}	2: MATRIX 1: UτPF(DF1,DF2, F)

The one–way analysis of variance is used to test if observed differences among r sample means can be attributed to chance or whether they are indicative of actual differences among the corresponding population means. **AVAR1** assumes that each row of M is a set of observations whose means and standard deviations can be computed using **ROWμ** and **ROWσ** below. **AVAR1** computes all the values of the complete ANOVA table and **UτPF**. The table is the matrix

$$\begin{matrix} \textbf{Treatments :} \\ \textbf{Within treatments :} \\ \textbf{Total :} \end{matrix} \begin{bmatrix} \textbf{TrSS} & \textbf{DF1} & \textbf{TrMS} \\ \textbf{ESS} & \textbf{DF2} & \textbf{EMS} \\ \textbf{TSS} & \textbf{Total} & \textbf{F} \end{bmatrix} .$$

The total sum of squares TSS is

$$\textbf{TSS} = \sum_{j=1}^{r}\sum_{k=1}^{n_j} M_{jk}^2 - \frac{\left(\sum_{j=1}^{r}\sum_{k=1}^{n_j} M_{jk}\right)^2}{\sum_{j=1}^{r} n_j}$$

$$\textbf{DF1} = r - 1$$

$$\textbf{DF2} = \sum_{j=1}^{r} n_j - r$$

$$\textbf{ESS} = \textbf{TSS} - \textbf{TrSS} .$$

Pathological values can result in divide-by-zero errors. Slightly change the input data values, if this occurs.

STATISTICAL OPERATIONS MENU { STAT }

FUNCTION	COMMAND	INPUTS	OUTPUTS

The treatment sum of squares TrSS is

$$\text{TrSS} = \sum_{j=1}^{r} \frac{\left(\sum_{k=1}^{n_j} M_{jk}\right)^2}{n_j} - \frac{\left(\sum_{j=1}^{r}\sum_{k=1}^{n_j} M_{jk}\right)^2}{\sum_{j=1}^{r} n_j}$$

$$\text{TrMS} = \frac{\text{TrSS}}{\text{DF1}} \qquad \text{EMS} = \frac{\text{ESS}}{\text{DF2}} \qquad F = \frac{\text{TrMS}}{\text{EMS}} .$$

n_j is the number of observations in the jth sample of M computed by **RNUM** below, r is the number of rows of M, ESS is the error sum of squares, DF1 is the treatment degrees of freedom, and DF2 is the error degrees of freedom. For example, consider the below scores obtained from four schools:

	1	2	3	4	5	6	7
School 1	88	99	96	68	85		
School 2	78	62	98	83	61	88	
School 3	80	61	74	92	78	54	77
School 4	71	65	90	46			

AVAR1
$$\begin{pmatrix} 88 & 99 & 96 & 68 & 85 & 0 & 0 \\ 78 & 62 & 98 & 83 & 61 & 88 & 0 \\ 80 & 61 & 74 & 92 & 78 & 54 & 77 \\ 71 & 65 & 90 & 46 & 0 & 0 & 0 \end{pmatrix} =$$

2:
$$\begin{bmatrix} 930.44 & 3 & 310.15 \\ 3599.56 & 18 & 199.98 \\ 4530 & 21 & 1.55 \end{bmatrix}$$

1: .2358

A small value of **UτPF** indicates that the means are significantly different. Since .2358 is not very small, we cannot conclude that the means are significantly different. However,

STATISTICAL OPERATIONS MENU { STAT }

FUNCTION	COMMAND	INPUTS	OUTPUTS

AVAR1$\left(\begin{bmatrix} 1 & 2 & 1 & 2 \\ 90 & 80 & 90 & 80 \end{bmatrix}\right)$ =

2: $\begin{bmatrix} 13944.5 & 1 & 13944.5 \\ 101 & 6 & 16.833 \\ 14045.5 & 7 & 828.386 \end{bmatrix}$

1: 1.165E-7

The zeros of each row are regarded as not being data. If your data contains many zeros, you may get divide-by-zero errors. Enter your *true zero* values as small numbers. **AVAR1** uses the **RNUM** command.

ANALYSIS OF VARIANCE (TWO-WAY)	**AVAR2**(M)	M = MATRIX ∈ **R** SIZE(M) ≥ {2 2}	3: MATRIX 2: UτPF(DF1,DF3, F1) 1: UτPF(DF2,DF3, F2)

The two-way analysis of variance examines variability amongst rows and columns of the matrix M. Each is considered independently. It is assumed that each element of M represents one observation and that row and column effects do not interact. **AVAR2** computes all the values of the ANOVA table plus both the row and column F distribution probabilities. The table is the matrix

$$\begin{array}{rl} \textbf{Row sum of squares :} & \begin{bmatrix} \text{RSS} & \text{DF1} & \text{F1} \\ \text{CSS} & \text{DF2} & \text{F2} \\ \text{ESS} & \text{DF3} & \text{TSS} \end{bmatrix} \\ \textbf{Column sum of squares :} & \\ \textbf{Error sum of squares :} & \end{array}$$

Pathological values can result in divide-by-zero errors. Slightly change the input data values, if this occurs.

STATISTICAL OPERATIONS MENU { STAT }

FUNCTION	COMMAND	INPUTS	OUTPUTS

$$RSS = \sum_{j=1}^{r}\left(\sum_{k=1}^{c} M_{jk}\right)^2/c - S \qquad TSS = \sum_{j=1}^{r}\sum_{k=1}^{c} M_{jk}^2 - S \qquad SS1 = \frac{RSS}{DF1}$$

$$CSS = \sum_{k=1}^{c}\left(\sum_{j=1}^{r} M_{jk}\right)^2/r - S \qquad ESS = TSS - RSS - CSS \qquad SS2 = \frac{CSS}{DF2}$$

$$S = \left(\sum_{j=1}^{r}\sum_{k=1}^{c} M_{jk}\right)^2/rc \qquad SS3 = \frac{ESS}{DF3}$$

The degrees of freedom are DF1 = r – 1, DF2 = c – 1, and DF3 = (r – 1)(c – 1). The row F ratio is F1 = SS1/SS3, and the column F ratio is F2 = SS2/SS3. For example:

$$\text{AVAR2}\left(\begin{bmatrix}7 & 6 & 8 & 7\\ 2 & 4 & 4 & 3\\ 4 & 6 & 5 & 3\end{bmatrix}\right) = \begin{matrix}3: \begin{bmatrix}29.167 & 2 & 15.909\\ 4.25 & 3 & 1.545\\ 5.5 & 6 & 38.917\end{bmatrix}\\ 2: \ 3.993\text{E}{-3}\\ 1: \ 0.297\end{matrix}$$

Now the small value of the row probability indicates a significant difference in the means of each row, while the not very small column probability 0.297 indicates that nothing can be concluded with respect to the means of the columns. In other words, the variances of each row are small relative to the differences in means between rows, whereas the variances of each column are large relative to the differences in means between columns.

STATISTICAL OPERATIONS MENU { STAT }

FUNCTION	COMMAND	INPUTS	OUTPUTS
ANALYSIS OF COVARIANCE (ONE-WAY)	**ACOVR**(M)	M = MATRIX $\in$ **R** SIZE(M) $\geq$ {4 2}	2: MATRIX 1: UτPF(DF3,DF4, F)

The **AVAR1** command dealt with testing the results of treatments without considering the statistics of that which was treated. **ACOVR** considers both. We denote the data before treatment as x_{jk} for k = 1, 2, . . ., n_j and the data after treatment as y_{jk} for k = 1, 2, . . ., n_j and for j = 1, 2, . . ., r where r is the number of groups. The **ACOVR** input matrix consists of 2r rows of data with the x and y data sets interleaved. The output is the complete ANOCOV table in the form of a matrix and **UτPF**.

$$\begin{array}{r} \textbf{Treatments :} \\ \textbf{Within treatments :} \\ \textbf{Total :} \end{array} \begin{bmatrix} \textbf{DF1} & \textbf{ASSx} & \textbf{ASP} & \textbf{ASSy} & \textbf{DF3} & \textbf{ASS}\hat{\textbf{y}} & \textbf{AMS}\hat{\textbf{y}} \\ \textbf{DF2} & \textbf{WSSx} & \textbf{WSP} & \textbf{WSSy} & \textbf{DF4} & \textbf{WSS}\hat{\textbf{y}} & \textbf{WMS}\hat{\textbf{y}} \\ \textbf{Total} & \textbf{TSSx} & \textbf{TSP} & \textbf{TSSy} & \textbf{Total} & \textbf{TSS}\hat{\textbf{y}} & \textbf{F} \end{bmatrix}$$

Treatments is sometimes called among groups and within treatments called within groups.

Matrix M can be expressed in terms of its X and Y rows, using the commands defined below:

M = RNLV(RCMB(MX , MY)) **RSPLT(RDLV(M) , r) =** $\begin{cases} \textbf{2: MX} \\ \textbf{1: MY} \end{cases}$

Now ASSx, WSSx, and TSSx are the TrSS, ESS, and TSS outputs of AVAR1(MX), and ASSy, WSSy, and TSSy are the TrSS, ESS, and TSS outputs of AVAR1(MY), respectively. Similarly, DF1 and DF2 are the DF1 and DF2 outputs of both AVAR1(MX) and AVAR1(MY). Equations for the other outputs are given on the next page.

STATISTICAL OPERATIONS MENU { STAT }

FUNCTION	COMMAND	INPUTS	OUTPUTS

$$TSP = \sum_{j=1}^{r}\sum_{k=1}^{n_j} x_{jk}\, y_{jk} - S$$

$$ASP = \sum_{j=1}^{r} \frac{\left(\sum_{k=1}^{n_j} x_{jk}\right)\left(\sum_{k=1}^{n_j} y_{jk}\right)}{n_j} - S$$

$$WSP = TSP - ASP$$

$$TSS\hat{y} = TSSy - \frac{(TSP)^2}{TSSx}$$

$$WSS\hat{y} = WSSy - \frac{(WSP)^2}{WSSx}$$

$$ASS\hat{y} = TSS\hat{y} - WSS\hat{y}$$

$$AMS\hat{y} = \frac{ASS\hat{y}}{DF3}$$

$$WMS\hat{y} = \frac{WSS\hat{y}}{DF4}$$

$$F = \frac{AMS\hat{y}}{WMS\hat{y}}$$

$$S = \frac{\sum_{j=1}^{r}\sum_{k=1}^{n_j} x_{jk} \sum_{j=1}^{r}\sum_{k=1}^{n_j} y_{jk}}{\sum_{j=1}^{r} n_j},$$

$$DF3 = r - 1, \qquad DF4 = \sum_{j=1}^{r} n_j - r - 1.$$

Consider the following example where [11 9 5 8 12] is the first set of Y data and [4 5 2 7 6] is the third set of X data. Use **RNLV** and **RCMB** to put your data into the required interleaved form.

$$ACOVR\left(\begin{bmatrix} 5 & 3 & 1 & 4 & 6 \\ 11 & 9 & 5 & 8 & 12 \\ 2 & 6 & 4 & 7 & 3 \\ 1 & 7 & 3 & 8 & 2 \\ 4 & 5 & 2 & 7 & 6 \\ 7 & 9 & 5 & 13 & 11 \end{bmatrix}\right) = \begin{cases} 2: \begin{bmatrix} 2 & 2.53 & -1.6 & 76.8 & 2 & 86.578 & 43.289 \\ 12 & 46.8 & 69.6 & 108.8 & 11 & 5.292 & .481 \\ 14 & 49.333 & 68 & 185.6 & 13 & 91.870 & 89.976 \end{bmatrix} \\ 1: \ 1.523E{-}7 \end{cases}$$

The small value of probability indicates there is a significant difference in means. If **AVAR1** is applied to just the Y data sets, the result is

STATISTICAL OPERATIONS MENU { STAT }

FUNCTION	COMMAND	INPUTS	OUTPUTS

AVAR1$\left(\begin{bmatrix} 11 & 9 & 5 & 8 & 12 \\ 1 & 7 & 3 & 8 & 2 \\ 7 & 9 & 5 & 13 & 11 \end{bmatrix}\right) = \begin{cases} 2: \begin{bmatrix} 76.8 & 2 & 38.4 \\ 108.8 & 12 & 9.067 \\ 185.6 & 14 & 4.235 \end{bmatrix} \\ 1: \ 4.058\text{E}{-2} \end{cases}$

Observe that the value of F is larger and the value of **UτPF** is smaller for the **ACOVR** test than the **AVAR1** test. Thus, use of the X data can improve the reliability of the test.

When the size of the data sets is not equal, the rows may be zero–filled. Observe the comments given with the **AVAR1** command concerning zero values in M.

ROW NUMBER	**RNUM**(M)	M = MATRIX ∈ ℂ	[VECTOR]
COLUMN NUMBER	**CNUM**(M)	M = MATRIX ∈ ℂ	[VECTOR]

RNUM and **CNUM** are useful when analyzing data sets of unequal length which have been entered as either rows or columns of a matrix where the nonexistent items of each set are entered as zeros in the rows or columns of matrix M. These commands return the number of items in each row (column) excluding the zeros. True zero values should be entered as small numbers.

LIST ROUND	**SRND**(L,N)	L N	ROUNDED LIST

Given list L of numbers and integer N, **SRND** applies ≪ N RND ≫ to each element of L.

ROW MEAN	**ROWμ**(M)	M = MATRIX ∈ ℂ	[VECTOR]

STATISTICAL OPERATIONS MENU { STAT }

FUNCTION	COMMAND	INPUTS	OUTPUTS
COLUMN MEAN	**COLμ**(M)	M = MATRIX ∈ ℂ	[VECTOR]
ROWμ and **COLμ** are useful when analyzing data sets of unequal length that have been entered as either rows or columns of a matrix where the nonexistent items of each set are entered as zeros in the rows or columns of matrix M. These commands compute the row (column) sums and normalize that sum by the number of nonzero items. See **RSUM** and **CSUM** in the MSAG menu.			
ROW STANDARD DEVIATION	**ROW**σ(M)	M = MATRIX ∈ ℂ	[VECTOR]
COLUMN STANDARD DEVIATION	**COL**σ(M)	M = MATRIX ∈ ℂ	[VECTOR]
ROWσ and **COL**σ are useful when analyzing data sets of unequal length that have been entered as either rows or columns of a matrix where the nonexistent items of each set are entered as zeros in the rows or columns of matrix M. The standard deviation formula given on page 274 is applied to each row (column) where the means are computed by **ROWμ** (**COLμ**) and N is computed by **RNUM** (**CNUM**).			
VECTOR SPLIT	**VSPLT**(V,N)	V N	TWO VECTORS
Given vector V and 0 < N < size of V, **VSPLT** divides V into two vectors. N is the size of the first. Given vectors V1 and V2, **VCMB** splices them into a single vector [V1 V2].			
VECTOR COMBINE	**VCMB**(V1,V2)	TWO VECTORS	ONE VECTOR

STATISTICAL OPERATIONS MENU { STAT }

FUNCTION	COMMAND	INPUTS	OUTPUTS
ROW INTERLEAVE	**RNLV**(M)	M = MATRIX	MATRIX

Given matrix M with an even number of rows, this command interleaves the second half of the matrix with the first half. For example:

$$\mathbf{M} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix} \qquad \mathbf{RNLV(M)} = \begin{bmatrix} 1 & 2 & 3 \\ 7 & 8 & 9 \\ 4 & 5 & 6 \\ 10 & 11 & 12 \end{bmatrix}$$

ROW DE-INTERLEAVE	**RDLV**(M)	M = MATRIX	MATRIX

Given matrix M with an even number of rows, this command de-interleaves each pair of two rows, thus providing the inverse transformations corresponding to **RNLV**.

ROW SPLIT	**RSPLT**(M,R)	M = MATRIX R	TWO MATRICES

Given matrix M, this command splits M into two matrices; the first having the first R rows of the original matrix, and the second having the remainder.

$$\mathbf{M} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix} \qquad \mathbf{RSPLT(M,2)} = \begin{cases} 2: \begin{bmatrix} 1 & 3 & 2 & 4 \\ 5 & 7 & 6 & 8 \end{bmatrix} \\ 1: [9 \;\; 11 \;\; 10 \;\; 12] \end{cases}$$

STATISTICAL OPERATIONS MENU { STAT }

FUNCTION	COMMAND	INPUTS	OUTPUTS
ROW COMBINE	**RCMB**(M1,M2)	TWO MATRICES	MATRIX
Given matrices M1 and M2, this command performs the inverse transformation corresponding to **RSPLT** of combining two matrices into one.			
HP SIZE	SIZE SIZE	[VECTOR] { LIST }	{ SIZE } SIZE

NONPARAMETRIC STATISTICS AND COMPUTING RANKS

Program RANK computes the ranks with midranking of sorted vector V.

```
RANK: ≪ → V ≪ V SIZE EVAL 1 0 → N J K ≪ WHILE 'J<N' REPEAT
IF 'V(J+1)≠V(J)' THEN J 1 'J' STO+ ELSE J 2 + 'K' STO WHILE
'V(MIN(N,K))==V(J) AND K≤N' REPEAT 1 'K' STO+ END J K + 1 - 2 / J
K 2 - START DUP NEXT K 'J' STO END END IF 'J==N' THEN N END
N →ARRY ≫ ≫ ≫
```

As an example of using **SRTI** and RANK, consider the Mann–Whitney test on two independent random samples. MWT computes the t statistic. See page 1018 of Pfaffenberger and Patterson for this example. Let V1 = [55 70 70 65 62 81 72 58 67 50] and V2 = [50 91 90 62 75 88 84 78 82 80]. t = MWT(V1,V2).

```
≪ → V1 V2 ≪ V1 V2 VCMB SRTI SWAP RANK V1 SIZE EVAL → I V
N ≪ 0 1 N FOR K 'V' 'I' K GET GET + NEXT N N 1 + × 2 / - N V2
SIZE EVAL ROT ≫ ≫ ≫
```

The sorted output of **SRTI** is the input to RANK. The index output of **SRTI** is used to index the appropriate values in the RANK output to be summed. t = 19. Other tests can be performed with similar programs. See Appendix A for more examples.

STATISTICAL OPERATIONS MENU { STAT }

FUNCTION	COMMAND	INPUTS	OUTPUTS
UP DIRECTORY	UPDIR	NONE	PARENT MENU

Allen, A., *Probability, Statistics, and Queueing Theory*, Boston, Academic Press, 1990.

Anderson, T., *An Introduction to Multivariate Statistical Analysis*, New York, Wiley, 1958.

Cohen, J., *Statistical Power Analysis for the Behavioral Sciences*, Hillsdale: N.J., Lawrence Erlbaum Assoc., 1988.

Cramer, H., *Mathematical Methods of Statistics*, Princeton, Princeton Univ. Press, 1971.

Draper, N., and Smith, H., *Applied Regression Analysis*, New York, Wiley, 1966.

Edwards, A., *Multiple Regression and the Analysis of Variance and Covariance*, San Francisco, Freeman, 1979.

Mason, R., and Lind, D., *Statistical Techniques in Business and Economics*, Homewood: Ill, Irwin, 1990.

Pfaffenberger, R., and Patterson, J., *Statistical Methods*, Homewood: Ill, Irwin, 1987.

Press, W., Flannery, B., Teukolsky, S., and Vetterling, W., *Numerical Recipes*, New York, Cambridge Univ. Press, 1989.

Raiffa, H., and Schlaifer, R., *Applied Statistical Decision Theory*, Cambridge, M.I.T. Press, 1961.

Rao, C., *Linear Statistical Inference and Its Applications*, New York, Wiley, 1965.

Scheffe, H., *The Analysis of Variance*, New York, Wiley, 1959.

Zacks, S., *The Theory of Statistical Inference*, New York, Wiley, 1971.

23

PROBABILITY DISTRIBUTIONS

INTRODUCTION

This chapter presents the 57 probability distribution commands in the PROB, IPROB, and BIVN menus. The PROB menu contains 27 of the more common univariate probability distributions used in statistics and engineering. To aid in the construction of hypothesis-testing thresholds and receiver operating characteristic curves (ROC), 19 inverse probability distribution functions are given in the IPROB menu. The BIVN menu provides four Bivariate Normal distribution commands.

PROB MENU

Definitions for each distribution are provided to avoid all possible confusion.

Chi-square	F distribution	Normal
T distribution	Exponential	Weibull
Extreme value	Uniform	Cauchy
Laplace	Gamma	Beta
Non-central χ^2	Non-central F	Non-central T
Binomial	Hypergeometric	Negative binomial
Poisson	Kolmogorov–Smirnov	Rayleigh
Rician	Marcum Γ_N	Marcum P_N
Combinations	Permutations	Factorial

IPROB MENU

The inverse probability functions provide direct computation of confidence limits and intervals. Where closed form inverse distribution functions exist, they are used. Where they do not, the HP 48 root solver is used for the computation.

Chi-square	F distribution	Normal
T distribution	Exponential	Weibull
Extreme value	Uniform	Cauchy
Laplace	Gamma	Beta
Binomial	Negative binomial	Poisson
Kolmogorov–Smirnov	Rayleigh	Marcum Γ_N

BIVN MENU

All combinations of upper tail and lower tail probabilities are given for the bivariate normal distribution.

ADDITIONAL PROBABILITY DISTRIBUTIONS

There are a few distributions that are not on the ROM. These include log–normal, Maxwell, sech–square, Fermi–Dirac, and Bose–Einstein. They are available along with a tabulation of means, variances, density functions, and characteristic functions for the ROM distributions. See Appendix A.

MATHEMATICAL PROBABILITY DISTRIBUTIONS MENU { PROB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
UPPER TAIL CHI-SQUARE	**UτPC**(v,χ^2)	v = DEGREES OF FREEDOM (DF)	VALUE

$$Q(\chi^2|v) = [2^{v/2}\,\Gamma(v/2)]^{-1}\int_{\chi^2}^{\infty} t^{v/2-1}e^{-t/2}\,dt \qquad \chi^2 \geq 0 \qquad v > 0$$

When v is an integer n, then **UTPC**(n,χ^2) = **UτPC**(n,χ^2).

UPPER TAIL F DISTRIBUTION	**UτPF**(v_N,v_D,F)	v_N and v_D are DF	VALUE

$$Q(F|v_N,\,v_D) = \frac{v_N^{v_N/2}\,v_D^{v_D/2}}{B\left(\frac{1}{2}v_N,\,\frac{1}{2}v_D\right)}\int_F^{\infty} t^{(v_N-2)/2}\,(v_D + v_N t)^{-(v_N+v_D)/2}\,dt \qquad F \geq 0 \qquad v_N,\,v_D > 0$$

When v_N and v_D are integers, then **UTPF** = **UτPF**.

UPPER TAIL NORMAL	**UτPN**(μ,σ,x)	μ = MEAN, σ = STANDARD DEV	VALUE

$$P\{X \geq x\} = \int_x^{\infty} \frac{1}{\sigma\sqrt{2\pi}}\,\mathrm{EXP}\left(\frac{-(t-\mu)^2}{2\sigma^2}\right)dt \qquad \sigma > 0$$

UTPN = **UτPN** always, but **UτPN** is algebraic.

MATHEMATICAL PROBABILITY DISTRIBUTIONS MENU { PROB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
UPPER TAIL T DISTRIBUTION	**UτPT**(v,t)	v = DF	VALUE

$$1 - \mathbf{A}(t|v) = 1 - \left[\sqrt{v}\ \mathbf{B}\left(\frac{1}{2}, \frac{v}{2}\right)\right]^{-1} \int_{-t}^{t} \left(1 + \frac{x^2}{v}\right)^{-\frac{v+1}{2}} dx \qquad t \geq 0 \quad v > 0$$

UτPT(v, t) = **UτPT**(v, |t|) $\quad t < 0.$

When v is an integer n, then **UTPT**(n, t) = .5 **UτPT**(n, t) $\quad t \geq 0.$

LOWER TAIL NORMAL	**LTPN**(μ,σ,x)	μ = MEAN, σ = STANDARD DEV	VALUE

LTPN(μ, σ, x) = 1 – **UτPN**(μ, σ, x)

UPPER TAIL EXPONENTIAL	**UTPE**(μ,σ,x)	μ = MEAN, σ = STANDARD DEV	VALUE

EXP(–(x – μ)/σ – 1) $\quad x \geq \mu - \sigma \quad \sigma > 0$

UPPER TAIL WEIBULL	**WEIB**(c,x)	Parameter c	VALUE

$P\{\mathbf{X} \geq x\} = EXP(-x\text{^}c) \qquad x \geq 0 \quad c > 0$

$\mu = \Gamma(c^{-1} + 1) \quad \sigma^2 = \Gamma(2c^{-1} + 1) - \mu^2$

The substitution $x = (y - \xi_0)/\alpha \quad \alpha > 0$ is common.

MATHEMATICAL PROBABILITY DISTRIBUTIONS MENU { PROB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
UPPER TAIL EXTREME VALUE	**EXTV**(μ,σ,x) TYPE I	μ = MEAN, σ = STANDARD DEV	VALUE
The Type I distribution is commonly written as $P\{\mathbf{X} \geq x\} = 1 - EXP\{ e^{-(x-\xi)/\theta} \}$ and can be expressed in terms of μ and σ as $1 - EXP(- EXP[-(x-\mu)\pi / (\sigma\sqrt{6}) - \gamma])$ $\sigma > 0$.			
UPPER TAIL UNIFORM	**UNIF**(μ,σ,x)	μ = MEAN, σ = STANDARD DEV	VALUE
$(\mu + \sqrt{(3\sigma^2)} - x) / \sqrt{(12\sigma^2)}$ $\quad |x-\mu| \leq \sqrt{(3\sigma^2)}$ $\quad \sigma > 0$			
UPPER TAIL CAUCHY	**CAUCH**(θ,α,x)	Parameters θ α	VALUE
$.5 - ATAN[(x - \theta)/\alpha]/\pi$ $\quad \alpha > 0$			
UPPER TAIL LAPLACE	**LAPL**(μ,σ,x)	μ = MEAN, σ = STANDARD DEV	VALUE
$\begin{cases} EXP[\sqrt{2}(\mu - x)/\sigma]/2 & x \geq \mu \quad \sigma > 0 \\ 1 - EXP[\sqrt{2}(x - \mu)/\sigma]/2 & x < \mu \quad \sigma > 0 \end{cases}$			

MATHEMATICAL PROBABILITY DISTRIBUTIONS MENU { PROB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
UPPER TAIL GAMMA (ERLANG)	**UTPG**(a,x)	Parameter a	VALUE
$\Gamma(a, x)/\Gamma(a) \quad x \geq 0 \quad a > 0$ Substitutions like $x = (y - \alpha)/\beta$ are common. With $a = \mu K$, $\mu > 0$, $K > 0$, this is also called the Erlang–K distribution.			
UPPER TAIL BETA	**UTP**β(a,b,x)	Parameters a b	VALUE
$I_{1-x}(b, a) \quad a, b > 0 \quad 0 \leq x \leq 1$ Substitutions like $x = (y - \alpha)/(\beta - \alpha)$ are common.			
LOWER TAIL BETA	**LTP**β(a,b,x)	Parameters a b	VALUE
LTPβ(a, b, x) = 1 − **UTP**β(a, b, x) = $I_x(a, b) \quad a,b > 0$ with x real, LTPβ(a, b, x) = 0 for $x \leq 0$ and LTPβ(a, b, x) = 1 for $x \geq 1$.			
UPPER TAIL NON-CENTRAL CHI-SQUARE	**U**τη**C**(λ,v,χ^2)	λ is non-centrality parameter	VALUE

MATHEMATICAL PROBABILITY DISTRIBUTIONS MENU { PROB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
$Q(\chi^2 \mid v, \lambda) = \sum_{n=0}^{\infty} \frac{(\lambda/2)^n}{n!} e^{-\lambda/2} Q(\chi^2 \mid v + 2n) \quad \chi^2 \geq 0 \quad v > 0 \quad \lambda \geq 0$			
UPPER TAIL NON-CENTRAL F DISTRIBUTION	**UτηF**(λ,v_N,v_D,F)	λ is non-centrality parameter	VALUE
$Q(F \mid v_N, v_D, \lambda) = \sum_{n=0}^{\infty} \frac{(\lambda/2)^n}{n!} e^{-\lambda/2} Q(F \mid v_N + 2n, v_D) \quad F \geq 0 \quad v_N, v_D > 0 \quad \lambda \geq 0$			
UτηC, **UτηF**, and **UτηT** evaluate the shown power series so accuracy degrades with large non-centrality. The closeness to 1 for zero arguments is a measure of this accuracy.			
UPPER TAIL NON-CENTRAL T DISTRIBUTION	**UτηT**(δ,v,T)	δ is non-centrality parameter	VALUE
$Q(T \mid v, \delta) = \sum_{n=0}^{\infty} \frac{(\delta^2/2)^n}{2n!} e^{-\delta^2/2} I_x\left(\frac{v}{2}, \frac{1}{2} + n\right) \quad x = \frac{v}{v+T^2} \quad T \geq 0 \quad v > 0 \quad \delta \geq 0$			

MATHEMATICAL PROBABILITY DISTRIBUTIONS MENU { PROB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
UPPER TAIL BINOMIAL	**BINM**(N,n,p)	N n p	VALUE

$$\sum_{s=n}^{N} \binom{N}{s} p^s (1-p)^{N-s} \qquad 0 \le n \le N \qquad n, N \in \mathbb{N} \qquad 0 < p < 1$$

LOWER TAIL HYPER-GEOMETRIC	**LTPH**(n,x,N,c)	x n N c	VALUE

$$\sum_{k=0}^{c} \frac{\binom{x}{k}\binom{N-x}{n-k}}{\binom{N}{n}} \qquad 0 \le c \le \text{MIN}(x,n) \qquad n + x \le N \qquad n, x, N, c \in \mathbb{N}$$

x and n are interchangeable.

UPPER TAIL NEGATIVE BINOMIAL	**NEGB**(n,x,p)	n x p	VALUE

$$\sum_{k=x}^{\infty} \binom{n+k-1}{k} p^k q^n \qquad q = \frac{1}{Q} \qquad p = \frac{P}{Q} = 1 - q \qquad 0 < p < 1 \qquad n, x \in \mathbb{N}$$

for x = 0, 1, 2, ... This is also known as the Pascal distribution.

MATHEMATICAL PROBABILITY DISTRIBUTIONS MENU { PROB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
UPPER TAIL POISSON	**POSN**(N,m)	N m	VALUE
$\sum_{n=N}^{\infty} e^{-m} \frac{m^n}{n!}$ $0 < m < \infty$ $N = 0, 1, 2, \ldots$			
UPPER TAIL KOLMOGOROV–SMIRNOV	**UTKS**(λ)	λ	VALUE
$2 \sum_{j=1}^{\infty} (-1)^{j-1} e^{-2j^2\lambda^2}$ $\lambda \geq 0$			
UPPER TAIL RAYLEIGH	**RAYL**(σ,x)	σ x	VALUE
$EXP(-x^2/2\sigma^2)$ $x \geq 0$ $\sigma > 0$			
UPPER TAIL RICIAN	**RICE**(r,β)	r β	VALUE
$\int_{\beta}^{\infty} e^{-(x+r)} I_0(\sqrt{4rx}) dx$ $\beta \geq 0$ $r \geq 0$			

MATHEMATICAL PROBABILITY DISTRIBUTIONS MENU { PROB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
This computation can be very slow. The Marcum Q function is available. See { MISC **MQ** }.			
UPPER TAIL MARCUM Γ_N	**MγN**(N,β)	N β	VALUE
$\int_{\beta}^{\infty} \frac{x^{N-1} e^{-x}}{(N-1)!} dx \qquad \beta \geq 0 \quad N = 1, 2, \ldots$			
UPPER TAIL MARCUM P_N	**MPN**(N,r,β)	N r β	VALUE
$\int_{\beta}^{\infty} \left(\frac{x}{Nr}\right)^{(N-1)/2} e^{-(x+Nr)} I_{N-1}(\sqrt{4N\,r\,x}) dx \qquad \beta \geq 0 \quad r \geq 0 \quad N = 1, 2, \ldots$ This computation is extremely slow. Consider setting the mode to 7 SCI.			
HP COMB	**COMB**(n,m)	n, m ∈ **ℕ**	VALUE
HP PERM	**PERM**(n,m)	n, m ∈ **ℕ**	VALUE
HP FACTORIAL	**!**	REAL NUMBER	VALUE
See page 147 of the HP 48 owner's manual.			
UP DIRECTORY	**UPDIR**	NONE	PARENT MENU

MATHEMATICAL PROBABILITY DISTRIBUTIONS MENU { PROB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

Abramowitz and Stegun, *Handbook of Mathematical Functions*, AMS 55, 1964.

Johnson, N., and Kotz, S., *Distributions in Statistics*, 4 Volumes, New York, Wiley, 1969–72

Marcum, J., "Statistical Theory of Target Detection by Pulsed Radar", *IRE Transactions on Information Theory*, New York, April 1960.

Raiffa, H., and Schlaifer, R., *Applied Statistical Decision Theory*, Cambridge, M.I.T. Press, 1961.

INVERSE PROBABILITY DISTRIBUTIONS MENU { IPROB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
INVERSE **UTPC**	**IUTPC**(v,P)	P = PROBABILITY	χ^2
Inverse Upper Tail Chi–Square: χ^2 such that **UTPC** = P $\in$ [0,1] $v > 0$ IUTPC(4.5 , 0.5) = 3.85375376767			
INVERSE **UTPF**	**IUTPF**(v_N,v_D,P)	P = PROBABILITY	F
Inverse Upper Tail F Distribution: F such that **UTPF** = P $\in$ [0,1] $v_N, v_D > 0$ IUTPF(4.5 , 6.3 , 0.5) = 0.955677662077			
INVERSE **UTPN**	**IUTPN**(μ,σ,P)	P = PROBABILITY	x
Inverse Upper Tail Normal Distribution: x such that **UTPN** = P $\in$ [0,1] $\sigma > 0$ IUTPN(4.3 , 1.7 , 0.5) = 4.3			
INVERSE **UTPT**	**IUTPT**(v,P)	P = PROBABILITY	t
Inverse Upper Tail T Distribution: t such that **UTPT** = P $\in$ [0,1] $v > 0$ IUTPT(4.3 , 0.5) = 0.735766623839			

INVERSE PROBABILITY DISTRIBUTIONS MENU { IPROB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
INVERSE **UTPE**	**IUTPE**(μ,σ,P)	P = PROBABILITY	x
Inverse Upper Tail Exponential: x such that **UTPE** = P ∈ [0,1] σ > 0 IUTPE(4.2 , 1.7 , 0.5) = 3.67835020695			
INVERSE **WEIB**	**IWEIB**(c,P)	P = PROBABILITY	x
Inverse Upper Tail Weibull: x such that **WEIB** = P ∈ [0,1] c > 0 IWEIB(1.7 , .5) = 0.806061017186			
INVERSE **EXTV**	**IEXTV**(μ,σ,P)	P = PROBABILITY	x
Inverse Upper Tail Extreme Value: x such that **EXTV** = P ∈ [0,1] σ > 0 IEXTV(–1.7 , 2.3 , .5) = –2.07785378824			
INVERSE **UNIF**	**IUNIF**(μ,σ,P)	P = PROBABILITY	x
Inverse Upper Tail Uniform: x such that **UNIF** = P ∈ [0,1] σ > 0 IUNIF(–4.5 , 1.7 , .5) = –4.5			

INVERSE PROBABILITY DISTRIBUTIONS MENU { IPROB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
INVERSE **CAUCH**	**ICAUCH**(θ,α,P)	P = PROBABILITY	x
Inverse Upper Tail Cauchy: x such that **CAUCH** = P ∈ [0,1] $\alpha > 0$ ICAUCH(–4.5 , 1.7 , .5) = –4.5			
INVERSE **LAPL**	**ILAPL**(μ,σ,P)	P = PROBABILITY	x
Inverse Upper Tail Laplace: x such that **LAPL** = P ∈ [0,1] $\sigma > 0$ ILAPL(–4.5 , 1.7 , .5) = –4.5			
INVERSE **UTPG**	**IUTPG**(a,P)	P = PROBABILITY	x
Inverse Upper Tail Gamma: x such that **UTPG** = P ∈ [0,1] a > 0 IUTPG(2.4 , .5) = 2.07615703799			
INVERSE **UTPβ**	**IUTPβ**(a,b,P)	P = PROBABILITY	x
Inverse Upper Tail Beta: x such that **UTPβ** = P ∈ [0,1] a, b > 0 IUTPβ(1.7 , 4.3 , 0.5) = 0.25812987721			

INVERSE PROBABILITY DISTRIBUTIONS MENU { IPROB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
INVERSE **LTPβ**	**ILTPβ**(a,b,P)	P = PROBABILITY	x
Inverse Lower Tail Beta: x such that **LTPβ** = P ∈ [0,1] a, b > 0 ILTPβ(1.7 , 4.3 , 0.5) = 0.25812987721			
INVERSE **BINM**	**IBINM**(N,n,P)	P = PROBABILITY	p
Inverse Upper Tail Binomial: p such that **BINM** = P ∈ [0,1] 0 ≤ n ≤ N n, N ∈ ℕ IBINM(7 , 3 , 0.5) = 0.364116086448			
INVERSE **NEGB**	**INEGB**(n,x,P)	P = PROBABILITY	p
Inverse Upper Tail Negative Binomial: p such that **NEGB** = P ∈ [0,1] n, x ∈ ℕ INEGB(3 , 7 , 0.5) = 0.713763331977			
INVERSE **POSN**	**IPOSN**(N,P)	P = PROBABILITY	m
Inverse Upper Tail Poisson: m such that **POSN** = P ∈ [0,1] N ∈ ℕ IPOSN(7 , 0.5) = 6.66963707455			

INVERSE PROBABILITY DISTRIBUTIONS MENU { IPROB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
INVERSE **UTKS**	**IUTKS**(P)	P = PROBABILITY	λ
Inverse Upper Tail Kolmogorov–Smirnov: λ such that **UTKS** = P $\in$ [0,1] IUTKS(0.5) = 0.82757355519			
INVERSE **RAYL**	**IRAYL**(σ,P)	P = PROBABILITY	x
Inverse Upper Tail Rayleigh: x such that **RAYL** = P $\in$ [0,1] $\sigma > 0$ IRAYL(1.7 , 0.5) = 2.00159703828			
INVERSE **MγN**	**IMγN**(N,P)	P = PROBABILITY	β
Inverse Upper Tail Marcum Γ_N: β such that **MγN** = P $\in$ [0,1] N $\in \mathbb{N}$ IMγN(4 , 0.5) = 3.67206074885			
UP DIRECTORY	**UPDIR**	NONE	PARENT MENU

BIVARIATE NORMAL DISTRIBUTION MENU { BIVN }

FUNCTION	COMMAND	INPUTS	OUTPUTS
PROBABILITY $P\{\mathbf{X} \geq x, \mathbf{Y} \geq y\}$	**PUXUY**(x,y,r)	$x, y \in \mathbf{R}$ $r \in [-1,1]$	VALUE
PROBABILITY $P\{\mathbf{X} \geq x, \mathbf{Y} \leq y\}$	**PUXLY**(x,y,r)	$x, y \in \mathbf{R}$ $r \in [-1,1]$	VALUE
PROBABILITY $P\{\mathbf{X} \leq x, \mathbf{Y} \geq y\}$	**PLXUY**(x,y,r)	$x, y \in \mathbf{R}$ $r \in [-1,1]$	VALUE
PROBABILITY $P\{\mathbf{X} \leq x, \mathbf{Y} \leq y\}$	**PLXLY**(x,y,r)	$x, y \in \mathbf{R}$ $r \in [-1,1]$	VALUE

$$L(h, k, r) = L(k, h, r) = \frac{1}{2\pi}\int_{\arccos r}^{\pi} \exp[-\tfrac{1}{2}(h^2+k^2-2hk\cos w)\operatorname{cosec}^2 w]dw$$

for h, k > 0. This calculation can be extended to the entire real number line.

Substitutions like $x = (X - \mu_x)/\sigma_x$ and $y = (Y - \mu_y)/\sigma_y$ are common.

There are much faster numerical approximations than this integral, but they are very unreliable when |r| is not small. This integral always delivers 10 good digits.

PUXLY(1 , 2 , .3) = 0.149967356516

A faster version of **PLXLY** is given on the next page.

UPPER TAIL NORMAL	**UτPN**(μ,σ,x)	see { PROB }	VALUE
LOWER TAIL NORMAL	**LTPN**(μ,σ,x)	see { PROB }	VALUE
UP DIRECTORY	**UPDIR**	NONE	PARENT MENU

BIVARIATE NORMAL DISTRIBUTION MENU { BIVN }

FUNCTION	COMMAND	INPUTS	OUTPUTS

FASTER COMPUTATION OF BIVARIATE NORMAL DISTRIBUTION

The below program **LTBN** provides faster computation of the bivariate probability **PLXLY** from which the other three probabilities can also be computed. However, its accuracy degrades as |r| approaches 1. Use **PLXLY** to determine the accuracy of **LTBN** in the region of interest.

```
≪ → X Y r ≪ "BIVARIATE NORMAL" 3 DISP 1 X 1 Y → X0 X1 Y0 Y1
≪ 'r+SQ(r)×X×Y/2' →NUM 3 50 FOR K X0 NEG X1 DUP 'X0' STO X × +
K 1 - / DUP 'X1' STO Y0 NEG Y1 DUP 'Y0' STO Y × + K 1 - / DUP
'Y1' STO × 'r^K×(K-1)!/K' →NUM × + NEXT 'EXP(-(SQ(X)+SQ(Y))/2)/2/π'
→NUM × 'LTPN(0,1,X)×LTPN(0,1,Y)' →NUM + » » »
```

PLXLY(1, 2, 0.1) = LTBN(1, 2, 0.1) = .823640898881

PLXLY(1, 2, 0.9) = .841096187037

LTBN(1, 2, 0.9) = .84109570565

Thus, we have the relations:

$P(x \le X, y \le Y, r) = LTBN(X, Y, r)$ $P(x \ge X, y \le Y, r) = LTBN(-X, Y, -r)$

$P(x \le X, y \ge Y, r) = LTBN(X, -Y, -r)$ $P(x \ge X, y \ge Y, r) = LTBN(-X, -Y, r)$

For *theory only* see AMS 55.

For theory and **CORRECT NUMBERS** and **CORRECT EXAMPLES**, see *Tables of the Bivariate Normal Distribution Function and Related Functions*, AMS 50, 1959.

24

MULTI-SERVER QUEUEING DISTRIBUTIONS

INTRODUCTION

This chapter presents the 12 queueing theory commands in the QUE menu. QUE gives commands for the probability distributions and moments that are associated with multi-server queues for an infinite population, but a finite waiting space. Since this model is not available in most textbooks, we overview the mathematics for the user.

MODEL FORMULATION

Let N(t) be the number of customers being processed plus those waiting to be processed at time $t > 0$. Thus, N(t) is the number of customers in the system. Define

$P_n(t) = P\{N(t) = n\}$ = the probability that exactly n customers are in the system.
λ_n = the mean arrival rate of customers (expected number of arrivals per unit time) when there are n customers in the system.
μ_n = the mean departure rate for customers (expected number of departures per unit time) when n customers are in the system.
$P_{ij}(\Delta t) = P\{ N(t + \Delta t) = j \mid N(t) = i \}$ = the Markovian transition probability for a change in the number of customers in the system.
M = maximum number of customers in the system.
s = the number of identical servers.

We make the following realistic assumptions relative to the customer transition probability:

BIRTH POSTULATE: $P_{n,n+1}(\Delta t) = \lambda_n \Delta t + o(\Delta t)$

DEATH POSTULATE: $P_{n,n-1}(\Delta t) = \mu_n \Delta t + o(\Delta t) \quad \mu_0 = 0$

MULTIPLE CHANGE: $P_{i,j}(\Delta t) = o(\Delta t) \quad |i - j| \geq 2$

where o(t) is any function such that $\lim_{t \to 0} \frac{o(t)}{t} = 0$. The first two equations characterize single transitions in terms of λ_n and μ_n. The third says transitions occur one at a time.

These equations lead to a differential equation for $P_n(t)$:

$$\lim_{\Delta t \to 0} \frac{P_n(t + \Delta t) - P_n(t)}{\Delta t} = \frac{dP_n(t)}{dt} = \lambda_{n-1} P_{n-1}(t) + \mu_{n+1} P_{n+1}(t) - (\lambda_n + \mu_n) P_n(t).$$

PURE ARRIVAL (BIRTH) PROCESS

Assume that $\lambda_n = \lambda$ and $\mu_n = 0$ for n = 0, 1, 2, ... Then the solution is

$$P_0(t) = e^{-\lambda t} \qquad P_n(t) = \frac{(\lambda t)^n}{n!} e^{-\lambda t} \qquad n \geq 1.$$

Thus, $P_n(t)$ is Poisson-distributed with parameter λt, and the probability of less than K arrivals during the interval (0, t) is

$$\sum_{n=0}^{K-1} \frac{(\lambda t)^n}{n!} e^{-\lambda t} = \frac{\Gamma(K, \lambda t)}{\Gamma(K)} \qquad K = 1, 2, \ldots,$$

which equals UTPG(K, λt) = 1 – POSN(K, λt). If T is the time required to obtain exactly K arrivals, then P{T > t} = UTPG(K, λt). The probability of no arrival in time T is UTPE(λ^{-1}, λ^{-1}, T).

PURE DEPARTURE (DEATH) PROCESS

Now we assume that $\lambda_n = 0$ and $\mu_n = \mu$ for n = 1, 2, ... This case mathematically parallels arrivals. Noting that (M – n) is the number of departures during the interval (0, t), the probability of no departure and the probability of (M – n) < M departures are

$$P_M(t) = e^{-\mu t}, \qquad P_n(t) = \frac{(\mu t)^{M-n}}{(M-n)!} e^{-\mu t}, \qquad P_0(t) = 1 - \sum_{n=1}^{M} P_n(t).$$

MATH LIBRARY MODEL

The differences between queueing theory models are in the definition of λ_n and μ_n. This model assumes an infinite population of customers but limited space. Thus, $\lambda_n = \lambda$ for $0 \le n \le M$ and zero otherwise, where M is the limit on the number of customers which can be either waiting or getting served. Let all the servers be identical with service rate μ. Departure rate is related to the service rate by $\mu_n = n\mu$ provided $0 \le n \le s$. However, when $n \ge s$, then the departure rate remains at $s\mu$ since we assume those waiting to be served either never leave the queue or if they do, then they are instantly replaced. This model has been called the M/M/c/K queueing system, which is different from the finite population multiple repairman M/M/c/K/K queueing system.

The more common M/M/c multi–server queueing model is a special case of the MATHLIB model. Simply set parameter M to a large positive integer in the following commands. See also the examples.

The resulting equations are given in the menu. Other queueing models are easily programmed.

OTHER QUEUEING MODELS

Other models are very easily programmed. For a nice summary of the equations used in the various queueing models, see Allen.

QUEUEING THEORY MENU {QUE}

FUNCTION	COMMAND	INPUTS	OUTPUTS

STEADY STATE MULTI-SERVER QUEUE WITH FINITE WAITING SPACE

Assumptions and definitions:

Arrival rate Poisson-distributed with the mean arrival rate λ.
Departure rate exponentially distributed with mean departure rate μ.
Number of identical servers equals $s \in \mathbb{N}$.
Loading intensity factor equals $\rho = \lambda/(\mu s) < 1$.
M is the maximum processing load modeling finite waiting space $M \geq s$, $M \in \mathbb{N}$.
D_q is the time that an arrival must wait in the queue before being processed.

LM is the expected queueing plus processing load.
LQM is the expected queueing load.
LPM is expected processing load.
WM is expected time an arrival is in the system including processing time.
WQM is expected time an arrival is waiting in the queue for processing.
PQB is the probability that all servers are busy.
PNSM is the steady state probability that n arrivals are in the system.
PDQW is the probability that $D_q > t$.

TOTAL LOAD	**LM**(s,M,λ,μ)	s, M, λ, μ	VALUE
QUE LOAD	**LQM**(s,M,λ,μ)	s, M, λ, μ	VALUE
PROC LOAD	**LPM**(s,M,λ,μ)	s, M, λ, μ	VALUE
TIME IN SYSTEM	**WM**(s,M,λ,μ)	s, M, λ, μ	VALUE
TIME IN QUEUE	**WQM**(s,M,λ,μ)	s, M, λ, μ	VALUE
PROB OF BUSY	**PQB**(s,M,λ,μ)	s, M, λ, μ	VALUE
PROB{ n in system }	**PNSM**(s,M,λ,μ,n)	s, M, λ, μ, n	VALUE
PROB{ $D_q > t$ }	**PDQW**(s,M,λ,μ,t)	s, M, λ, μ, t	VALUE

QUEUEING THEORY MENU {QUE}

FUNCTION	COMMAND	INPUTS	OUTPUTS
EXPONENTIAL	**UTPE**(μ,σ,x)	see { PROB }	VALUE
GAMMA (ERLANG)	**UTPG**(a,x)	see { PROB }	VALUE
POISSON	**POSN**(N,m)	see { PROB }	VALUE
UP DIRECTORY	**UPDIR**	NONE	PARENT MENU

Define the Poisson arrival and exponential departure rates to be

$$\lambda_n = \begin{cases} \lambda & 0 \le n < M \\ 0 & n \ge M \end{cases} \qquad \mu_n = \begin{cases} n\mu & 0 \le n \le s \\ s\mu & n \ge s \end{cases} .$$

This model corresponds to an infinite population of automobiles that need repair, s available mechanics to perform the repairs, and an equivalent marketing staff of M – s individuals to go out and find customers to come stand in the queue to be served. When M = s, marketing only takes place when one or more mechanics is idle, so **LQM** = **WQM** = **PDQW** = 0. The model thus characterizes customer search requirements. The probability equations are

$$\text{PNSM} = P_n^M = \begin{cases} \dfrac{\left(\dfrac{\lambda}{\mu}\right)^n}{n!} P_0^M & 0 \le n \le s \\ \dfrac{\left(\dfrac{\lambda}{\mu}\right)^n}{s!\, s^{n-s}} P_0^M & s \le n \le M \\ 0 & n > M \end{cases}$$

where P_0^M is given by:

QUEUEING THEORY MENU {QUE}

FUNCTION	COMMAND	INPUTS	OUTPUTS

$$P_0^M = \left[\sum_{n=0}^{s} \frac{\left(\frac{\lambda}{\mu}\right)^n}{n!} + \frac{\left(\frac{\lambda}{\mu}\right)^s}{s!} \frac{\rho(1-\rho^{M-s})}{(1-\rho)} \right]^{-1}$$

and is the probability that all servers are idle. The probability that all servers are busy is

$$P_B = 1 - \sum_{n=0}^{s-1} P_n^M.$$

The probability that the waiting time to be served is greater than t is

$$PDQW = P\{D_q > t\} = \frac{\left(\frac{\lambda}{\mu}\right)^s}{s!} P_0^M \sum_{n=0}^{M-s-1} \rho^n \sum_{k=0}^{n} \frac{(\mu s t)^k}{k!} e^{-\mu s t},$$

from which the moments can be evaluated.

$$LM = LPM + LQM \qquad WM = WQM + \frac{1}{\mu} \qquad WQM = \frac{LQM}{\lambda}$$

$$LPM = \frac{\lambda}{\mu}\left[1 - \frac{s^s \rho^M}{s!} P_0^M\right] \qquad LQM = \frac{\left(\frac{\lambda}{\mu}\right)^s}{s!} P_0^M \frac{\rho}{(1-\rho)^2}\left[1 - \rho^{M-s}(1 + (M-s)(1-\rho))\right]$$

QUEUEING THEORY MENU {QUE}

FUNCTION	COMMAND	INPUTS	OUTPUTS

QUEUEING EXAMPLES

Consider a message-switching center which receives an average of 250 messages per minute. The transmission rate is 1,600 characters per second with an average length of 360 characters. Then the average arrival rate λ = 250/60 = 25/6 messages per second, and the average service rate is μ = 1600/360 = 40/9. The loading intensity factor λ/μ = .9375 and s = 1.

First assume an infinite number (K = 200) of buffers are provided. Then we have: average number of messages in the system = LM = 15.00, average number of messages in the queueing buffer = LQM = 14.06, average utilization = LPM = .94, average response time = WM = 3.60 seconds, average time in the queue = WQM = 3.37 seconds, probability of the queue being busy = PQB = .94. The probability that the 199 buffers (200 – 1) are full is PNSM(1,200,λ,μ,199) = 1.65E–7. To keep from having to enter the data more than once, use the LAST STACK command.

Messages are lost if there is no available buffer. Suppose the probability of all the buffers being full is chosen to be 5E–3. How many buffers are required? Since PNSM(1,41,λ,μ,40) = 5.06E–3 and PNSM(1,42,λ,μ,41) = 4.73E–3, 41 buffers are required, and LM = 12.14, LQM = 11.21, WM = 2.91 seconds, and WQM = 2.69 seconds.

Suppose a computer software manufacturer is planning a telephone help facility and the average question takes 5 minutes to answer. An average of 36 calls per hour are expected. If less than 0.5% of all callers must wait more than a minute, how large a staff is required?

The calling rate is λ = 36/60 = 0.6 per minute, and the service rate μ = 1/5 = 0.2 per minute. Since PDQW(8, 200,0.6,0.2,1) = 0.00476, eight people answering eight phones are required. For 8 people, the probability of a caller having to wait is PQB = 0.0129.

QUEUEING THEORY MENU {QUE}

FUNCTION	COMMAND	INPUTS	OUTPUTS

Allen, A., *Probability, Statistics, and Queueing Theory*, Cambridge, Academic Press, 1990.

Kleinrock, L., *Queueing Systems*, Volume I: Theory, New York, Wiley, 1975.

Kleinrock, L., *Queueing Systems*, Volume II: Computer Applications, New York, Wiley, 1976.

Saaty, T., *Elements of Queueing Theory*, New York, McGrapw-Hill, 1961.

25

SYMBOLIC ARRAYS AND ADVANCED CALCULUS

INTRODUCTION

This chapter presents the 57 symbolic array and differential equation commands. The SYMB commands extend the numerous array commands for numbers to functions. These extensions provide the mathematical framework for:

- Vector and matrix algebra
- Vector and matrix calculus
- Gradient, divergence, curl, and Laplacian in eight orthogonal curvilinear coordinates
- Numerical solution of scalar, vector, and matrix differential equations
- Symbolic solution of scalar, vector, and matrix differential equations
- Scalar and matrix partial fraction expansions
- Symbolic calculation of scalar and matrix inverse Laplace and z transforms
- Evaluation of complex contour integrals
- Evaluation of scalar and matrix complex residues
- Solution of state space differential equations
- Evaluation of vector line integrals
- Evaluation of vector volume integrals
- Applications in electronic and control systems

To accomplish this, we exploit the generality of the HP 48 list object to formulate symbolic vector and matrix objects. Some of the material is rather advanced, and we assume the reader is familiar with basic vector, matrix, calculus, and differential equation concepts.

STRUCTURE OF SYMBOLIC VECTORS AND MATRICES

Symbolic vectors and matrices follow the same conventions as HP 48 array objects, and the dimensionality rules are the same. However, the elements of symbolic arrays can include functions as well as real and complex numbers. The array delimiters are braces instead of brackets:

	HP ARRAY	SYMBOLIC ARRAY
VECTOR	[1 2 3 4]	{ A B C D }
COLUMN VECTOR	[[1][2][3]]	{{ A }{ B }{ C }}
ROW VECTOR	[[1 2 3 4]]	{{ A B C D }}
MATRIX	[[1 2][3 4]]	{{ A B }{ C D }}

Thus, a symbolic vector is an HP 48 list and a symbolic matrix is a list of lists.

LINEAR ALGEBRA

Symbolic array algebraic commands **SADD**, **SSUB**, **SMPY**, and **SDOT** provide addition, subtraction, and multiplication. **SCHS** provides negation and **STRN** provides non-Hermitian transposition. **STRAC** and **SDET** provide the trace and determinant operations. **SABS** is the Frobenius (Euclidean) norm. **SMI** provides symbolic matrix inverses. **SRND**, **SRE**, and **SIM** extend the HP 48 commands **RND**, **RE**, and **IM** to symbolic matrices. **TIFRE** is a handy cleanup function which automatically discards the imaginary part of the elements of a symbolic matrix or list when it is insignificant. **SSIZE** and **SCNJ** extend the **SIZE** and **CONJ** commands to symbolic objects. These and only these sixteen commands work for either symbolic or HP 48 arrays, BUT ALL MATRIX (VECTOR) ARGUMENTS MUST BE THE SAME TYPE AND DIMENSION. You cannot add a symbolic array to an HP 48 array object nor can you add a vector to a column vector. The dimensions and the type must be the same. Type conversion commands are given below.

SMNR provides the matrix minor operation, and **SCROS** provides vector cross-product operations. **SADDS** and **SSUBS** provide the matrix add and subtract scalar function operations.

LREV in the VECTR menu is available for reversing symbolic vectors (lists). Symbolic vectors may be rotated with command **VROT**. This will also roll (rotate) the rows of a symbolic matrix. Commands **ZFILN** and **REDN** will zero-fill and reduce the size of symbolic vectors. The last three commands are discussed in Chapter 26.

CONVERSION COMMANDS

M→SO and **M←SO** provide conversion between HP 48 matrices and symbolic matrices. **L→SO** and **L←SO** provide conversion between symbolic column vectors and vectors.

The commands **VREV**, **→ROW**, and **→COL** do accept symbolic vector (list) inputs, but all the elements must be numbers since the output is an HP array object.

SUBSET EXTRACTION AND REPLACEMENT

Vectors may be extracted from matrices or used to replace rows or columns of matrices with the commands **SERW**, **SECOL**, **SRRW**, **SRCOL**. The **VSUBS** and **VREPL** commands in the VECTR menu can also be used for extracting and replacing parts of symbolic vectors (lists).

OBJECT COMMANDS

SOB→ puts a symbolic array on the stack, and **→SOB** converts the elements on the stack into a symbolic array. They are the equivalent of the HP 48 **OBJ→** and **→OBJ** commands and follow the same rules. **SGET** and **SPUT** perform the **GET** and **PUT** operations for symbolic arrays.

SYMBOLIC ALGEBRA OPERATIONS

Symbolic algebra commands include **SCOLC**, **SEXPD**, **SEXCO**, **SEVAL**, **SVBAR** (where), **MAT↑**, **MAT↓**, **SNUM**, **S1F1**, and **L1F1**. Program S2F1 (M2F1 for symbolic matrices) is available. See Appendix A.

Vector commands **VECTX**, **VECTD**, and **V2F1** in the VSAG menu do accept symbolic vectors as inputs and return a symbolic vector if either input is a symbolic vector.

VECTOR AND MATRIX CALCULUS

SDERV and **SINTG** provide vector and matrix differentiation and integration. Gradient, divergence, and curl commands are provided by **SGRD**, **SDIV**, and **SCURL**. The Laplacian is provided as the composition of the gradient and divergence.

RUNGE-KUTTA DIFFERENTIAL EQUATION SOLVER

DESOL is a very flexible and powerful Runge-Kutta differential equation integrator for scalar, vector, and matrix differential equations. Nth order differential equations can be expressed in terms of vector differential equations and solved. Examples for each case are given.

SYMBOLIC DIFFERENTIAL EQUATION SOLUTIONS

HVSDE provides Heaviside partial fraction expansions. **RESDP** and **RESDA** compute complex residues. The **IXFRM** command provides both symbolic inverse Laplace and z transforms. These commands combined with **SMI** provide powerful capability for solving scalar, vector, and matrix linear differential equations by Laplace transform techniques. Examples are given in the menu.

If, for the specified root or roots and the value of $\varepsilon > 0$, these programs conclude that a specified root is in fact not a root, they will abort with a "Bad Guess(es)" error. Either ε is too small or you need to more accurately compute the root(s). Polynomials with repeated roots are very difficult to compute accurately. See the **AROOT** example.

STATE SPACE SOLUTIONS

SMI can also compute the resolvent matrix for state space matrix differential equation applications. Related commands are also given in the FILTR menu discussed in Chapter 27, including zero-pole calculations, controllable, observable, and Jordan canonical forms. Chapter 28 extends these techniques to discrete state space systems.

VECTOR LINE AND VOLUME INTEGRATION

Examples of vector line and volume integration are given.

PLOTTING SYMBOLIC VECTORS

The vector plot programs in the PLOT menu discussed in Chapter 2 do support plotting symbolic vectors (SV), which are lists. However, all the elements must be numbers.

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

SYMBOLIC MATRICES are defined to have the form:

{ LIST1 LIST2 LIST3 ... LISTN }

where N is the number of rows, each of the N lists has the same size, and that size is the number of columns. The elements may be numbers or functions. Dimensionality rules are the same as those for HP 48 arrays. Let SM denote the set of symbolic matrices. Two special cases are the set of symbolic column vectors denoted by SC and the set of symbolic row vectors denoted by SR. Closely related to the set of symbolic matrices is the list object itself, which also can be used with many of the below operations. Let SV denote the set of symbolic vectors which are the same as the HP 48 list object. Let 𝔸 demote a standard HP 48 array. Below are simple examples:

```
{ { A  B  C }                { { A }          { { A  B  C  D } } ∈ SR
  { D  E  F }                  { B }
  { G  H  I }                  { C }           SIZE = { 1 , 4 }
  { J  K  L }                  { D }
  { M  N  O } } ∈ SM           { E } } ∈ SC    { A  B  C  D  E } ∈ SV

  SIZE = { 5 , 3 }           SIZE = { 5 , 1 }      SIZE = { 5 }
```

WARNING: Lower case "e" and "i" are specially defined symbols on the HP 48. The SADD example below will halt with a + error if the "E" is changed to "e."

A + B	SADD(A,B)	A, B ∈ SM OR SV OR 𝔸	A + B

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
A – B	**SSUB**(A,B)	A, B ∈ SM OR SV OR 𝔸	A – B

A and B may be SM, SC, SR, or SV, *but A and B must have the same dimensions.*

SADD({ { A B } { C D } }) , { { E F } { G H } })
= { { 'E+A' 'F+B' } { 'G+C' 'H+D' } }

SSUB({ A B C } , { D E F }) = { 'A–D' 'B–E' 'C–F' }

A × B	**SMPY**(A,B)	A ∈ SM OR 𝔸	A × B

A is an SM, and B is a properly dimensioned SM, SV or scalar. For example:

SMPLY({ { A B } { C D } } , { { E } { F } }) = { { 'A×E + B×F' } { 'C×E + D×F' } }

SMPLY({ { A B } { C D } } , { E F }) = { { 'A×E + B×F' } { 'C×E + D×F' } }

SMPLY({ { A B } { C D } } , E) = { { 'A×E' 'B×E' } { 'C×E' 'D×E' } }

DOT PRODUCT	**SDOT**(A,B)	A, B ∈ SM OR SV OR 𝔸	VALUE

Vectors A and B may be SC or SR, *but A and B must have the same dimensions.*

SDOT({ A B C D } , { E F G H }) = 'A×E + B×F + C×G + D×H'

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
CHANGE SIGN	**SCHS**(M)	M ∈ SM OR SV OR 𝔸	– M
SCHS({ { A B } { C D } }) = { { '–A' '–B' } { '–C' '–D' } }			
TRANSPOSE	**STRN**(M)	M ∈ SM OR 𝔸	TRANSPOSE M
STRN({ { A B C } { E F G } }) = { { A E } { B F } { C G } }			
SYMBOLIC OBJECT TO STACK	**SOB→**(M)	M ∈ SM OR SV SOB→({ {A B} {C D} {E F} }) SOB→({A B C })	STACK 7: 'A' 6: 'B' 5: 'C' 4: 'D' 3: 'E' 2: 'F' 1: { 3 2 } 4: 'A' 3: 'B' 2: 'C' 1: { 3 }
This command operates like the **OBJ→** command for HP vectors and arrays.			

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
STACK TO SM	→**SOB**(L)	L = {#R #C}, {#}, #	SM

→**SOB** operates like the →**ARRY** command for vectors and arrays and provides the functional inverse operation for the command **SOB**→. Like the →**ARRY** command, the { } may be omitted when specifying the SIZE of a symbolic vector SV.

EXTRACT ROW	**SERW**(M,ROW)	M ∈ SM ROW #	SV
EXTRACT COLUMN	**SECOL**(M,COL)	M ∈ SM COL #	SV

SERW and **SECOL** extract the specified row (column) of M as a symbolic vector.

SECOL({ { A B } { C D } } , 2) = { B D }

REPLACE ROW	**SRRW**(M,V,ROW)	M ∈ SM V ∈ SV ROW #	M WITH V INSERTED
REPLACE COLUMN	**SRCOL**(M,V,COL)	M ∈ SM V ∈ SV COLUMN #	M WITH V INSERTED

SRRW and **SRCOL** replace the specified row (column) of M with SV V.

SRCOL({ { A B } { C D } } , { E F } , 2) = { { A E } { C F } }

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
SYMBOLIC MATRIX DERIVATIVE	**SDERV**(M,v)	M ∈ SM OR SV v = VARIABLE	M_{jk} V ∂

The derivative of a matrix or vector is the derivative of each element. **SDERV** applies differentiation for the specified variable v to each element of M. See Chapter 23 of the HP 48 owner's manual.

SDERV({ 'SIN(z)' 'COS(z)' } , z) = { 'COS(z)' '–SIN(z)' }

SYMBOLIC MATRIX INTEGRAL	**SINTG**(M,L,U,v)	M ∈ SM OR SV L = LOWER LIMIT U = UPPER LIMIT v = VARIABLE	INTEGRAL L U M_{jk} V ∫

The integral of a matrix or vector is the integral of each element. **SDERV** applies integration for the specified variable v and limits to each element of M. See Chapter 23 of the HP 48 owner's manual.

SINTG({ 'SIN(z)' 'COS(z)' } , 0 , z , z) = { '1–COS(z)' 'SIN(z)' }

SYMBOLIC MATRIX SCALAR ADD	**SADDS**(M,s)	M ∈ SM OR SV s = SCALAR	$M_{jk} = M_{jk} + s$

SADDS({ 'COS(z)' 'SIN(z)' } , 'z^3') = { 'z^3+COS(z)' 'z^3+SIN(z)' }

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
SYMBOLIC MATRIX SCALAR SUBTRACT	**SSUBS**(M,s)	M ∈ SM OR SV s = SCALAR	$M_{jk} = M_{jk} - s$
SSUBS({ 'COS(z)' 'SIN(z)' } , 'z^3') = { '–z^3+COS(z)' '–z^3+SIN(z)' }			
SM COLCT	**SCOLC**(M)	M ∈ SM	M
SM EXPAND	**SEXPD**(M)	M ∈ SM	M
SCOLC and **SEXPD** apply the algebraic commands **COLCT** and **EXPAN** to each element of M. See Chapter 22 of the HP 48 owner's manual.			
MATRIX TO SYMBOLIC MATRIX	**M→SO**(M)	MATRIX M VECTOR M	SM M SV M
SYMBOLIC MATRIX TO MATRIX	**M←SO**(M)	SM M SV M	MATRIX M VECTOR M
M→SO and **M←SO** convert between ordinary HP 48 array objects and symbolic matrices and vectors. The elements must be numbers, not functions. M→SO([[1 2] [3 4]]) = { { 1 2 } { 3 4 } } M←SO({ 1 2 3 4 }) = [1 2 3 4]			
SV (LIST) TO SM	**L→SO**(L)	L ∈ SV	SC

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
SM TO SV (LIST)	**L←SO**(V)	V ∈ SC OR SR OR SV	SV
L→SO and **L←SO** are for conversion of vectors only. L→SO({ A B C }) = { { A } { B } { C } } L←SO({ { A B C } }) = { A B C }			
GET FOR SM	**SGET**(M,L)	M ∈ SM L = {R C}	{R C} ELEMENT
SGET is the symbolic matrix equivalent of the HP 48 **GET** command. For symbolic vectors, use **GET**. Both row and column must be specified. SGET({ { A B } { C D } } , { 1 2 }) = B			
PUT FOR SM	**SPUT**(M,L,V)	M ∈ SM L = {R C} V = NEW ELEMENT	M WITH V AT {R C}
SPUT is the symbolic matrix equivalent of the HP 48 **PUT** command. For symbolic vectors, use **PUT**. Both row and column must be specified. SPUT({ { A B } { C D } } , { 1 2 } , E) = { { A E } { C D } }			
SM EVAL	**SEVAL**(M)	M ∈ SM OR SV	M_{jk} EVAL
SM EXCO	**SEXCO**(M)	M ∈ SM OR SV	M_{jk} EXCO
SM | (where)	**SVBAR**(M,L)	M ∈ SM OR SV L = LIST	M_{jk} L |

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
SM ↑MATCH	**MAT↑**(M,L)	M ∈ SM OR SV L = LIST	M_{jk} L ↑MATCH DROP
SM ↓MATCH	**MAT↓**(M,L)	M ∈ SM OR SV L = LIST	M_{jk} L ↓MATCH DROP
SM →NUM	**SNUM**(M)	M ∈ SM OR SV	M_{jk} →NUM

The above six commands simply apply the stated algebraic operation to each of the elements of the symbolic matrix or vector. See Chapter 22 of the HP 48 owner's manual.

SV CROSS-PRODUCT	**SCROS**(A,B) DIM 3 ONLY	A, B ∈ SC OR SV	3 DIM SV

SCROS computes the cross-product for two three–dimensional symbolic vectors.

SCROS({ A B C } , { D E F }) = { 'B×F–C×E' '–A×F+C×D' 'A×E–B×D' }

DETERMINANT	**SDET**(M)	M ∈ SM OR 𝔸	VALUE

SDET computes the determinant of square m × m symbolic matrices.

SDET({ { A B } { C D } }) = 'D×A – B×C'

For m > 3 and M ∈ SM, **SDET** is quite slow due to the numerous algebraic collections.

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

SYMBOLIC LINEAR SOLUTIONS – CRAMER'S RULE

Given the fully determined linear system Ax = b where A is n × n and nonsingular, the ith value of the solution vector x, say x_i, is given by x_i = SDET(A_i) / SDET(A), where A_i is the A matrix with its ith column replaced with vector b. An example of this technique is given below under simultaneous differential equations.

MINOR	**SMNR**(M,R,C)	M ∈ SM ROW COLUMN	M MINOR

Performs first step in computing a minor or cofactor, that of extracting the submatrix where row R and column C have been deleted. To complete, call **SDET** and adjust the sign. An example application of **SDET** and **SMNR** in solving state space differential equations using Laplace transform techniques is given near the end of this table.

SMNR({ { A B C } { D E F } { G H I } } , 2 , 3) = { { A B } { G H } }

GRADIENT ∇ ϕ	**SGRD**(ϕ,L)	ϕ L = {z_1 . . . z_N}	VECTOR

SGRD computes the N–dimensional gradient of a scalar function $\phi(z_1, z_2, \ldots, z_N)$.

Given $\phi = 2x^3y^2z^4$, then $\nabla\phi = 6x^2y^2z^4\,\mathbf{i} + 4x^3yz^4\,\mathbf{j} + 8x^3y^2z^3\,\mathbf{k}$

SGRD('2x^3y^2z^4' , { x y z }) = { '6x^2y^2z^4' '4x^3yz^4' '8x^3y^2z^3' }

where the multiplication operators × are not shown but obviously required.

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
DIVERGENCE $\nabla \bullet V$	**SDIV**(V,x,y,z)	DIM 3 VECTOR x y z	SCALAR

SDIV computes the divergence of a three–dimensional vector function V $\in$ SC or SV.

Given **A** = x^2z **i** – $2y^3z^2$ **j** + xy^2z **k**, then $\nabla \bullet$**A** = $2xz - 6y^2z^2 + xy^2$

SDIV({ 'x^2z' '–2y^3z^2' 'xy^2z' } , x , y , z) = ' – 6y^2z^2 + xy^2 + 2xz'

where the multiplication operators $\times$ are not shown but obviously required.

CURL $\nabla \times V$	**SCURL**(V,x,y,z)	DIM 3 VECTOR x y z	VECTOR

SCURL computes the curl of a three–dimensional vector function V $\in$ SC or SV.

Given **A** = xz^3 **i** – $2x^2yz$ **j** + $2yz^4$ **k**, then $\nabla \times$ **A** = $(2z^4 + 2x^2y)$ **i** + $3xz^2$ **j** – 4xyz **k**

SCURL({ 'xz^3' '–2x^2yz' '2yz^4' } , x , y , z) = { '2x^2y+2z^4' '3xz^2' '–4xyz' }

where the multiplication operators $\times$ are not shown but obviously required.

LAPLACIAN $\nabla^2 \phi$	$\nabla^2 \phi(x, y, z) = \nabla \bullet [\ \nabla \phi(x, y, z)\ ]$

The Laplacian operation is implemented as the composition of the gradient and the divergence commands. It is not provided as a command since that would limit its generality for the orthogonal curvilinear coordinate systems discussed below.

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

GENERAL ORTHOGONAL CURVILINEAR COORDINATES

The above three commands and the Laplacian can be used with other orthogonal coordinate systems. For scalar ϕ and vector **A**, which are functions of the orthogonal curvilinear coordinates u_1, u_2, and u_3, the general expressions are

$$\nabla \phi = \begin{bmatrix} \frac{1}{h_1} & 0 & 0 \\ 0 & \frac{1}{h_2} & 0 \\ 0 & 0 & \frac{1}{h_3} \end{bmatrix} [\, \nabla \phi(u_1,u_2,u_3) \,]$$

$$\nabla \cdot \mathbf{A} = \frac{1}{h_1 h_2 h_3} \nabla \cdot \left(\begin{bmatrix} h_2h_3 & 0 & 0 \\ 0 & h_3h_1 & 0 \\ 0 & 0 & h_1h_2 \end{bmatrix} \mathbf{A}(u_1,u_2,u_3) \right)$$

$$\nabla \times \mathbf{A} = \frac{1}{h_1 h_2 h_3} \begin{bmatrix} h_1 & 0 & 0 \\ 0 & h_2 & 0 \\ 0 & 0 & h_3 \end{bmatrix} \nabla \times \left(\begin{bmatrix} h_1 & 0 & 0 \\ 0 & h_2 & 0 \\ 0 & 0 & h_3 \end{bmatrix} \mathbf{A}(u_1,u_2,u_3) \right).$$

Examples are given on the next page.

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

Rectangular coordinates O{x, y, z} $u_1 = x$, $u_2 = y$, $u_3 = z$ $x = x$, $y = y$, $z = z$
$h_1 = h_2 = h_3 = 1$

Cylindrical polar coordinates O{r ,ϕ ,z} $u_1 = r$, $u_2 = \phi$, $u_3 = z$
$x = r \cos \phi$, $y = r \sin \phi$, $z = z$
$h_1 = 1$, $h_2 = r$, $h_3 = 1$ $r \geq 0$ $\phi \in [-\pi, \pi]$

Spherical polar coordinates O{r, θ, ϕ} $u_1 = r$, $u_2 = \theta$, $u_3 = \phi$
$x = r \sin \theta \cos \phi$, $y = r \sin \theta \sin \phi$, $z = r \cos \theta$
$h_1 = 1$, $h_2 = r$, $h_3 = r \sin \phi$ $r \geq 0$ $\theta \in [0, \pi]$ $\phi \in [-\pi, \pi]$

SPECIAL NOTE: the HP 48 uses the nonstandard convention of making the angle off the z axis the third value of the triplet {r,θ,ϕ} instead of the second. Thus, the θ and ϕ on page 171 of the HP 48 owner's manual correspond to the ϕ and θ here, respectively. Also, the calculator allows the angle off the z axis to be outside the range [0, π], which is mathematically wrong since θ and ϕ become inconsistent.

Parabolic cylindrical coordinates O{u, v, z} $h_1 = h_u$ $h_2 = h_v$ $h_3 = h_z$

$$x = \tfrac{1}{2}(u^2 - v^2) \quad y = uv \quad z = z \quad v \geq 0$$

$$h_u = h_v = \sqrt{u^2 + v^2} \quad h_z = 1 \quad u, z \in \mathbb{R}$$

Paraboloidal coordinates O{u, v, ϕ} $h_1 = h_u$ $h_2 = h_v$ $h_3 = h_\phi$

$$x = uv \cos \phi \quad y = uv \sin \phi \quad z = \tfrac{1}{2}(u^2 - v^2) \quad u, v \geq 0 \quad \phi \in [0, 2\pi]$$

$$h_v = h_u = \sqrt{u^2 + v^2} \quad h_\phi = uv$$

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

Elliptic cylindrical coordinates O{u, v, z) $h_1 = h_u$ $h_2 = h_v$ $h_3 = h_z$ $a > 0$

$x = a \cosh u \cos v$, $y = a \sinh u \sin v$ $z = z$ $u \geq 0$ $v \in [0, 2\pi]$

$h_u = h_v = a\sqrt{\sinh^2 u + \sin^2 v}$ $h_z = 1$

Prolate spheroidal coordinates O{ξ, η, ϕ} $h_1 = h_\xi$ $h_2 = h_\eta$ $h_3 = h_\phi$ $a > 0$

$x = a \sinh\xi \sin\eta \cos\phi$ $y = a \sinh\xi \sin\eta \sin\phi$ $z = a \cosh\xi \cos\eta$ $\xi \geq 0$

$h_\xi = h_\eta = a\sqrt{\sinh^2\xi + \sin^2\eta}$ $h_\phi = a \sinh\xi \sin\eta$ $\eta \in [0, \pi]$ $\phi \in [0, 2\pi]$

Oblate spheroidal coordinates O{ξ, η, ϕ} $h_1 = h_\xi$ $h_2 = h_\eta$ $h_3 = h_\phi$ $a > 0$

$x = a \cosh\xi \cos\eta \cos\phi$ $y = a \cosh\xi \cos\eta \sin\phi$ $z = a \sinh\xi \sin\eta$ $\xi \geq 0$

$h_\xi = h_\eta = a\sqrt{\sinh^2\xi + \sin^2\eta}$ $h_\phi = a \cosh\xi \cos\eta$ $\eta \in [-\pi/2, \pi/2]$ $\phi \in [0, 2\pi]$

EXAMPLE – DIRECTIONAL DERIVATIVES

In what direction from the point (3, 1, –1) is the directional derivative of $\phi = x^2y^4z^3$ a maximum? Evaluate 'SGRD(x^2y^4z^3, { x y z })' = { '2xy^4z^3' '4x^2y^3z^3' '3x^2y^4z^2' }. Then put { x 3 y 1 z –1 } on the stack and push **SVBAR** to obtain the symbolic vector { –6 –36 27 }, which corresponds to –6**i** – 36**k** + 27**k**. The directional derivative is a maximum, in this direction. Push **SABS** to compute the magnitude of this maximum, which equals 45.39824.

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
ABSOLUTE VALUE	**SABS**(M)	M ∈ SM OR SV OR 𝔸	VALUE

Square root of the sum of the squares of the absolute values of the elements.

SABS({ { 2 5 } { A 3 } }) = √(38 + ABS(A)^2)

TRACE	**STRAC**(M)	M ∈ SM OR 𝔸	VALUE

The trace of a square matrix is the sum of the diagonal elements. M must be square.

STRAC({ { A B C } { D E F } { G H I } }) = 'I + E + A'

MATRIX ELEMENT OPERATIONS	**S1F1**(M,P)	M ∈ SM OR SV P = PROGRAM	M_{jk} < P > EVAL

S1F1 executes the program P on each element of M. It is similar to the **V1F1** and **M1F1** commands. See also the **L1F1** command at the end of this menu.

S1F1({ 1.25 3.46 –8.16 } , < →Q >) = { '5/4' '173/50' '–(204/25)' }

S1F1({ A B C D } , < EXP >) = { 'EXP(A)' 'EXP(B)' 'EXP(C)' 'EXP(D)' }

S1F1({ { A B } { C D } } , < ATAN >) =

{ { 'ATAN(A)' 'ATAN(B)' } { 'ATAN(C)' 'ATAN(D)' } }

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
ORDER FOUR RUNGE–KUTTA DIFFERENTIAL EQUATION SOLUTIONS	**DESOL** (P,α,a,b,N)	5: P = PROGRAM 4: α =INITIALCOND 3: a = START VAL 2: b = END VALUE 1: N = # POINTS	SOLUTION LIST

DESOL solves scalar, vector, and matrix nth order nonlinear differential equations by the Runge–Kutta method. P is a program, and the differential equation must be of the form y' = P(t, y) where y is a scalar, vector, or matrix, and t is the integration variable. The initial condition α = y(a) must have the same dimension as y(t) and y'(t, y). The integration is performed N times using (b – a)/N for the step size, where a and b are the start and end values. Build adaptive step size logic around **DESOL**.

SCALAR DIFFERENTIAL EQUATION EXAMPLE

As a simple scalar example consider y' = t – y + 1 with y(0) = 1.
P = « → t y ' t – y + 1 ' » and α = 1.
With a = 0, b = 1, and N = 10 we obtain the result { 1 1.0048375 1.01873090141 1.040818422 1.07032028892 1.10653093443 1.14881193438 1.19658561867 1.24932928973 1.3065699912 1.36787977441 } with a maximum error 3.332E–7.

VECTOR DIFFERENTIAL EQUATION EXAMPLE

Next consider the response of a simple two–loop RLC electrical circuit whose currents are described by the differential equations:
$I_1' = -4I_1 + 3I_2 + 6$, $I_1(0) = 0$, and $I_2' = -2.4I_1 + 1.6I_2 + 3.6$, $I_2(0) = 0$. Then let P = « → t v « [[–4 3] [–2.4 1.6]] [[6] [3.6]] → A C « ' A × v + C ' →NUM » » »
and α = [[0] [0]]. With a = 0, b = 0.5, and N = 5 we obtain the result:

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

{ [[0] [0]] [[.5382552] [.31962624]] [[.968498737528] [.568782173035]] [[1.31071903921] [.760733131868]] [[1.58126523897] [.906320617947]] [[1.79350749012] [1.01440241677]] }. The maximum integration error for this example is 2.193E–5.

SECOND ORDER DIFFERENTIAL EQUATION EXAMPLE

Next consider the second order initial–value problem $y'' - 2y' + 2y = e^{2t} \sin t$, $0 \le t \le 1$, $y(0) = -0.4$, $y'(0) = -0.6$. Define $u_1'(t) = u_2(t)$ and $u_2'(t) = e^{2t} \sin t - 2u_1(t) + 2u_2(t)$ with initial conditions $u_1(0) = -0.4$ and $u_2(0) = -0.6$. Let P = ≪ → t v ≪ [[0 1] [–2 2]] {{ 0 } { 'EXP(2×t)SIN(t)' }} → A C ≪ A v × C SEVAL M←SO + ≫≫≫ and α = [[–0.4] [–0.6]]. With a = 0, b = .5, and N = 5, we obtain:
{ [[–0.4] [–0.6]] [[–.461733342331] [–.631631242117]] [[–.525559883217] [–.640148947771]] [–.588601435615] [–.613663805926]] [[–.646612306037] [–.536582028658]] [[–.693566655301] [–.388738097323]] }
with a maximum error of 5.96E–7.

POLYNOMIAL ROOT	**PROOT**(L, ε, MAX)	3: COEF LIST 2: ε > 0 1: MAX STEPS	[VECTOR]

PROOT takes the complex coefficient list L (see **FEVAL** for definition and order) and creates a Frobenius matrix M. **EIGEN** is used to compute the eigenvalues of M, which are the complex roots of L. For example let $f(x) = x^5 - 0.2x^4 + 7x^3 + x^2 - 3.5x + 2 = 0$. Then list L = { 2 –3.5 1 7 –0.2 1 }. With ε = 1E–10, the roots are (.140377913, ±2.7314490896), (–.908563257, 0), (.413903715, ±.3506447075). MAX is the maximum number of iterations (say 20) **EIGEN** is allowed to converge each eigenvalue. This description is continued on the next page.

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
If the solution fails at some eigenvalue, CONVERGENCE FAILED will be displayed. Push ATTN, and the diagonal matrix will be returned, but only the last values of the vector (the ones which converged) are actual roots.			
POLYNOMIAL ROOT	**AROOT** (L, v, ε, MAX)	L v ε MAX	[VECTOR]

AROOT uses Laguerre's method for solving for all the roots of the polynomial defined by the list L. Value v is the initial guess for all of the roots, and ε sets the convergence criterion. MAX (say 100) is the maximum iterations allowed to converge each root. After converging each root, **DEFLT** is used to remove that root from polynomial L. The most difficult polynomial root solutions occur when there are repeated roots. Consider the example: '$x^{12} + 99x^{11} - 377x^{10} - 26395x^9 + 149080x^8 + 1703048x^7 - 15440048x^6 + 8684864x^5 + 302914240x^4 - 1377763200x^3 + 2718976000x^2 - 2620320000x + 1008000000$' = $(x-2)^5$ (x–5) (x+6) (x–7) (x+10) (x–10) (x+15) (x+100). With v = 0, ε = 1E–8, and MAX = 100, we only get two roots: (1.98607080, 0) and (1.99562237, –1.32705219E–2). Dropping ε = 1E–4 yields the roots (1.98607080, 0) (1.99563474, –1.32669871E–2) (1.99563459, .0132665015) (2.01132960, 8.27241365E–3) (2.01133027, –8.27192808E–3) (5, 0) (–6, 0) (7, 0) (–10, 0) (10, 0) (–15, 0) (–100, 0). The average of the first five roots is (2, 0) to 11 digits. **LROOT** can be used to verify that (2, 0) is the true root, and **DEFLT** can be used to remove it from P, giving the reduced polynomial { (–31500000, 0) (3135000, 0) (1619500, 0) (–108650, 0) (–23945, 0) (673, 0) (109, 0) 1 }, which is easily solved since all the roots are unique.

If the solution fails at some root, CONVERGENCE FAILED will be displayed. Push ATTN, and the computed roots will be returned.

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
HEAVISIDE EXPANSION FORMULA	**HVSDE** (P, Q, r, ε)	4: NUMERATOR 3: DENOMINATOR 2: ROOT r 1: ε > 0	LIST

HVSDE computes the partial fraction expansion coefficients associated with the ratio of polynomials P and Q using the Heaviside expansion rule. r is a real or complex root of Q, and ε is used by **DEFLT** internally to determine if the root is repeated. If r is not a root of Q within ε, then a "Bad Guess(es)" error message is given. An example is given below.

Consider the computation of the inverse Laplace transform of the expression:

$$\mathcal{L}^{-1}\left\{\frac{5s^2 - 15s - 11}{s^4 - 5s^3 + 6s^2 + 4s - 8}\right\} = \mathcal{L}^{-1}\left\{\frac{A}{s+1} + \frac{B}{(s-2)^3} + \frac{C}{(s-2)^2} + \frac{D}{s-2}\right\}$$

Define the coefficient lists P = { −11 −15 5 } and Q = { −8 4 6 −5 1 } and set ε = 1E−10. Then **HVSDE**(P, Q, −1, ε) = { −1/3 } and **HVSDE**(P, Q, 2, ε) = { −7 4 1/3}, so A = −1/3, B = −7, C = 4, and D = 1/3. The inverse transform is thus given by:

$$-\frac{1}{3}e^{-t} + e^{2t}\left[-7\frac{t^{3-1}}{(3-1)!} + 4\frac{t^{(3-2)}}{(3-2)!} + \frac{1}{3}\right] = -\frac{1}{3}e^{-t} + e^{2t}\left[-3.5t^2 + 4t + \frac{1}{3}\right]$$

Programs can be built around **HVSDE** to automate inverse Laplace transforms in order to solve linear constant coefficient differential equations when the numerator is a polynomial.

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
POLYNOMIAL RESIDUE EVALUATION	**RESDP** (P, Q, r, ε)	4: NUMERATOR 3: DENOMINATOR 2: ROOT r 1: ε > 0	VALUE

RESDP evaluates the residues of the poles of Q in the case where both P and Q are polynomial lists and can be used to evaluate contour integrals by the Residue Theorem. If r is not a root of Q within ε, then a "Bad Guess(es)" error is given.

For example, determine the residue at z = i of the equation $z^2/(z^3 - 2z^2 + z - 2)$. Converting the equation to polynomial coefficients gives P = { 0 0 1 } and Q = { −2 1 −2 1 }. With r = (0, 1) and ε = 1E−10, the residue is equal to (.1, −.2).

Next consider the residue of $z^{-1}(z + 2)^{-3}$ at z = −2. Let P = { 1 } and Q = { 0 8 12 6 1 }, r = −2, and ε = 1E−10. Then the residue at −2 equals −1/8.

RESIDUE INTEGRATION EXAMPLE

$$\int_{-\infty}^{\infty} \frac{x^2\,dx}{x^6+2x^5+4x^4+4x^3+5x^2+2x+2} = \oint_C \frac{z^2\,dz}{z^6+2z^5+4z^4+4z^3+5z^2+2z+2}$$

$$= 2\pi i\left\{\frac{9i-12}{100} + \frac{3-4i}{25}\right\} = \frac{7\pi}{50},$$

where contour C includes the real axis and encloses the upper half complex plane. Use **AROOT** to determine that the enclosed poles are at z = i of order 2 and z = −1 + i of order 1. Use **COEFL** to compute P = { 0 0 1 } and Q = { 2 2 5 4 4 2 1 }. With ε = 1E−10, use **RESDP** to evaluate the residue of (0, 1), which equals (−.12, .09) and the residue of (−1, 1), which equals (.12, −.16). Now add the residues and multiply by 2π(0, 1) to obtain .14π.

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

RESIDUE INTEGRATION EXAMPLE

$$\int_0^{2\pi} \frac{d\theta}{5 + 3\sin\theta} = \oint_C \frac{\frac{dz}{iz}}{5 + 3\left(\frac{z - z^{-1}}{2i}\right)} = \oint_C \frac{2\,dz}{3z^2 + 10iz - 3} = 2\pi i \left(\frac{1}{4i}\right) = \frac{\pi}{2},$$

where contour C is the circle of unit radius with center at the origin. Use **AROOT** to find the poles, which are at –3i and –i/3. Only –i/3 lies inside C. Use **COEFL** to compute P = { 2 } and Q = { (–3,0) (0,10) 3 }. With ε = 1E–10 and r = (0, –1/3), the residue equals (0, –.25).

ARBITRARY RESIDUE EVALUATION	**RESDA** (F, α, Q, r, ε)	5: NUMERATOR 4: VARIABLE α 3: DENOMINATOR 2: ROOT r 1: ε > 0	VALUE OR EQUATION

RESDA evaluates the residues of the poles of Q, where Q is a polynomial and F is a function of α. **RESDA** provides the capability to directly integrate inverse Laplace transforms and solve differential equations. Transform the integral into the form of a complex contour integral and apply the Residue Theorem to obtain the result.

If r is not a root of Q within ε, then a "Bad Guess(es)" error message is given.

Examples are given on the next page.

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

SYMBOLIC INVERSE LAPLACE TRANSFORM EXAMPLES

Compute the inverse Laplace transform of $s\,(s+1)^{-3}\,(s-1)^{-2}$. Let $\alpha = s$ and define F = 's×EXP(s×t)'. $(s+1)^3\,(s-1)^2$ implies Q = { 1 1 –2 –2 1 1 }, and with ε = 1E–10 the residues are

$$\frac{1}{16}e^{-t} - \frac{1}{8}t^2e^{-t} \text{ for } r = -1 \quad \text{and} \quad \frac{1}{8}te^{t} - \frac{1}{16}e^{t} \text{ for } r = 1.$$

The sum of these residues is the desired inverse transform.

Similarly, evaluate the inverse Laplace transform of $1/(s^4 + 2s^2 + 1)$. From **AROOT** the roots are $(0, \pm 1) = \pm i$, so with F = 'EXP(st)', α = s, and Q = { 1 0 2 0 1 }, we find that the residues equal $-.25\,t\,e^{\pm it} \mp .25\,i\,e^{\pm it}$, so the sum of the residues equals $(\sin t - t\cos t)/2$, which is the desired inverse Laplace transform.

If e = 2.718... evaluates, then what is returned is correct, but is complex numbers raised to the t power. See the next few pages for techniques to keep e^{at} symbolic.

SYMBOLIC INVERSE TRANSFORM	**IXFRM** (F, α, D, ε)	F α D $\varepsilon > 0$	SYMBOLIC EQUATION

IXFRM computes symbolic inverse Laplace and z transforms using residue integration. F is the numerator function, α is the variable of integration, D is the denominator root vector, and $\varepsilon > 0$ is used internally by **RESDA** to compute the residues. **UNIQE** internally determines if there are repeated roots in D. The degree of the numerator must be less than that of denominator D so that the inverse transform does not contain Dirac delta functions; see the second example.

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

If ε is too small, then a "Bad Guess(es)" error message may be given by **RESDA**.

INVERSE LAPLACE TRANSFORM EXAMPLE

Consider first the example given above with the **HVSDE** command. We will compute the symbolic equation for the response.

$$\mathcal{L}^{-1}\left\{\frac{5s^2 - 15s - 11}{s^4 - 5s^3 + 6s^2 + 4s - 8}\right\} = \mathcal{L}^{-1}\left\{\frac{A}{s+1} + \frac{B}{(s-2)^3} + \frac{C}{(s-2)^2} + \frac{D}{s-2}\right\}.$$

Clear Flag –2 so that e will not evaluate to a number. Define F = $(5s^2 - 15s - 11)e^{st}$. The integration variable is s. The denominator root vector is [–1 2 2 2] and ε = 1E–10. Using these inputs to **IXFRM**, the following program computes the symbolic equation for the inverse Laplace transform.

IXFRM1: « { –11 –15 5 } s XEQN 'e^(s×t)' × s 1E–10 IXFRM »

The result is '–(.333333333333×e^–t)
+((3.49999999999×LN(e)^2×t^2×e^(2×t))+3.99999999999×LN(e)×t×e^(2×t)
+.333333333323×e^(2×t))'.

Observing that LN(e) = 1, this expression reduces to:

$$-\frac{1}{3}e^{-t} + e^{2t}\left[-3.5t^2 + 4t + \frac{1}{3}\right].$$

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

INVERSE LAPLACE TRANSFORM EXAMPLE

Consider next an example where the degree of the numerator and denominator are equal. Then the result will contain a Dirac delta function since the inverse Laplace transform of 1 is the Dirac delta function $\delta(t)$.

$$\mathcal{L}^{-1}\left\{\frac{2s^3 - 4s + 6}{s^3 + 7s^2 + 17s + 15}\right\} = \mathcal{L}^{-1}\left\{A + \frac{B}{s + (2, -1)} + \frac{C}{s + (2, 1)} + \frac{D}{s + 3}\right\}.$$

Clear Flag –2 so that e will not evaluate to a number. Then the following program will compute the inverse Laplace transform for you.

```
IXFRM2: « { 6 -4 0 2 } [ (-2,1) (-2,-1) (-3,0) ] → N D « N D CLIST
PDVD DROP s XEQN 'e^(s×t)' × s D 1E-10 IXFRM 't>0' SWAP ROT
EVAL IFTE » »
```

The result is: 'IFTE(t>0 , (2,–7)×e^((–2,+1)×t) + (2,+7)×e^((–2,–1)×t)
+ (–18,0)×e^((–3,0)×t)) , 2)',

so the inverse transform is given by:

$$2\delta(t) + (2, -7)e^{(-2,1)t} + (2, 7)e^{(-2,-1)t} - 18e^{-3t},$$

where we see that the 2 if t = 0 corresponds to 2 times the Dirac delta function $\delta(t)$.

In general, **IXFRM** will only compute the finite portion of the inverse Laplace transform. See page 384 for ideas on how to automate the process of dealing with the Dirac delta function $\delta(t)$ part of the response.

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

LINEAR DIFFERENTIAL EQUATIONS

A linear ordinary differential equation of order r has the form

$$a_0(z)\frac{d^r w}{dz^r} + a_1(z)\frac{d^{r-1}w}{dz^{r-1}} + \ldots + a_r(z)w = f(z).$$

The general solution can be expressed as the sum of any particular integral and the general solution of the homogeneous linear differential equation

$$a_0(z)\frac{d^r w}{dz^r} + a_1(z)\frac{d^{r-1}w}{dz^{r-1}} + \ldots + a_r(z)w = 0.$$

The superposition theorem applies, and every linear combination of solutions is also a solution. The r solutions $w_k(z)$ are linearly independent in domain D if and only if the Wronskian determinant differs from zero for all $z \in D$.

$$W(w_1, w_2, \ldots w_r) = \begin{vmatrix} w_1(z) & w_2(z) & \cdots & w_r(z) \\ w_1^{(1)}(z) & w_2^{(1)}(z) & \cdots & w_r^{(1)}(z) \\ \vdots & \vdots & \ddots & \vdots \\ w_1^{(r-1)}(z) & w_2^{(r-1)}(z) & \cdots & w_r^{(r-1)}(z) \end{vmatrix}$$

SDET provides the capability to compute Wronskians.

In the case where the linear differential equations have constant coefficients, the homogeneous solution is probably most easily written down by inspection. However, it is instructive to understand that **IXFRM** will compute it also. For example, consider the homogeneous linear differential equation on the next page.

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

$$\frac{d^4w(z)}{dz^4} - 5\,\frac{d^3w(z)}{dz^3} + 9\,\frac{d^2w(z)}{dz^2} - 7\,\frac{dw(z)}{dz} + 2\,w(z) = 0.$$

In differential operator notation using D to denote differentiation, the equation is

$$[\ D^4 - 5\ D^3 + 9\ D^2 - 7\ D + 2\ ]\ w(z) = 0,$$

which can be written in Laplace transform notation as

$$[\ s^4 - 5\ s^3 + 9\ s^2 - 7\ s + 2\ ]\ W(s) = 0.$$

The root vector may be computed using **AROOT** and equals [1 1 1 2].

Letting F = '((C1×(s−1)^2+C2×(s−1)+C3×2!)×(s−2)+C4×(s−1)^3)×e^(s×t)', α = s, D = [1 1 1 2], and ε = 1E−10, we obtain from **IXFRM** the solution

'LN(e)^2×t^2×C3×e^t + LN(e)×t×C2×e^t + C1×e^t + C4×e^(2×t)'.

Observing that LN(e) = 1, the solution is

$$w(t) = C1\ e^{t} + C2\ t\ e^{t} + C3\ t^2\ e^{t} + C4\ e^{2t}.$$

Given the initial conditions w(0) = W0, $w^{(1)}(0)$ = W1, $w^{(2)}(0)$ = W2, and $w^{(3)}(0)$ = W3, the coefficients C1, C2, C3, and C4 can be computed. The below program first computes the symbolic equation for the solution w(t). Then it computes equations for the first three derivatives of w(t). All four equations are evaluated at zero and the symbolic equations stored in a list L. Then the **SGRD** and **L1F1** commands are used to create a symbolic coefficient matrix for the unknown coefficients. The matrix is then inverted, yielding equations for the coefficients in terms of the initial conditions. The program IXFRM3 is given on the next page.

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

```
IXFRM3: « '((C1×(s-1)^2+C2×(s-1)+C3×2!)×(s-2)+C4×(s-1)^3)×e^(s×t)' s
[ 1 1 1 2 ] → F α D « F α D .0000000001 IXFRM HALT { 'LN(e)' 1 }
↑MATCH DROP COLCT HALT { } → W L « W { t 0 } | 'L' STO+ W 1 3
START t ∂ { 'LN(e)' 1 } ↑MATCH DROP EXCO DUP { t 0 } | 'L' STO+
NEXT DROP L LREV « { C1 C2 C3 C4 } SGRD » L1F1 M←SO INV 10
RND M→SO { W0 W1 W2 W3 } SMPY L←SO HALT 'L' STO W 1 4 FOR
K "C" K + OBJ→ L K GET 2 →LIST ↑MATCH DROP NEXT EXCO » » »
```

After the coefficients are computed, they are substituted into the equation for w(t) and the result returned. Be sure that Flag –2 is clear so that e does not evaluate.

```
't^2×W0×e^t – 2.5×t^2×W1×e^t + 2×t^2×W2×e^t – .5×t^2×W3×e^t – 2×t×W1×e^t +
3×t×W2×e^t – t×W3×e^t – W0×e^(2×t) + 3×W1×e^(2×t) – 3×W2×e^(2×t) +
W3×e^(2×t) + 2×W0×e^t – 3×W1×e^t + 3×W2×e^t – W3×e^t'
```

Now suppose W0 = 2, W1 = –3, W2 = 4, and W3 is not specified, but w(1) = 8. The following program computes w(t) in terms of these specified parameters.

```
IXFRM4: « { W0 2 W1 –3 W2 4 } | DUP –2 SF EVAL –2 CF { t 1 } |
EXCO 8 = W3 ISOL EQ→ 2 →LIST | EXCO »
```

The result is given by:

```
'6.1304837674×t^(2)×e^t – 4.7390324652×t×e^t – .2609675348×e^(2×t) +
2.2609675348×e^t'
```

Other initial and boundary conditions can be handled similarly.

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

INVERSE LAPLACE TRANSFORM DIFFERENTIAL EQUATION SOLUTIONS

Solve w''(t) + 2 w'(t) + 5 w(t) = e^{-t} sin t, with w(0) = 0 and w'(0) = 1. The first thing we need is the Laplace transform of e^{-t} sin t. This is most easily read out of a table, but it is instructive to actually compute it with IXFRM5. Observing that

$$e^{-t} \sin t = [\ e^{(-1,1)t} - e^{(-1,-1)t}\ ]/(0,\ 2),$$

IXFRM5 integrates the Laplace transform integral definition from 0 to T. The ↑**MATCH** operation essentially takes the limit as T → ∞. The following two ↓**MATCH** commands algebraically isolate the numerator and denominator. **AROOT** is then used to compute the roots of the denominator polynomial.

IXFRM5: « 0 T 'EXP(((−1,1)−s)×t)/(0,2) − EXP(((−1,−1)−s)×t)/(0,2)' t ∫ EVALC { 'EXP(&×T)' 0 } ↑MATCH DROP COLCT DUP { '&A/(&B−s)' '&A×(CONJ(&B)−s)' } ↓MATCH DROP EXCO SWAP { '&A/&B+&C/&D' '&B×&D' } ↓MATCH DROP EXCO DUP s 4 COEFL 0 .0000000001 100 AROOT »

$$\mathcal{L}\{w''(t)\} + 2\ \mathcal{L}\{w'(t)\} + 5\ \mathcal{L}\{w(t)\} = \mathcal{L}\{e^{-t} \sin t\ \}$$

$$\{\ s^2\ W(s) - sw(0) - w'(0)\ \} + 2\ \{\ s\ W(s) - w(0)\ \} + 5\ W(s) = \frac{1}{s^2 + 2s + 2}$$

$$W(s) = \frac{s^2 + 2s + 3}{s^4 + 4s^3 + 11s^2 + 14s + 10}$$

From **AROOT**, the denominator root vector is [(−1,1) (−1,−1) (−1,−2) (−1,2)]. The program on the next page uses **IXFRM** to compute the solution.

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

« 's^2+2×s+3' 'e^(s×t)' × s [(–1,1) (–1,–1) (–1,–2) (–1,2)] .0000000001 IXFRM
»

The result is '(0,–1/6)×e^((–1,1)×t) + ((0,1/6)×e^((–1,–1)×t) + ((0,1/6)×e^((–1.–2)×t) + (0.–1/6)×e^((–1,2)×t)))' = 1/3 e^{-t}(sin t + sin 2t).

SIMULTANEOUS DIFFERENTIAL EQUATION SOLUTION EXAMPLE

Solve x'(t) = 2x(t) – 3y(t) and y'(t) = y(t) – 2x(t) with initial conditions x(0) = 8 and y(0) = 3. Taking the Laplace transform yields the linear system

$$\begin{bmatrix} s-2 & 3 \\ 2 & s-1 \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} 8 \\ 3 \end{bmatrix}$$

This system can be solved either using Cramer's rule or the symbolic matrix inverse utility command **SMI** given below. By Cramer's rule we have

$$X(s) = \frac{\begin{vmatrix} 8 & 3 \\ 3 & s-1 \end{vmatrix}}{\begin{vmatrix} s-2 & 3 \\ 2 & s-1 \end{vmatrix}} = \frac{8s - 17}{s^2 - 3s - 4},$$

$$Y(s) = \frac{\begin{vmatrix} s-2 & 8 \\ 2 & 3 \end{vmatrix}}{\begin{vmatrix} s-2 & 3 \\ 2 & s-1 \end{vmatrix}} = \frac{3s - 22}{s^2 - 3s - 4},$$

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

so $x(t) = 5e^{-t} + 3e^{4t}$ and $y(t) = 5e^{-t} - 2e^{4t}$.

The following program computes these solutions using Cramer's rule.

≪ { { 's-2' 3 } { 2 's-1' } } SDET s 4 COEFL 0 .0000000001 100 AROOT 'e^(s×t)' → V K ≪ { { 8 3 } { 3 's-1' } } SDET EXCO K × s V .0000000001 IXFRM { { 's-2' 8 } { 2 3 } } SDET EXCO K × s V .0000000001 IXFRM ≫ ≫

COEFFICIENTS	**COEFL**(F,V,N)	F V N	COEFFICIENTS

Reduces function F in variable V to a coefficient list of size N + 1, which is its Maclaurin series expansion. See { MISC } for more detail.

COEFFICIENT LIST	**CLIST**(V)	[VECTOR]	LIST

Given vector V containing the roots of some polynomial, **CLIST** computes the corresponding polynomial list.

SM AND SV ROUND	**SRND**(L,N)	L ∈ SM OR SV OR 𝔸	ROUNDED SM

Given SM L of numbers and integer N, **SRND** applies ≪ N RND ≫ to each element of L.

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
SYMBOLIC MATRIX INVERSE UTILITY	**SMI**(M, α) α ∈ [0, 1, 2]	M ∈ SM OR 𝔸 SQUARE M SIZE(M) ≥ {2 2}	2: D = DENOMINATOR 1: N = NUMERATOR

For α = 0, D is the determinant of M, N is the adjugate of M, and N/D is the inverse of M. For α = 1, 2, **SMI** returns polynomial lists for computing $(s\,I - M)^{-1}$, which is the resolvent matrix used for solving state space differential equations by Laplace transform techniques using Leverrier's algorithm. **SMI** is significantly faster when M is an ordinary square HP 48 matrix.

The next few pages will explain this rather complicated command.

STATE SPACE LINEAR DIFFERENTIAL EQUATION APPLICATIONS

There was no room on the ROM to program the following example techniques, but the user is invited to write his or her own applications programs using the below techniques. These techniques demonstrate some of the power that is built into the Math Library. The numerical Runge–Kutta solution approach was illustrated above, so we focus on symbolic solutions.

Find the state transition matrix Φ(t) for the system defined by the vector differential equation

$$\frac{dx}{dt} = \mathbf{A}\,\mathbf{x} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -4 & 4 \\ 0 & -1 & 0 \end{bmatrix} \mathbf{x} \quad \Rightarrow \quad \frac{d\Phi(t)}{dt} = \mathbf{A}\,\Phi(t) \quad \Rightarrow \quad \Phi(t) = e^{\mathbf{A}t}.$$

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

STATE SPACE EXAMPLE – TAYLOR SERIES EVALUATION

The simplest way to obtain a numerical solution is by evaluating the first N terms of the Taylor series expansion of e^{At}. The following program "EATN" does this:

« → A α N « A SIZE EVAL DUP IDN → R C I « IF 'R≠C' THEN 1281 DOERR END A α × 'A' STO I N 1 FOR K A × K / I + −1 STEP » » ».

$$\text{EATN(A,5,100)} = \begin{bmatrix} 0.006738 & 0 & 0 \\ 0 & -0.0004086 & 0.0009081 \\ 0 & -0.0002270 & 0.0004994 \end{bmatrix}$$

which is the value of Φ(5) with an error of 7.8E−8. The program "EATE" evaluates the exact solution Φ(t):

« → α « { { 'EXP(−t)' 0 0 } { 0 '(1−2×t)×EXP(−2×t)' '4×t×EXP(−2×t)' } { 0 '−t×EXP(−2×t)' '(1+2×t)×EXP(−2×t)' } } t α 2 →LIST SVBAR M←SO » ».

STATE SPACE EXAMPLE – ALGEBRAIC SERIES EXPANSION METHOD

The first step is to evaluate **SCHRD**(A,1E−10,0,20). The eigenvalues of A are the diagonal elements of the output matrix and equal −1, −2, and −2. By the Jordan Decomposition Theorem, the elements of Φ(t) must be a linear combination of e^{-t}, e^{-2t}, and te^{-2t}, where the last function is associated with the repeated −2 eigenvalue. The program "EAT" algebraically computes the first few terms of Φ(t). Most of the execution time is the algebraic cleanup by SEXCO.

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

EAT: « → A α N « A SIZE EVAL DUP IDN M→SO → R C I « IF 'R≠C' THEN 1281 DOERR END A M→SO α SMPY 'A' STO I N 1 FOR K A SMPY K INV SMPY I SADD -1 STEP SEXCO »»»

Using **TALR1** it may be verified that:

$$e^{-t} = 1 - t + \frac{1}{2}t^2 - \frac{1}{6}t^3 + \frac{1}{24}t^4 - \bullet\bullet\bullet,$$

$$e^{-2t} = 1 - 2t + \frac{4}{2}t^2 - \frac{8}{6}t^3 + \frac{16}{24}t^4 - \bullet\bullet\bullet.$$

Use EAT(A,t,4) to compute the first few terms of the Taylor expansion of EXP(**A**t).

$$e^{At} = \begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix} + \begin{bmatrix}-t&0&0\\0&-4t&4t\\0&-t&0\end{bmatrix} + \begin{bmatrix}t^2/2&0&0\\0&6t^2&-8t^2\\0&2t^2&-2t^2\end{bmatrix} + \bullet\bullet\bullet\bullet = I + A\frac{t}{1!} + A^2\frac{t^2}{2!} + \bullet\bullet\bullet$$

$$= \begin{bmatrix} 1-t+\frac{1}{2}t^2-\frac{1}{6}t^3+\frac{1}{24}t^4-\bullet\bullet\bullet & 0 & 0 \\ 0 & 1-4t+6t^2-\frac{16}{3}t^3+\frac{4}{3}t^4-\bullet\bullet\bullet & 4t-8t^2+8t^3-\bullet\bullet\bullet \\ 0 & -t(1-2t+2t^2-\frac{4}{3}t^3+\bullet\bullet\bullet & 1-2t^2+\frac{8}{3}t^3-2t^4+\bullet\bullet\bullet \end{bmatrix}$$

Recognizing the series expression for e^{-t} and e^{-2t} gives the solution:

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

$$e^{At} = \begin{bmatrix} e^{-t} & 0 & 0 \\ 0 & (1-2t)e^{-2t} & 4t\,e^{-2t} \\ 0 & -t\,e^{-2t} & (1+2t)e^{-2t} \end{bmatrix}$$

where the math trick on e^{-2t} terms is to use the Jordan result to prove each of the A_{jk} for j, k = 2, 3 must in fact have the form $(a_{jk} + b_{jk}t)e^{-2t}$, so one simply arranges the terms to solve for a_{jk} and b_{jk}. The program EATE above evaluates this equation. More directly, use SCHRD(A,1E-10,1,20) to perform the Schur decomposition of $A = Q\,T\,Q^H$.

$$T = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -2 & -5 \\ 0 & 0 & -2 \end{bmatrix} \qquad Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & .8944 & .4472 \\ 0 & .4472 & -.8944 \end{bmatrix}$$

The Schur decomposition is as close to Jordan form as one can get with unitary transformations. The required functions are included in the Math Library to implement the Bartels–Stewart algorithm, which will transform the Schur into the non-unitary Jordan decomposition:

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 \\ 0 & -2 & 1 \\ 0 & 0 & -2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$

Computation of Jordan canonical form is discussed in Chapters 20 and 27.

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

STATE SPACE EXAMPLE – RESOLVENT MATRIX (LAPLACE TRANSFORM) APPROACH

Laplace transforming the state differential equation gives the below equation for the inverse of the resolvent matrix R(s):

$$\mathbf{R}^{-1}(s) = [s\mathbf{I}-\mathbf{A}] = \begin{bmatrix} s+1 & 0 & 0 \\ 0 & s+4 & -4 \\ 0 & 1 & s \end{bmatrix}$$

SMNR and **SDET** may be used to evaluate the cofactors and the determinant of $\mathbf{R}^{-1}(s)$, yielding the inverse. The below program "RESOV" performs this symbolic matrix inverse with inputs A and s:

« → A α « A SIZE EVAL DUP IDN M→SO → R C I « IF 'R≠C' THEN 1281 DOERR END I α SMPY A M→SO SSUB 'A' STO 1 R FOR J 1 C FOR K A J K SMNR SDET '(-1)^(J+K)' →NUM × I K J 2 →LIST ROT SPUT 'I' STO NEXT NEXT I A SDET » » ».

$$\mathbf{R}(s) = \frac{1}{(s+1)(s+2)^2}\begin{bmatrix} (s+2)^2 & 0 & 0 \\ 0 & s(s+1) & 4(s+1) \\ 0 & -(s+1) & (s+1)(s+4) \end{bmatrix}$$

$$= \frac{1}{s+1}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \frac{1}{(s+2)^2}\begin{bmatrix} 0 & 0 & 0 \\ 0 & s & 4 \\ 0 & -1 & s+4 \end{bmatrix}$$

Perform a partial fraction expansion by evaluating the coefficients with **HVSDE**.

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

$$\mathbf{R(s)} = \frac{1}{s+1}\begin{bmatrix}1&0&0\\0&0&0\\0&0&0\end{bmatrix} + \frac{1}{s+2}\begin{bmatrix}0&0&0\\0&1&0\\0&0&1\end{bmatrix} + \frac{1}{(s+2)^2}\begin{bmatrix}0&0&0\\0&-2&4\\0&-1&2\end{bmatrix}$$

Summing and inverse transforming using **RESDA** give the desired result.

STATE SPACE EXAMPLE – LEVERRIER's THEOREM AND SYMBOLIC MATRIX INVERSION

The example on the previous page demonstrated the inverse Laplace transform solution approach. Now we explain how the **SMI** command can be used to both generalize and simplify the solution.

Define the $n \times n$ complex matrices $F_1, F_2, \ldots, F_n$ and the scalars $\theta_1, \theta_2, \ldots, \theta_n$ where $n \geq 2$ as follows:

$$F_1 = I, \quad \theta_1 = -\text{Tr } AF_1/1, \quad F_2 = AF_1 + \theta_1 I, \quad \theta_2 = -\text{Tr } AF_2/2, \ldots,$$
$$F_n = AF_{n-1} + \theta_{n-1} I, \quad \theta_n = -\text{Tr } F_n/n$$

where Tr denotes the trace operation and I is the identity matrix. Then:

$$R(s) = (sI - A)^{-1} = \frac{s^{n-1}F_1 + s^{n-2}F_2 + \bullet\bullet\bullet + sF_{n-1} + F_n}{s^n + \theta_1 s^{n-1} + \bullet\bullet\bullet + \theta_{n-1}s + \theta_n} \qquad AF_n + \theta_n I = 0,$$

$$\text{DET } (sI - A) = s^n + \theta_1 s^{n-1} + \ldots + \theta_{n-1}s + \theta_n = (s-\lambda_1)(s-\lambda_2)\ldots(s-\lambda_n),$$

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

where the λ_j are the eigenvalues of A and the poles of the system.

For s = 0, we also have a method for computing the symbolic inverse $A^{-1} = -F_n/\theta_n$, where $-\theta_n$ is the determinant of A. Thus, Leverrier's Theorem gives us an alternative method for performing the matrix inversion associated with the resolvent matrix approach. **SMI** implements this algorithm.

The adjugate or non–Hermitian adjoint of a matrix A is the transpose of the cofactors of A. F_n is the adjugate of A, and by Cramer's rule A^{-1} = ADJ A/DET A. With these definitions, we have

$$\text{SMI(A,0)} = \begin{cases} 1\text{: DET A} \\ 2\text{: ADJ A} \end{cases}$$

$$\text{SMI}\left(\begin{bmatrix} -1 & 0 & 0 \\ 0 & -4 & 4 \\ 0 & -1 & 0 \end{bmatrix}, 0\right) = \begin{cases} 2\text{: } -4 \\ 1\text{: } \begin{bmatrix} 4 & 0 & 0 \\ 0 & 0 & 4 \\ 0 & -1 & 4 \end{bmatrix} \end{cases} \qquad A^{-1} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & .25 & -1 \end{bmatrix}$$

$$\text{SMI}\left(\begin{bmatrix} s+1 & 0 & 0 \\ 0 & s+4 & -4 \\ 0 & 1 & s \end{bmatrix}, 0\right) = \begin{cases} 2\text{: } s^3+5s^2+8s+4 \\ 1\text{: } \begin{bmatrix} s^2+4s+4 & 0 & 0 \\ 0 & s^2+s & 4s+4 \\ 0 & -s-1 & s^2+5s+4 \end{bmatrix} \end{cases}$$

after algebraic cleanup with **EXCO** and **SEXCO** (this takes a long time). This is the result obtained in the previous resolvent matrix example.

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

A much faster way to compute the same result is SMI(A,1) and SMI(A, 2), which avoid symbolic algebraics and work with polynomial lists. Specifically, the above result can be written as:

$$R(s) = \frac{s^2\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix} + s\begin{bmatrix}4&0&0\\0&1&4\\0&-1&5\end{bmatrix} + \begin{bmatrix}4&0&0\\0&0&4\\0&-1&4\end{bmatrix}}{s^3 + 5s^2 + 8s + 4}$$

SMI(A,2) returns the numerator matrices in a list that is a matrix polynomial list for $A \in \mathbb{A}$ and a corresponding list of symbolic vectors for $A \in SM$. *NOTE THAT* **SMI** *RETURNS A MEANINGLESS RESULT FOR 1 x 1 MATRICES: SIZE(A) ≥ {2 2}.*

Computation of the inverse Laplace transform requires the evaluation of the 9 residues represented by the 9 numerator polynomials divided by the denominator polynomial. SMI(A,1) computes these 10 polynomials directly for ease of residue integration with **RESDP** and **RESDA**. The result is

$$\mathbf{SMI}\left(\begin{bmatrix}-1&0&0\\0&-4&4\\0&-1&0\end{bmatrix}, 1\right) = \begin{cases}2: D\\ \{N_{11} & N_{12} & \cdots & N_{1n}\\ 1: N_{21} & N_{22} & \cdots & N_{2n}\\ \vdots & \vdots & \cdots & \vdots\\ N_{n1} & N_{n2} & \cdots & N_{nn}\}\end{cases} = \begin{cases}2: \{4\ 8\ 5\ 1\}\\ \{\{4\ 4\ 1\}\ \{0\ 0\ 0\}\\ \{0\ 0\ 0\}\ \{0\ 0\ 0\}\\ 1: \{0\ 1\ 1\}\ \{4\ 4\ 0\}\\ \{0\ 0\ 0\}\ \{-1\ -1\ 0\}\\ \{4\ 5\ 1\}\}\end{cases}$$

where { 4 4 1 } / { 4 8 5 1 } is the $R_{11}(s)$ polynomial ratio and { 4 4 0 } / { 4 8 5 1 } is the $R_{23}(s)$ polynomial ratio. Solution of these nine polynomial ratios yields the nine scalar differential equation solutions which constitute the desired matrix differential equation solution Φ. See the FILTR menu in Chapter 27 for additional commands and discussion on State Space.

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

STATE SPACE EXAMPLE – COMPLETE SOLUTIONS

The previous examples have illustrated the theory associated with solving symbolic state space differential equations. Using the command **L1F1** defined below, the following program EASOV provides the complete solution. A similar program is given in Chapter 28 for discrete state space systems.

```
EASOV: ≪ → A ≪ A SIZE → S ≪ IF S { 1 1 } SAME THEN 515 DOERR
  END A 1 SMI SWAP 0 .0000000001 100 AROOT 'V' STO ≪ s XEQN
  'e^(s×t)' × s V .0000000001 IXFRM ≫ L1F1 'V' PURGE { 'LN(e)' 1 } MAT↑
  { 'LN(INV(e))' -1 } MAT↑ SEXCO SOB→ DROP S →SOB ≫ ≫ ≫
```

Consider the following complex A matrix:

$$A = \begin{bmatrix} (-1,2) & (0,0) & (0,0) \\ (0,0) & (0,-4) & (-4,2) \\ (0,0) & (2,-1) & (-6,-1) \end{bmatrix}$$

Then the solution is

$$\text{EASOV(A)} = e^{At} = \begin{bmatrix} e^{(-1,2)t} & 0 & 0 \\ 0 & 2e^{(-2,-3)t} - e^{(-4,-2)t} & -2e^{(-2,-3)t} + 2e^{(-4,-2)t} \\ 0 & e^{(-2,-3)t} - e^{(-4,-2)t} & -e^{(-2,-3)t} + 2e^{(-4,-2)t} \end{bmatrix}$$

The following program will test the solution by computing the differential equation.

```
TEA: ≪ → A EA v ≪ EA v SDERV { 'LN(e)' 1 } MAT↑ { 'LN(INV(e))' -1 }
  MAT↑ SEXCO A M→SO EA SMPY SSUB SEXCO ≫ ≫
```

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS
REAL PART OF SM	**SRE**(M)	M ∈ SM OR SV OR 𝔸	REAL PART
IMAGINARY PART OF SM	**SIM**(M)	M ∈ SM OR SV OR 𝔸	IMAGINARY PART
TEST IF REAL	**TIFRE**(M)	M ∈ SM OR SV OR 𝔸	SM OR SV OR 𝔸

TIFRE tests its argument M to see if every element has an imaginary part whose magnitude is less than 1E–8. If every element does, then **TIFRE** returns the real part of M. M may be a standard HP 48 vector or matrix object. It can also be a list of numbers or a symbolic vector or matrix.

SRE, **SIM**, **TIFRE**, and **SCNJ** also work for real and complex number arguments.

SYMBOLIC SIZE	**SSIZE**(M)	M ∈ SM OR SV OR 𝔸	SIZE
SYMBOLIC CONJ	**SCNJ**(M)	M ∈ SM OR SV OR 𝔸	M_{jk} CONJ
LIST ELEMENT OPERATIONS	**L1F1**(L,P)	2: LIST 1: PROGRAM	L_j « P » EVAL

L1F1 is similar to **S1F1** but it specifically treats the input L as a list. **L1F1** provides the capability to apply program P to the elements of lists, including when the elements are lists but L is not a symbolic matrix. See the previous page for an example usage of **L1F1**.

UP DIRECTORY	**UPDIR**	NONE	PARENT MENU

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

VECTOR INTEGRATION – LINE INTEGRALS

Compute the total work done in moving a particle in a force field given by **F** = 3xy**i** – 5z**j** + 10x**k** along a curve $x = t^2 + 1$, $y = 2t^2$, $z = t^3$ from t = 1 to t = 3. TOTAL WORK = $\int_P$ **F**•**dr**. Type in the following program and store it in a variable LI:

« → F r P L U v « L U F 1 3 FOR K r K GET P K GET 2 →LIST MAT↑ SCOLC NEXT P v SDERV SDOT EXCO v ∫ EVALC » ».

Define force F = { '3xy' '–5z' '10x' }, position r = { x y z }, path P = { 't^2+1' '2t^2' 't^3' } and put them on the stack in that order. Next put 1, 3, and 't' on the stack and evaluate the program to get 2440 for the total work done. Other line integrals can be computed similarly.

VECTOR INTEGRATION – VECTOR VOLUME INTEGRALS

Let **F** = 2xz**i** – x**j** + y^2**k**. Evaluate $\iiint_R$ **F** dV over the volume defined by the region R bounded by the surfaces x = 0, y = 0, y = 6, $z = x^2$, and z = 4. Type in the following program:

« → F r R « F 1 3 FOR K R K SERW EVAL r K GET SINTG SEXCO NEXT » ».

Place on the stack the symbolic force vector { '2xz' '–x' 'y^2' }, the variable vector { z y x } which also specifies the order of integration, and the symbolic matrix { {'x^2' 4} {0 6} {0 2} } which defines the integration limits in the same order as the variable vector. Now evaluate the program to obtain the vector volume integral { 128 –24 384 } or 128**i** – 24**j** + 384**k**.

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

Bellman, R., *Introduction to Matrix Analysis*, New York, McGraw–Hill, 1970.

Brockett, R., *Finite Dimensional Linear Systems*, New York, Wiley, 1970.

Bronson, R., *Matrix Methods*, Boston, Academic Press, 1991.

Bryson, A., and Ho, Y., *Applied Optimal Control*, Waltham: MA, Blaisdell Publishing Co. 1969.

Byerly, W., *An Introduction to the Use of Generalized Coordinates in Mechanics and Physics*, New York, Dover, reprint of 1916 edition, 1965.

Chen, C., *Introduction to Linear System Theory*, New York, Holt Rinehart and Winston, 1970.

Courant, R., and Hilbert, D., *Methods of Mathematical Physics*, Volume 1, New York, Interscience Publishers, 1965.

Courant, R., and Hilbert, D., *Methods of Mathematical Physics*, Volume 2, New York, Interscience Publishers, 1966.

Derusso, P., Roy, R., and Close, C., *State Variables for Engineers*, New York, Wiley, 1966.

Dunford, N., and Schwartz, J., *Linear Operators, Part I and II*, New York, Interscience Publishers, 1957 and 1963.

Erdelyi, A., Magnus, W., Oberhettinger, F., and Tricomi, F., *Tables of Integral Transforms*, 2 Volumes, New York, McGraw–Hill, 1954.

Forsyth, A., *Theory of Differential Equations*, 6 Volumes, New York, Dover, 1959.

Garabedian, P., *Partial Differential Equations*, New York, Wiley, 1964.

Gardner, M., and Barnes, J., *Transients in Linear Systems*, New York, Wiley, 1942.

Goertzel, G., and Tralli, N., *Some Mathematical Methods of Physics*, New York, McGraw–Hill, 1960.

Gohberg, I., and Krein, M., *Theory and Applications of Volterra Operators in Hilbert Space*, Providence: RI, American Mathematical Society, 1970.

Kaplan, M., *Modern Spacecraft Dynamics & Control*, New York, Wiley, 1976.

Kellogg, O., *Foundations of Potential Theory*, New York, Dover, 1953.

Korn, G., and Korn, T., *Mathematical Handbook For Scientists and Engineers*, New York, McGraw–Hill, 1961.

SYMBOLIC VECTORS & MATRICES PLUS ADVANCED CALCULUS { SYMB }

FUNCTION	COMMAND	INPUTS	OUTPUTS

Magnus, J., and Neudecker, H., *Matrix Differential Calculus with Applications in Statistics and Econometrics*, New York, Wiley, 1988.
Maxwell, J., *A Treatise on Electricity and Magnetism*, 2 Volumes, New York, Dover, 1954 based on 1891 edition.
Misner, C., Thorne, K., and Wheeler, J., *Gravitation*, San Francisco, Freeman, 1973.
Morse, P., and Feshbach, H., *Methods of Theoretical Physics*, New York, McGraw–Hill, 1953.
Paley, R., and Wiener, N., *Fourier Transforms in the Complex Domain*, Providence: RI., American Mathematical Society, 1934.
Papoulis, A., *The Fourier Integral and its Applications*, New York, McGraw–Hill, 1962.
Pars, L., *A Treatise on Analytical Dynamics*, London, Heinemann, 1965.
Rayleigh, J., *The Theory of Sound*, 2 Volumes, New York, Dover, 1945 based on 1877 edition.
Schiff, L., *Quantum Mechanics*, New York, McGraw–Hill, 1955.
Stakgold, I., *Boundary Value Problems of Mathematical Physics*, Volume 1, London, Macmillan, 1969.
Stakgold, I., *Boundary Value Problems of Mathematical Physics*, Volume 2, London, Macmillan, 1971.
Titchmarsh, E., *Theory of Fourier Integrals*, Oxford, Oxford Press, 1967.
Tolman, R., *The Principles of Statistical Mechanics*, Oxford, Oxford University Press, 1967.
Tricomi, F., *Integral Equations*, New York, Interscience Publishers, 1957.
Wiberg, D., *State Space and Linear Systems*, New York, McGraw–Hill, 1971.
Widder, D., *The Laplace Transform*, Princeton, Princeton University Press, 1941.
Zadeh, L., and Desoer, E., *Linear System Theory*, New York, McGraw–Hill, 1963.
Zadeh, L., and Polak, E., *System Theory*, New York, McGraw–Hill, 1969.

26

DATA PROCESSING AND SIMULATION

INTRODUCTION

This chapter presents the 48 data processing and simulation commands in the PROC menu. The PROC commands provide fast Fourier transforms (FFT), Wiener–Levinson least squares solutions for Toeplitz systems, design of least squares filters, convolution, correlation, data interpolation and decimation, peak and valley analysis, random and special waveform creation, numerical integration of waveforms, numerical differentiation of waveforms, plus a large number of processing tools.

Related commands are given in the FILTR and WIND menus for designing common filters and windows. Statistics functions are given in the STAT menu. Perfect reconstruction filter banks and wavelet transforms are discussed in Appendix G.

NUMERICAL DERIVATIVES AND INTEGRALS

DER1 and **DER2** provide numerical differentiation of waveforms and data contained in vectors. **DER1** uses five-point interpolation to accurately estimate the first derivative while **DER2** uses five-point interpolation to accurately estimate the second derivative. **LINT** provides numerical integration from the left, while **RINT** provides numerical integration from the right. Except at the end of the vectors, both integration commands use five-point interpolation for accuracy.

PEAK AND VALLEY DATA ANALYSIS

When analyzing data, it is often desirable to know the location of the next peak, valley, maximum, or minimum in the data. These functions are provided by the commands **FINDP**, **FINDV**, **FINDX**, and **FINDN**, respectively.

FAST FOURIER TRANSFORM ANALYSIS

FFT and **IFFT** provide forward and inverse fast Fourier transforms of data. The FFT twiddles are computed by the command **TWIDL**, and **BITRV** performs the FFT bit reversal that properly orders the output data. The FFT software is limited to powers of 2 in vector size. Programs DFT and IDFT provide discrete Fourier transforms of any vector size, but are slower.

FFT implements a forward discrete Fourier transform (DFT), and **IFFT** implements an inverse DFT.

$$\mathrm{DFT}(X)_n = y_n = \sum_{k=0}^{N-1} x_k e^{-i2\pi kn/N} \qquad \mathrm{IDFT}(Y)_k = x_k = \frac{1}{N} \sum_{n=0}^{N-1} y_n e^{i2\pi kn/N}$$

These are the standard electrical engineering definitions.

WIENER–LEVINSON SOLUTIONS FOR TOEPLITZ SYSTEMS

WL1 and **WL2** provide solutions to Hermitian Toeplitz linear systems. By exploiting Toeplitz symmetries, far larger linear systems can be solved than is numerically possible with the linear solvers given in the LINAG menu. A matrix R is Toeplitz if the values of its elements $R_{jk} = f(j - k)$ depend only on the difference between j and k. A matrix that is symmetric and Toeplitz has elements that only depend on the absolute value of this difference. Thus, the value of an element is uniquely determined by its distance from the upper left corner to lower right corner diagonal. As a result, all the data values are uniquely determined by either the first row or first column of R. Covariance matrices of complex stationary systems have the Hermitian Toeplitz property.

CONVOLUTION AND CORRELATION

CONV and **CCORR** provide the data convolution and correlation operations using the FFT software. The data sets need not be the same length since the data vectors are automatically zero-filled. **VMPY**, **VDVD**, and **VCORR** provide data convolution, deconvolution, and correlation without using the FFT commands. **ICONV** provides the convolution operation for IIR recursive filters. It can also be used to numerically solve difference equations.

DIGITAL INTERPOLATION AND DECIMATION

VDEC decimates data sets to a lower sampling rate. Offsets are selectable. **VINTP** performs the first step in digital data interpolation, that of zero filling. The interpolation process may then be completed using the FFT and filtering software. Examples are given in the FILTR menu.

RANDOM AND WAVEFORM VECTORS

RNDU and **RNDN** provide random waveforms with uniform and normal amplitude statistics. **RNDC** provides two jointly normal waveforms with a user-specified correlation coefficient. **RNDC** can also be used to create random complex waveforms.

Any waveform for which the Math Library or HP 48 has a function can be created using the **CSERS** command in the MISC menu. Squarewave and triangular waveforms are created by **SQWV** and **ΔWAV**. **INDEX** and **PINDX** provide amplitude and phase ramps.

PROCESSING TOOLS

An additional 16 commands are provided to simplify data processing operations. Commands include zero filling, data truncation, rotation, reflection, spectrum, phase, phase unwrapping, moving averages, and data deglitching. Programs for Hermitian Toeplitz matrix inversion and the power method of eigen analysis are given for spectral estimation. Two linear convolution programs are also provided.

DATA PROCESSING MENU {PROC}

FUNCTION	COMMAND	INPUTS	OUTPUTS
NUMERICAL FIRST DERIVATIVE	**DER1**(V)	V = [VECTOR] ∈ ℂ	[VECTOR]
NUMERICAL SECOND DERIVATIVE	**DER2**(V)	V = [VECTOR] ∈ ℂ	[VECTOR]

DER1 and **DER2** use five-point interpolation over the entire vector in order to maintain accuracy. SIZE(V) ≥ 7. The following program will provide a nice demo for these two commands:

```
« { { HOME } 'SIN(z)' z 0 10 100 } 0 CSERS DUP "DER1" 3 DISP DER1
  10 × SWAP "DER2" 3 DISP DER2 100 ×
  { (49,0) "10X" "COS(X) & -SIN(X)" } PLT2L »
```

NUMERICAL INTEGRATION FROM THE LEFT	**LINT**(V)	V = [VECTOR] ∈ ℂ	[VECTOR]
NUMERICAL INTEGRATION FROM THE RIGHT	**RINT**(V)	V = [VECTOR] ∈ ℂ	[VECTOR]

LINT and **RINT** use five-point interpolation except at the ends of the vector to maintain accuracy. SIZE(V) ≥ 5. The following program is a nice demo:

```
« { {HOME} 'SIN(z)' z 0 10 100 } 0 CSERS DUP "INTEGRATE" 3 DISP LINT .1
  × 1 VSUB {(49,0) "10X" "SIN(X) & -COS(X)"} PLT2L »
```

CREATE INDEX VECTOR	**INDEX**(N)	N = SIZE	[1 2 3 . . . N]

DATA PROCESSING MENU {PROC}

FUNCTION	COMMAND	INPUTS	OUTPUTS
CREATE PHASE INDEX VECTOR	**PINDX**(N)	N = SIZE	[(1,∠0) (1,∠1) (1,∠2) . . . (1,∠[N−1])]
INDEX(8) = [1 2 3 4 5 6 7 8] and PINDX(8) = [(1,0) (.5403,.8415) (−.4161,.9093) (−.99,.1411) (−.6536,−.7568) (.2837,−.9589) (.9602,−.2794) (.7539,.657)], which equals, when displayed in polar form, [(1,∠0) (1,∠1) (1,∠2) (1,∠3) (1,∠−2.2832) (1,∠−1.283) (1,∠−.2832) (1,∠.7168)]. In order to see the linear nature of the phase, evaluate VTRUR, which unwraps the phase in radians: 'VTRUR(PINDX(8))' EVAL to get the result [0 1 2 3 4 5 6 7].			
FIND NEXT PEAK STARTING AT s	**FINDP**(V,s)	V = [VECTOR] ∈ **R** OF SIZE N s ∈[1, N]	POSITION OF NEXT PEAK VALUE
FIND NEXT VALLEY STARTING AT s	**FINDV**(V,s)	V = [VECTOR] ∈ **R** OF SIZE N s ∈[1, N]	POSITION OF NEXT VALLEY
FIND MAXIMUM VALUE OF V IN RANGE n ∈[s, N]	**FINDX**(V,s)	V = [VECTOR] ∈ **R** OF SIZE N s ∈[1, N]	POSITION OF MAXIMUM VALUE
FIND MINIMUM VALUE OF V IN RANGE n ∈[s, N]	**FINDN**(V,s)	V = [VECTOR] ∈ **R** OF SIZE N s ∈[1, N]	POSITION OF MINIMUM VALUE
Observe that the outputs of the above four commands are the location, not the value. Let vector V = [1 5 3 4 2 1]. Then FINDP(V,2) = 4, FINDV(V,2) = 3, but FINDV(V,3) = NONE FOUND. Similarly, FINDX(V,3) = 4, FINDN(V,1) = 1, and FINDN(V,2) = 6.			
SUM OF VALUES	**VΣ**(V)	V = [VECTOR] ∈ **C**	$\Sigma\ (n=1, N, V_n)$

DATA PROCESSING MENU {PROC}

FUNCTION	COMMAND	INPUTS	OUTPUTS
CUMULATIVE SUM OF VALUES	**CUMΣ**(V)	V = [VECTOR] ∈ ℂ	CUMULATIVE SUM OF VALUES
Let V = [1 5 3 4 2 1]. Then VΣ(V) = 16 and CUMΣ(V) = [0 1 6 9 13 15 16]. Note that the SIZE of CUMΣ is 1 greater than V so that it can be used as a cumulative distribution function. The input to **V**Σ and **CUM**Σ may be symbolic vectors (lists), but in the case of **CUM**Σ, all the elements must be numbers because the output is a vector.			
FAST FOURIER TRANSFORM (FFT)	**FFT**(V,W,N) N = SIZE OF V & W	V = [VECTOR] ∈ ℂ W, N OUTPUT OF **TWIDL**	TRANSFORMED [VECTOR]
INVERSE FAST FOURIER TRANSFORM (IFFT)	**IFFT**(V,W,N) N = SIZE OF V & W	V = [VECTOR] ∈ ℂ W, N OUTPUT OF **TWIDL**	TRANSFORMED [VECTOR]
COMPUTE FFT TWIDDLE VECTOR	**TWIDL**(N)	N = TRANSFORM SIZE = 2^n	2: TWIDDLES 1: N
FFT BIT REVERSAL	**BITRV**(V,N)	V N = 2^n N = SIZE OF V	BIT REVERSED V
The size of the transform must be a power of 2, such as 4, 8, 16, 32, 64, 128, ... The twiddle vector may be precomputed and stored in memory in a variable W, for example. **BITRV** is called from the FFT so that the FFT output is in proper order. The order of the values is [x_0 x_1 x_2 x_3 . . . x_{N-1}]. As part of the computation, **BITRV** changes the HP 48 word size to LOG_2 N.			

DATA PROCESSING MENU {PROC}

FUNCTION	COMMAND	INPUTS	OUTPUTS
For example, create the program: « {64} 0 CON [1] 3 VREPL HALT 64 TWIDL IFFT PLTC » and store it in a variable FFT1. Now evaluate FFT1, and the calculator will halt with the vector: [0 0 1 0 0 ... 0] displayed. Press CONT, and the real and imaginary parts of the inverse transform will be plotted. The real part is a COS wave and the imaginary part is a SIN wave. They both have exactly 2 cycles because $x_2 = 1$. **BITRV** resets the word size to its original value when it finishes. However, if **BITRV** does not finish, you must manually reset it with: 64 STWS.			
WIENER–LEVINSON SOLUTION	**WL1**(V)	V = COVAR VECTOR V ∈ ℂ	2: h VECTOR 1: e VECTOR
WIENER–LEVINSON SOLUTION	**WL2**(RHS,V)	RHS VECTOR V = COVAR VECTOR ∈ ℂ	2: h VECTOR 1: e VECTOR

WL1 and **WL2** provide solutions to Hermitian Toeplitz systems of linear equations: R h = ϕ where R is an N × N Toeplitz matrix such as an autocovariance matrix, h is the unknown column vector, and ϕ is the known "right-hand side" (RHS) column vector. In statistical applications h represents the minimum mean square error filter associated with the Wiener–Hopt Equation defined by R and ϕ. The V argument in **WL1** and **WL2** is the first column of the R matrix. There are two cases:

WL1 provides the solution in the case where TRNP(ϕ) = [P_N 0 0 ... 0] and P_N is the resulting mean square error associated with filter solution h.

WL2 provides the solution in the general case where ϕ is equal to the RHS vector.

With both solutions the sequence of errors is returned to Level 1 of the stack in a vector. The size of V and RHS must be the same and be 2 or greater. Applications in spectral estimation and analysis are given at the end of this menu.

Examples are given on the next page and in Chapter 28.

DATA PROCESSING MENU {PROC}

FUNCTION	COMMAND	INPUTS	OUTPUTS

WL1 and **WL2** provide statistical least squares solutions. If you specify a singular (noise-free) deterministic problem, expect to get a *divide-by-zero error* during the calculation!

The program

« 0 15 FOR K 'EXP(-K^2)' →NUM NEXT 16 →ARRY DUP PLT1 HALT WL1 PLT1 PLT1 »

will create a covariance vector in Level 1 of the stack and HALT. Press CONT, and **WL1** will compute the solution and plot it. The errors in the solution will be plotted first. Then the solution is plotted. The first value of e is V(1), which must be real, and the last value of e is the number P_N. The next program illustrates **WL2**:

« 16 UNITI 0 15 FOR K 'EXP(-K^2)' →NUM NEXT 16 →ARRY DUP PLT1 HALT WL2 PLT1 PLT1 ».

FFT CONVOLUTION	**CONV**(V1,V2)	V1, V2 ∈ ℂ	[VECTOR]
FFT CROSS-CORRELATION	**CCORR**(V1,V2)	V1, V2 ∈ ℂ	[VECTOR]

CONV and **CCORR** both provide FFT convolution. In the case of **CCORR** the FFT of V2 is conjugated before multiplying the transforms. V1 and V2 do not need to be the same size. The larger vector is zero-filled to twice its size (and the next power of 2) and the smaller vector is zero-filled to the same size before Fourier transforming. The resulting product of transforms is then inverse-transformed. The result is a linear convolution and correlation since the circularly convolved values are deleted from the vector. See also **VMPY** and **VCORR** on pages 381 and 382.

CCORR([1 5 3 4 2 1], [1 2 3 4 5 6 7]) =
[7 41 56 75 76 68 52 36 21 11 4 1]

DATA PROCESSING MENU {PROC}

FUNCTION	COMMAND	INPUTS	OUTPUTS
RANDOM VECTOR WITH UNIFORM AMPLITUDE STATISTICS	**RNDU**(μ,σ,s,N)	μ = MEAN σ = STAND DEV s = SEED N = SIZE	RANDOM [VECTOR]
RANDOM VECTOR WITH NORMAL AMPLITUDE STATISTICS	**RNDN**(μ,σ,s,N)	μ = MEAN σ = STAND DEV s = SEED N = SIZE	RANDOM [VECTOR]
RNDU(1,2,11,8) creates an 8-dimensional vector with elements that are uniformly distributed with mean value of 1 and standard deviation of 2.			
POWER SPECTRUM	**SPECT**(V)	V = [VECTOR] $\in \mathbb{C}$	VECTOR: $\lvert V_j \rvert^2$
PHASE IN DEGREES	**PHASE**(V)	V = [VECTOR] $\in \mathbb{C}$	VECTOR: + ARG(V_j) $\times$ 180/π
SPECT([(1,2) (2,3) (3,4) (4,5)]) = [5 13 25 41] PHASE([(1,2) (2,3) (3,4) (4,5)]) = [63.4349 56.3099 53.1301 51.3402] Phase is simply the argument in degrees.			
ZERO FILL	**ZFILL**(V)	[VECTOR]	[VECTOR]
Doubles the length of V and fills the tail with zeros. ZFILL([1 2 3]) = [1 2 3 0 0 0]			

DATA PROCESSING MENU {PROC}

FUNCTION	COMMAND	INPUTS	OUTPUTS
TRUNCATE	**HALF**(V)	[VECTOR]	[VECTOR]
Truncates V of SIZE N to a length equal to the integer part of (N/2) = IP(N/2) for N > 1. HALF([1 2 3 0 0 0]) = [1 2 3]			
ZERO FILL	**ZFIL1**(V)	[VECTOR]	[VECTOR]
If N > 2 is the SIZE of V, **ZFIL1** creates a vector of length 2N – 2 and zero-fills the tail. ZFIL1([1 2 3]) = [1 2 3 0]			
TRUNCATE	**HALF1**(V)	[VECTOR]	[VECTOR]
Truncates V of SIZE N to length integer part of (N/2) + 1 = IP(N/2) + 1 for N > 2. HALF1([1 2 3 0]) = [1 2 3]			
ZERO FILL	**ZFILN**(V,n)	[VECTOR] n	[VECTOR]
Zero-fills vector V to a size n ≥ the SIZE of V. ZFILN([1 2 3] , 5) = [1 2 3 0 0]			
ZFILN, **REDN**, and **VROT** also work for symbolic vectors (lists).			
TRUNCATE	**REDN**(V,n)	[VECTOR] n	[VECTOR]

DATA PROCESSING MENU {PROC}

FUNCTION	COMMAND	INPUTS	OUTPUTS
Truncates vector V to length n. $0 < n \leq$ the size of V. REDN([1 2 3 0 0] , 3) = [1 2 3]			
VECTOR ROTATE	**VROT**(V,n)	[VECTOR] n	[VECTOR]
Rotates the elements of vector V. $n > 0$ is a rotate left and $n < 0$ is right. $\|n\| <$ SIZE of V VROT([1 2 3 4 5] , 2) = [3 4 5 1 2]			
VECTOR REFLECT	**REFLT**(V)	V = [VECTOR]	[VECTOR]
If N is the SIZE of V, **REFLT** creates a vector of length 2N – 2. The additional N – 2 values are: for j = N + 1 . . ., 2N – 2, $V_j = CONJ(V_{2N+2-j})$. $N \geq 3$. REFLT([(1,2) (2,3) (3,4) (4,5)]) = [(1,2) (2,3) (3,4) (4,5) (3,–4) (2,–3)]			
VECTOR DECIMATE	**VDEC**(V,n,F)	V = [VECTOR] n, F $\in$ $\mathbb{N}$	[VECTOR]
F is the decimation factor, and n is the position of first value to be retained. $0 < n \leq F$ always. If V = [1 2 3 ... 15] then **VDEC**(V,1,3) = [1 4 7 10 13] If V = [1 2 3 ... 15] then **VDEC**(V,2,3) = [2 5 8 11 14] If V = [1 2 3 ... 15] then **VDEC**(V,2,5) = [2 7 12]			

DATA PROCESSING MENU {PROC}

FUNCTION	COMMAND	INPUTS	OUTPUTS
VECTOR INTERPOLATE	**VINTP**(V,F)	V = [VECTOR] F ∈ **N**	[VECTOR]

Performs the first step in FFT interpolation, that of zero-filling. F > 1 must be an integer. For example, if
V = [1 4 7 10 13], then **VINTP**(V,3) = [1 0 0 4 0 0 7 0 0 10 0 0 13 0 0].

Other examples of decimation and interpolation are given with the **FTRV2** command in the FILTR menu discussed in Chapter 27.

SQUARE WAVE	**SQWV**(L)	L = LIST	[VECTOR]

Generates a rectangular waveform of values that are either zero or one. L = { T1 T2 T3 TS N }, where the times T1, T2, T3, and TS have the same units. T1 is the offset to the beginning of the first 1 value, T2 is the length of time that a 1 value persists, and T3 is the period of the waveform. N is the number of samples to be computed of the waveform, and TS is the sampling period of those samples. For example, the program « { 1.5 3 5 .1 100 } **SQWV PLT1** » will give a picture of the waveform. The reason the transition points are not perfectly vertical in the plot is associated with the interpolation scheme used by the plot program, and not the data itself.

TRIANGLE WAVE	**ΔWAV**(L)	L = LIST	[VECTOR]

Generates a triangular waveform of values that vary between zero and one. L = { T1 T2 T3 TS N }, where the times T1, T2, T3, TS have the same units. T1 is the offset to the first 0 value, T2 is the rise time, and T3 is the period of the waveform. N is the number of samples to be computed of the waveform, and TS is the sampling period of those samples. For example, the program « { 1.5 3 5 .1 100 } **ΔWAV PLT1** » will give a picture of the waveform.

DATA PROCESSING MENU {PROC}

FUNCTION	COMMAND	INPUTS	OUTPUTS
PHASE UNWRAP IN DEGREES	**PHASU**(V)	V = REAL [VECTOR]	UNWRAPPED PHASE VECTOR
PHASU provides phase unwrapping where the input vector contains phases in degrees, such as the output of **PHASE**. The values must be in the range [–180, 180]. **VTRUD** uses **PHASE** to compute the phase and then calls **PHASU** to do the unwrapping. For radian arguments use the **ARGU** and **VTRUR** commands in the VSAG menu discussed in Chapter 30.			
PHASE UNWRAP IN DEGREES	**VTRUD**(V)	V = COMPLEX [VECTOR]	UNWRAPPED PHASE VECTOR
VTRUD(PINDX(7)) = [0 57.2958 114.5916 171.8873 229.1831 286.4789 343.7747]			
MOVING AVERAGE	**MAVE**(V,n)	V = [VECTOR] ∈ ℂ	[VECTOR]
Performs a moving average on the data in V using n points in each average. If less than n points are available, then it uses the points that are available. MAVE([–1 1 –2 2 –1 1] , 2) = [–1 0 –.5 0 .5 0]			
CONVOLUTION	**VMPY**(V1,V2)	V1, V2 = [VECTORS]	[VECTOR]
DECONVOLUTION	**VDVD**(V1,V2)	V1, V2 = [VECTORS]	2: [QUOTIENT] 1: [REMAINDER]

DATA PROCESSING MENU {PROC}

FUNCTION	COMMAND	INPUTS	OUTPUTS
CORRELATION	**VCORR**(V1,V2)	V1, V2 = [VECTORS]	[VECTOR]

The above three commands provide convolution, deconvolution, and cross-correlation for complex vectors without use of the FFT commands. They are generally faster than FFT convolution when one of the two vectors is short, such as when performing a digital Hilbert transform.

VMPY([1 2 3] , [4 5 6]) = [4 13 28 27 18]

CONV provides the same result: [4 13 28 27 18], but takes longer.

$$\text{VDVD([4 13 28 27 18] , [1 2 3])} = \begin{cases} 2\text{: [4 5 6]} \\ 1\text{: [0]} \end{cases}$$

Convolution and deconvolution are mathematically equivalent to polynomial multiplication and division. The zero remainder in this example shows that the deconvolution is exact.

VCORR([1 2 3] , [4 5 6]) = [6 17 32 23 12]

where as with **CCORR** and **COVAR**, it is the second vector (V2) that has been conjugated. The corresponding output from **CCORR** is [6 17 32 23 12]. See the Hilbert transform demo program in the FILTR menu discussed in Chapter 27 which uses **VMPY** and the applications of **VCORR** in spectral estimation discussed on page 386 of this chapter.

IIR CONVOLVE	**ICONV**(N,D,Y,X)	N D Y X	[VECTOR]

Given the numerator N and denominator D, z transform polynomial lists in z^{-1}, initial condition vector Y = [y(–1) y(–2) ...] of size 1 less than D, and input vector X, **ICONV** convolves X with the filter and outputs the response. If X = [1 0 0 ... 0] and Y = [0 ... 0], then the output is the impulse response of the filter.

DATA PROCESSING MENU {PROC}

FUNCTION	COMMAND	INPUTS	OUTPUTS

Given the IIR filter defined by the difference equation

$$y(n) = -\sum_{k=1}^{N} D_{k+1}\, y(n-k) + \sum_{k=0}^{M} N_{k+1}\, x(n-k),$$

the one-sided (causal) z transform is given by

$$Y^{+}(z) = -\sum_{k=1}^{N} D_{k+1}\, z^{-k}\left[Y^{+}(z) + \sum_{n=1}^{k} y(-n)\, z^{n}\right] + \sum_{k=0}^{M} N_{k+1}\, z^{-k}\, X^{+}(z).$$

The response is the inverse z transform of the equation

$$Y^{+}(z) = \frac{\sum_{k=0}^{M} N_{k+1}\, z^{-k}}{A(z)}\, X^{+}(z) - \frac{\sum_{k=1}^{N} D_{k+1}\, z^{-k} \sum_{n=1}^{k} y(-n)\, z^{n}}{A(z)} \qquad A(z) = 1 + \sum_{k=1}^{N} D_{k+1}\, z^{-k}.$$

The following program, ICNVN, provides K values of the IIR convolution where input X is the impulse response of the filter specified by the polynomials NX and DX.

```
ICNVN: ≪ → N D Y NX DX K ≪ NX DX DUP SIZE 1 - ZERO K UNITI
       ICONV N D Y 4 ROLL ICONV ≫ ≫
```

The following program, TIRRC, will perform an IIR convolution using ICNVN and then will compute it symbolically using the program ICNVS on the next page.

```
TIRRC: ≪ → N D Y NX DX K ≪ N D Y NX DX K ICNVN → V ≪ N D Y
       NX DX ICNVS { HOME } OVER n 0 K 1 - K 6 →LIST 0 CSERS V ≫ ≫ ≫
```

If the unit step response of the filter is desired, then NX = { 1 } and DX = { 1 −1 }. Let N = { 1 }, D = { 1 −.9 .81 } corresponding to y(n) = .9y(n−1) − .81y(n−2) + x(n). With Y = [1 1], TIRRC correctly works Example 3.6.1 in Proakis and Manolakis twice. The answer in my edition of the textbook is in error.

DATA PROCESSING MENU {PROC}

FUNCTION	COMMAND	INPUTS	OUTPUTS

Using command **IXFRM** discussed in Chapters 25 and 27, allows the same result to be computed symbolically. Program ICNVS performs the computation.

```
ICNVS: « → N D Y NX DX « N NX PMPY D DX PMPY PZINV SWAP
OVER PDVD DROP z XEQN 'z^(n-1)' × z 4 ROLL 0 1E-10 100 AROOT
1E-10 IXFRM IF OVER SABS 0 > THEN 'n>0' SWAP ROT EVAL IFTE
ELSE SWAP DROP END IF Y ABS 0 > THEN D SIZE 1 - → E S « Y
1E-90 VADD D LREV 1 S VSUBS PMPY 1 S VSUBS LREV D PZINV
SWAP OVER PDVD DROP z XEQN 'z^(n-1)' × z 4 ROLL 0 1E-10 100
AROOT 1E-10 IXFRM IF OVER SABS 0 > THEN 'n>0' SWAP ROT EVAL
IFTE ELSE SWAP DROP END E SWAP - » END » »
```

Program ICNVS demonstrates several techniques. First of all, it automates the proper handling of improper rationals, e.g., whether the symbolic residue integration performed by **IXFRM** provides the value for n = 0, or it must be computed by **PDVD**. This automation is provided in both halves of the program.

The first half computes the response to input x(n). If the degree of the numerator N × NX exceeds that of the denominator D × DX, you will crash with a divide-by-zero error because you have violated causality. To fix this, choose integer m > 0 and replace 'z^(n-1)' in ICNVS with 'z^(n-1+m)'. Also replace 'n>0' by 'n>m' and increase the **CSERS** inputs by m.

The second half computes the response to the initial conditions y. If you choose m > 0, then the second half must be similarly adjusted. The fancy combination of the **LREV**, **VSUBS**, and **PMPY** commands performs the double sum in the numerator of the above equation. The "Y 1E-90 VADD" guarantees this trick does not fail due to zero initial conditions, while having no numerical effect on the result.

These techniques can also be applied to the computation of inverse Laplace transforms. For theory, see either Stearns and David or Proakis and Manolakis.

Examples are also given in the FILTR menu discussed in Chapter 27, where the related commands **PLDVD** and **IXFRM** are discussed.

DATA PROCESSING MENU {PROC}

FUNCTION	COMMAND	INPUTS	OUTPUTS
UNIT IMPULSE	**UNITI**(N)	N = SIZE	[VECTOR]
UNITI creates a vector of size N whose first value is one and the rest are zero. UNITI(5) = [1 0 0 0 0]			
RANDOM VECTOR WITH NORMAL AMPLITUDE STATISTICS	**RNDC**(μ,σ,ρ,s,N) s = SEED N = VECTOR SIZE	μ = COMPLEX MEAN σ = COMPLEX σ ρ = CORRELATION COEFFICIENT	RANDOM [VECTOR]
RNDC creates two normally distributed vectors and stores them as a complex vector. The real part of the mean and standard deviation is associated with the real part of the resulting vector. The correlation coefficient ρ is real and $\rho \in [-1, 1]$. The command **C→R** will separate the vectors. RNDC((1,2) , (3,4) , .5 , 1 , 8) creates a complex, normally distributed random vector of size 8. The real part has mean 1 and standard deviation 3, while the imaginary part has mean 2 and standard deviation 4. The correlation between the real and imaginary parts is 0.5.			
DEGLITCH	**DGLIT**(V,F,L)	V = [VECTOR] $\in \mathbb{C}$	[VECTOR]
This command replaces a small amount of bad data in V with linearly interpolated values over the range of [F + 1, L − 1] where $F + 2 \leq L$. For example: [1 2 10 4 5] 2 4 **DGLIT** yields [1 2 3 4 5].			
REAL PLOT	**PLT1**(V)	[VECTOR]	ONE PLOT
UP DIRECTORY	**UPDIR**	NONE	PARENT MENU

DATA PROCESSING MENU {PROC}

FUNCTION	COMMAND	INPUTS	OUTPUTS

DIGITAL SPECTRAL ANALYSIS APPLICATIONS

MATHLIB provides the tools to implement many of the digital spectral analysis techniques in the literature. Included are the classical techniques of autocorrelation, cross-correlation, correlograms, periodograms, autoregressive moving average (ARMA), autoregressive (AR), moving average (MA), Yule–Walker and modified Yule–Walker, maximum entropy method (MEM), linear prediction (LP), Prony method, and singular value and eigen subspace decomposition techniques. For details, see Marple's textbook. See also *IEEE Signal Processing Magazine*, April 1992 and *IEEE PGIT*, March 1992 for general time-frequency representations with wavelets, STFTs, quadratic TFRs, and multiresolution processing. The following program creates and plots a complex time series for N a power of 2.

```
« → N « "COMPUTE DATA " N + 3 DISP N INDEX DUP 'i×2×π×.25'
 →NUM × VEXP SWAP 'i×2×π×.35' →NUM × VEXP + 0 (.2,.3) 0 1 N
 RNDC + DUP PLTC DUP N TWIDL FFT DUP HALF1 VABS PLT1 » »
```

TOEPLITZ MATRIX INVERSE

TINV inverts a Hermitian Toeplitz matrix given its first column V. The inverse is not Toeplitz.

```
TINV: « → V « V SIZE EVAL DUP UNITI → N U « 0 N 1 - FOR K U
        K NEG VROT V WL2 DROP →COL NEXT 2 N START CCMB
        NEXT » » »
```

Define the Hermitian Toeplitz matrix by the vector V = [(3,0) (–2,.5) (.7,–1)]. Then its inverse is

$$\text{TINV(V)} = \begin{bmatrix} (.75637,0) & (.65287,-.02389) & (.26274,-.15924) \\ (.65287,.02389) & (1.19586,0) & (.65287,-.02389) \\ (.26274,.15924) & (.65287,.02389) & (.75637,0) \end{bmatrix}$$

DATA PROCESSING MENU {PROC}

FUNCTION	COMMAND	INPUTS	OUTPUTS

POWER METHOD OF EIGEN ANALYSIS

The below two programs compute the smallest and largest eigenvalues with associated eigenvectors, respectively. They use the power method with Rayleigh quotient and are for Hermitian Toeplitz matrices only. V is the first column of the matrix, T = HTOEP(V) is the matrix, and $\varepsilon > 0$ specifies the desired convergence accuracy. If only eigenvalues are required, the second DO UNTIL END loop may be omitted in the below programs. MAXλ is easily modified for arbitrary matrices.

```
MINλ: « → V ε « V (1,0) CON 10 → U λ « DO U U V WL2 DROP 'U'
STO U CONJ DUP ROT DOT SWAP U DOT DUP INV 'U' STO× / UNTIL
λ λ 3 PICK 'λ' STO ROT - SWAP / ABS ε < END V HTOEP → T «
DO U V WL2 DROP DUP DUP CONJ DOT / 'U' STO UNTIL T λ U ε 10
× λV? END » U λ » » »
```

```
MAXλ: « → T ε « T 1 ECOL (1,0) CON 10 → U λ « DO U U CONJ
DUP T U × DUP 'U' STO DOT ROT ROT DOT DUP INV 'U' STO× /
UNTIL λ λ 3 PICK 'λ' STO ROT - SWAP / ABS ε < END DO T U × U
DUP CONJ DOT / 'U' STO UNTIL T λ U ε 10 × λV? END U λ »»»
```

Define the Hermitian Toeplitz matrix by the vector V = [(3,0) (–2,.5) (.7,–1)].

MINλ(V,1E–10) = 2: [(.13791,–.01742) (.21272,0) (.13791,.01742)] , 1: 0.48869

MAXλ(T,1E–10) = 2: [(.48359,.18404) (–.59688,0) (.48359,–.18404)] , 1: 6.5491

Program DM uses MAXλ to solve for the eigenvalues and eigenvectors of matrix M = HTOEP(V). Then it deflates the space (removes the eigenvalue and eigenvector from M) and solves for the next one. By this means, all of the eigenvalues and eigenvectors are computed, and the final M is the zero matrix.

```
DM: « → M ε « M SIZE 1 GET { } { } → N Lλ LV « 1 N START M ε
MAXλ DUP2 'Lλ' STO+ 'LV' STO+ SWAP →COL DUP ABS / DUP TRN ×
× M SWAP - 'M' STO NEXT LV Lλ » » »
```

DATA PROCESSING MENU {PROC}

FUNCTION	COMMAND	INPUTS	OUTPUTS

LINEAR CONVOLUTION OF REAL WAVEFORMS WITH FILTERS (FFT)

The program FFTLC linearly convolves any real waveform defined by its digital values contained in the vector X with a frequency domain filter of length N, which must be a power of 2. The program FILTR given below is a dummy filter of all ones, which can be modified by the user for his or her application. FFTLC uses the overlap-and-save algorithm to produce a linearly (not circularly) convolved output. Since the input is real, the number of FFTs can be reduced by a factor of 2 by exploiting symmetries as explained in most texts on digital processing. This is provided by MUX, DMUX, and SPLIT below. TSTW below provides an example test waveform. The n input in TSTW must be ≥ 1.5N.

```
FFTLC: « → X N « IF X SIZE EVAL N 1.5 × < THEN 515 DOERR END
"INITIALIZE SE" 4 DISP N 1 - N 2 / N TWIDL DROP → N2 N1 T
« X 1 N VSUBS X N1 1 + DUP N2 + VSUBS R→C T N FFT DMUX N
FILTR MUX T N IFFT SPLIT N1 1 + N VSUBS VCMB X SIZE EVAL N
N1 + - N / IP → Y M
« IF M 0 > THEN "RUN SE" 4 DISP 1 1 M + 1 - FOR K X K N × 1
+ DUP N2 + VSUBS X K N × N1 + 1 + DUP N2 + VSUBS R→C T N
FFT DMUX N FILTR MUX T N IFFT SPLIT R→C N1 1 + N VSUBS C→R
VCMB Y SWAP VCMB 'Y' STO NEXT END Y » » » »

FILTR: « → X Y N « X Y » »

MUX: « → X Y « X RE Y IM - X IM Y RE + (0,1) × + » »

DMUX: « → V « V 2 / C→R DUP2 VREV -1 VROT SWAP VREV -1
VROT 4 PICK OVER + 4 PICK 4 PICK - R→C ROT 4 ROLL +
3 ROLL 4 ROLL - R→C » »

SPLIT: « → V « C→R TIFRE SWAP TIFRE SWAP » »

TSTW: « → n « 0 3 8 1 n 5 →LIST SQWV LINT » »
```

DATA PROCESSING MENU {PROC}

FUNCTION	COMMAND	INPUTS	OUTPUTS

LINEAR CONVOLUTION OF REAL WAVEFORMS WITH FILTERS (TMIC)

The FFT is a genus 1 transform defined by the block size N. An example of a genus 3 transform is given below. In the digital processing and multi-rate filtering literature, it is called transmultiplexing. H3ON computes a polyphase filter of length 3N. Program ITM combined with a forward FFT provides a forward transform into the frequency domain. Program TM combined with a forward FFT provides an inverse transform back to the time domain. Using these transforms, TMIC3 provides linear convolution of input waveform vector X with filter FILTR defined above. TMIC is an abbreviation for transmultiplexer interference canceler. Higher genus transforms provide better orthogonality in the frequency decomposition of waveforms. N = 8, 16, 32, ... Do not use N = 2 or 4.

```
TMIC3: « → X N « IF X SIZE EVAL N 5.5 × < THEN 515 DOERR END
  " INITIALIZE TMIC" 4 DISP N H3ON 3 N × DUP 1 - N 2 / N TWIDL
DROP -1 N EINDX → H N3 N2 N1 T S « 0 2 FOR K H X K N × 1 +
  DUP N2 + VSUBS N ITM H X K N × N1 + 1 + DUP N2 + VSUBS N
  ITM R→C T N FFT DMUX S VECTX NEXT 1 3 START 6 ROLL 6 ROLL
N FILTR MUX T N FFT SPLIT NEXT VCMB VCMB VCMB VCMB VCMB N
ZERO SWAP VCMB DUP H N N1 TM N1 1 + N VSUBS X SIZE EVAL 5
  N × N1 + - N / IP → V Y M « IF M 0 > THEN " RUN TMIC" 4 DISP
3 3 M + 1 - FOR K H X K N × 1 + DUP N2 + VSUBS N ITM H X K
  N × N1 + 1 + DUP N2 + VSUBS N ITM R→C T N FFT DMUX S
  VECTX N FILTR MUX T N FFT SPLIT VCMB V SWAP VCMB 'V' STO K
  2 × 4 - N × 1 + DUP N2 + N + N3 + V ROT ROT VSUBS H N N1
    TM Y SWAP VCMB 'Y' STO NEXT END Y » » » »

ITM: « → H X N « 0 N 1 - FOR n 'Σ(k=0,2,H(n+N×k+1)×X(3×N-n-N×k))'
                →NUM NEXT N →ARRY » »

TM: « → X H N N1 « 0 1 FOR j 0 N1 1 - FOR n
  'Σ(k=0,5,H(n+N1×k+1)×X(5×N+n+1-N×k+j×(N+N1)))' →NUM NEXT
  NEXT N →ARRY N1 × » »
```

DATA PROCESSING MENU {PROC}

FUNCTION	COMMAND	INPUTS	OUTPUTS

H3ON: ≪ → N ≪ .911437827766 .411437827766 → H1 H2 ≪ 0 3 N × 2 / FOR K '1+2×(H1×COS(2×π×K/3/N)+H2×COS(4×π×K/3/N))' →NUM NEXT 3 N × 2 / 1 + →ARRY 3 / N / REFLT 3 N × 2 / VROT ≫ ≫ ≫

DISCRETE FOURIER TRANSFORMS

The following two programs provide forward and inverse DFTs. They are slower than the FFT but do allow the input vector α to be any size η.

DFT: ≪ → α η ≪ { } η 1 − → τ η1 ≪ α OBJ→ DROP 1 η1 FOR K + NEXT 'τ' STO+ 1 η1 FOR β " FORWARD DFT " β + 3 DISP 'EXP(−i×2×π×β/η)' →NUM → W ≪ 1 1 η1 FOR K DUP × NEXT η →ARRY α DOT 'τ' SWAP STO+ ≫ NEXT τ OBJ→ →ARRY ≫ ≫ ≫

IDFT: ≪ → α η ≪ { } η 1 − → τ η1 ≪ α OBJ→ DROP 1 η1 FOR K + NEXT η / 'τ' STO+ 1 η1 FOR β " INVERSE DFT " β + 3 DISP 'EXP(i×2×π×β/η)' →NUM → W ≪ 1 1 η1 FOR K DUP × NEXT η →ARRY α DOT η / 'τ' SWAP STO+ ≫ NEXT τ OBJ→ →ARRY ≫ ≫ ≫

Abramowitz, M., and Stegun, I., *Handbook of Mathematical Functions*, AMS 55, 1964, Chapter 25.

Cadzow, J., *Foundations of Digital Signal Processing and Data Analysis*, New York, Macmillan, 1987.

Marple, *Digital Spectral Analysis*, Englewood Cliffs: NJ, Prentice-Hall, 1987.

Hlawatsch, F., and Boudreaux-Bartels, G., "Linear and Quadratic Time–Frequency Signal Representations," *IEEE Signal Processing Magazine*, New York, Vol. 9, No. 2, April, 1992.

IEEE Transactions on Information Theory, New York, Vol. 38, No. 2, March, 1992, special issue on wavelet transforms and multiresolution signal analysis.

The rest of the references are listed at the end of Chapter 27.

27

FILTER DESIGN AND ANALYSIS

INTRODUCTION

This chapter presents the 54 filter design and analysis operations in the FILTR menu. The FILTR commands provide Butterworth, Chebyshev, elliptic, and Bessel filter design software, with heavy emphasis on elliptic filters. Tools for analyzing the complex response and group delay are also provided. Additional tools are provided for extending the capabilities to many other filter designs.

Combined with the bilinear transform and windowing functions in the WIND menu, these designs can be extended to digital IIR and FIR filter design. **IXFRM** provides symbolic inverse Laplace and z transforms. State space filter design and analysis are also covered in this chapter.

LINEAR AND LOGARITHMIC FILTER RESPONSE

Given the zero and pole vectors, or the corresponding polynomial lists plus the response specification, **FTRV1** and **FTRV2** compute the complex response. The output is a complex vector with the response as a function of frequency, which may be Fourier transformed with the FFT command to obtain the impulse response and plotted by any of the plot programs. Similarly, **τVT1** and **τVT2** provide filter group delay as a function of frequency. **FTRVL** provides the complex response using a logarithmic frequency scale. Numerous analysis tools are given in the PROC menu.

BUTTERWORTH FILTERS

BPOLE computes the Butterworth pole vector required for computing the response and computing the component values for realization.

CHEBYSHEV FILTERS

CPOLE computes the Chebyshev pole vector required for computing the response and computing the component values for realization.

ELLIPTIC FILTERS

ESOLV combined with **EPOLE** and **EZERO** solves for the location of the zeros and poles based on the design specification by solving the necessary elliptic nonlinear equations. The zeros and poles are returned in vectors and can be used to compute the component values for realization. **ΩMIN** computes the frequencies of minimum stopband attenuation, and **ΩMAX** computes the frequencies of maximum passband insertion loss.

BESSEL FILTERS

A simple program is given as an example for computing the Bessel polynomial filter. The complex coefficient root solvers **PROOT** and **AROOT** are available to compute the corresponding pole vector. Bessel filters of any order can be designed with these tools. This example shows that the MATHLIB software is written in sufficient generality to support the analysis of almost any filtering problem. The **BESLF** command provides the Bessel filter coefficients.

HIGHPASS, BANDPASS, AND BANDSTOP DESIGN

The command **L→LP** is available for scaling lowpass designs. **L→HP**, **L→BP**, and **L→BS** provide scaling and conversion from lowpass designs to highpass, bandpass, and bandstop designs.

DIGITAL FILTER DESIGN

BILNT provides the bilinear transform used to transform analog filter designs into digital ones. It is also useful for transforming differential equations into difference equations. Frequency warping is provided by the command **WARP**. **DHBRT** designs digital Hilbert transformers for creating analytic signals and introducing 90° phase shifts.

STATE SPACE ANALYSIS AND SYNTHESIS

Conversion from state space to the zeros and poles of the transfer function is the difficult computational problem solved by command **S→ZP**. Given the transfer function, **TF→C** and **C↑O** can be used to compute the state space representations in both the controllable and observable canonical form realizations. **COMT** computes the controllability and observability matrices. From the zeros and poles, the Jordan canonical forms can also be computed.

COMPLEX PLANE POLE PLOTS

Given the pole or zero vector, **POLEP** provides a plot of the roots in the complex plane. **POLEP** can be used to plot the roots of both Laplace and z transform root vectors.

POLYNOMIAL COEFFICIENT AND ROOT VECTOR CONVENTIONS

The commands in the FILTR menu follow the same conventions as defined in the ALGB menu discussed in Chapter 19. You must understand these conventions to operate the software in this menu.

DIMENSIONALITY

The "linear" in linear filtering allows easy normalization of units. The squared magnitude function for an analog N-pole Butterworth filter is of the form:

$$|H(i\omega)|^2 = \frac{1}{1 + (i\omega/i\omega_c)^{2N}} \qquad H(s)H(-s) = \frac{1}{1 + (s/i\omega_c)^{2N}},$$

where ω is the usual Fourier transform variable and s is the usual Laplace transform variable. Then the poles which are the roots of the denominator are located at:

$$s_p = (-1)^{\frac{1}{2N}} (i\, 2\pi\, f_c) \qquad \omega_c = 2\pi\, f_c.$$

Defining the normalized variable $q = s_p/(2\pi\, f_c)$, we obtain an equation for the normalized roots. If you place 2: '(q/i)^(2×N)=-1', 1: 'q' on the stack and push **ISOL**, the HP 48 returns the symbolic solution 'q=EXP(2×π×i×n1/(2N))×(-1)^INV(2×N)×i'. Thus, the normalized roots are on the unit circle, and the s roots are on the $2\pi\, f_c$ circle. Hence, we can perform filter design in radians/sec, Hertz, or normalized ($f_c = 1$) Hertz and later transform the filter the desired case. All of the MATHLIB software is based on normalized designs with commands provided to later transform the filters to the actual frequencies.

In the case of using the bilinear transform to map the Laplace transform equations into z transform equations, this normalization concept has an additional subtlety. In the transformation we express the variable s in terms of the variable z, and in particular its reciprocal z^{-1}.

$$2\Omega\, \frac{1-z^{-1}}{1+z^{-1}} \quad \rightarrow \quad s\,,$$

where Ω is usually written as 1/T in most of the texts, but, in fact, in linear filtering theory, because of the normalization concepts just presented, Ω need have no relation at all to the digital sampling period T. In particular,

$$i2\Omega\, \frac{1 - z^{-1}}{1 + z^{-1}} \quad \text{for} \quad z = e^{i2\pi f_c T} \quad \text{equals} \quad i2\Omega\, \text{TAN}\left(\frac{2\pi f_c T}{2}\right) \quad \rightarrow \quad i\Omega_c\,.$$

Hence, the proper units must be preserved between T and the cutoff frequency f_c when prewarping the cutoff frequency, but afterward the units may be again normalized. While input parameter Ω in **BILNT** is included for generality in some mathematics applications, in filtering problems using MATHLIB, Ω should always equal 0.5 so that the resulting digital filters are properly normalized.

ROOT VECTOR CONVENTION

Computing polynomial lists from root vectors is very fast compared to using the complex root solvers to go the other direction. Consequently, the MATHLIB filter design software uses root vectors as the basic input, with commands available to always compute the equivalent lists. This approach also allows easy sectioning and component computation from the roots. The disadvantage to this approach is that the filter overall gain is not carried along with the filter, but is easily introduced when the design is complete, using the **GAIN1** command.

A second reason for using the root vector convention is numerical. As soon as a filter design is translated out even into kilohertz, the polynomial coefficients will take on a huge range in values, making it difficult if not impossible for any root solver using a reasonable number of digits precision to solve for the roots without first scaling the design back to unity. Consequently, designing filters with the roots is far more accurate than designing with the polynomial coefficients.

SYMBOLIC INVERSE TRANSFORMS

The **IXFRM** command provides symbolic computation of inverse Laplace and z transforms. This is useful for computing impulse responses and performing filter approximations. It can also be used to symbolically solve linear differential equations.

INSERTION LOSS

The insertion loss or passband peak-to-peak ripple in dB denoted by a_{max} may also be specified in terms of the reflection coefficient ρ or the ripple factor ε, which is not in dB. Defining equations and conversion commands are provided by $\rho\rightarrow$**AX**, $\rho\leftarrow$**AX**, **AX**$\rightarrow\varepsilon$, and **AX**$\leftarrow\varepsilon$ in the FILTR menu. These and additional conversion commands are available in the WIND menu discussed in Chapter 28.

BANDPASS AND BANDSTOP DESIGNS

MATHLIB follows the standard practice of using geometric bandpass and bandstop designs. The low-frequency input FL and the high-frequency input FH define the analog band edges. The analog bandwidth is then FH – FL, and the center frequency is then $\sqrt{}$(FH FL) on a logarithmic scale.

STATE SPACE CONVENTIONS

A time-invariant dynamical or state or phase space system may be represented in terms of the equations:

$$\frac{dx(t)}{dt} = \mathbf{A}\ x(t) + \mathbf{B}\ u(t) \qquad y(t) = \mathbf{C}\ x(t) + \mathbf{D}\ u(t),$$

where x(t) is the n × 1 state vector, y(t) is the 1 × m output vector, u(t) is the 1 × p input vector, and A, B, C, D are the state, input, and output matrices. While the commands in the FILTR menu can be extended using the MATR menu commands, all of the commands in this menu are limited to single-input, single-output systems. Thus, y(t) and u(t) are scalars, D is a 1 × 1 matrix, B is a column vector, C is a row vector, and A is an n × n matrix. See page 447 for discrete state space systems.

STATE SPACE TO ZEROS AND POLES

MATHLIB uses the symbolic matrix inverse command **SMI** to compute the zeros and poles of a state space system. This method was chosen both for speed and because it easily handles zeros at infinity. A detailed description of **SMI** is given in Chapter 25.

PRACTICAL ISSUES WITH JORDAN DECOMPOSITION

Within numerical constraints, program EVSOV given at the end of Chapter 20 will compute the Jordan decomposition of matrix M if all the eigenvalues are distinct. If you then compute Z^{-1} M Z, and the result is not diagonal within $\varepsilon > 0$, you probably have repeated eigenvalues, and EVSOV will not compute the Jordan decomposition.

Chapter 9 of Bronson gives a very detailed discussion of creating Jordan canonical decompositions by computing linearly independent generalized eigenvectors. MATHLIB can implement all of those techniques. However, the issues of accurate eigenvalue and rank determination make the techniques difficult in practice. Round-off error makes it very difficult to determine the existence and multiplicity of repeated eigenvalues. Furthermore, difficult rank decisions must be made at every stage, and the final computed block structure critically depends on those decisions.

The technique we demonstrate at the end of this chapter, while still suffering from the transfer function root determination accuracy problem, is possibly more intuitive than generalized eigenvectors. See the literature for more details.

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
FILTER VECTOR	**FTRV1**(Z,FS,K,δ)	Z FS K ∈ **N** δ ∈ [0, 1]	[VECTOR]

FTRV1 computes frequency samples of a transfer function's response where the transfer function is specified by the root (zero) vector Z. FS is the digital sample rate, and even $K \geq 4$ is the total length of the output response vector spanning the frequency range [0 , (K–1) FS / K]. Thus, the frequency scale is linear and K MUST BE EVEN. The positive frequency response values are given by the index values $k \in [0, K/2]$ and the negative frequency response values are given by the index values $k \in [K/2 + 1, K - 1]$ and are the conjugate reflection of the positive frequency values. This symmetry corresponds to a time domain impulse response which is real. Also, the complex value of the K/2 term, which should be small in realistic applications, is replaced by its absolute value to make sure that it does not introduce imaginary values into the impulse response when **IFFT** is used to compute the impulse response of the filter. A complex time domain filter is handled by computing the response associated with the real part and the imaginary part separately and adding the responses. In the case of an analytic filter, this corresponds to zeroing the negative frequency values of the response. The response of the empty vector [ ] symbolized by { } is a vector of all ones. Examples are given below after introducing the next command.

When $\delta = 0$, **FTRV1** treats the input as the roots of the Laplace transform variable s, which is evaluated as $i\omega$, and thus provides the normal Fourier frequency response. When $\delta = 1$, then the input is treated as a digital filter in the variable z^{-1}, which is evaluated as $e^{-i\omega} = e^{-i2\pi f}$. In the digital case FS is normally 1 or 0.5 since the period of $e^{-i2\pi f}$ is 1.

Z may be either a root vector or the corresponding polynomial list.

FILTER VECTOR	**FTRV2**(N,D,FS,K,δ)	N D FS K ∈ **N** δ ∈ [0, 1]	[VECTOR]

FTRV2 computes the response in the case where there are both zeros and poles.

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS

FTRV2 computes the response in the case where there are both zeros and poles specified by the N vector, such as the output of **EZERO**, and the D vector, such as the output of **EPOLE**. **FTRV2** calls **FTRV1** twice, inverts the denominator response with **VSINV**, and multiplies the two vectors using **VECTX**. **VSINV** discussed in Chapter 30 handles zeros in a special way. The program

« 6 BPOLE 4 40 0 FTRV2 20 VROT DUP SPECT VLOG 10 × PLT1
PHASE PLT1 »

computes the poles and zeros of a 6-pole Butterworth filter and plots the spectrum in dB and the phase in degrees over the frequency range [–2, 2] Hertz. The 3 dB frequency of this filter is 1 Hz, and since FS = 4 Hz, the 3 dB point on the plot is at the value of 10 = 40/4. At 2 Hz the filter response is down 35 dB from the DC response. **PHASU** is available in the PROC menu to unwrap the phase response. The inverse FFT command **IFFT** in the PROC menu will compute the impulse response of a filter, given the output of FTRV2. The "20 VROT" simply rotates the response so that zero frequency is at the center of the plot. A digital example is given on the next page.

N and D may be either root vectors or the corresponding polynomial lists.

Consider the digital filter represented by the numerator root vector { } representing the empty root vector [ ] and the denominator root vector [–1/0.6]. Using the below command **R→CL** to compute the coefficient lists, we have

$$R\rightarrow CL(\ \{\ \}\ ,\ [\ -1.66666666667\ ]\ ) = \begin{cases} 2:\ \{\ 1\ \} \\ 1:\ \{\ 1.66666666667\ \ \ 1\ \} \end{cases},$$

so the root vectors represent the digital filter

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS

$$H(z) = \frac{1}{1.66666666667 + z^{-1}} = \frac{0.6}{1 + 0.6\ z^{-1}}.$$

The following program plots the amplitude response and phase of this digital filter.

« { } [-1.66666666667] 1 40 1 FTRV2 DUP VABS PLT1 PHASE PLT1 »

This is an example of an infinite impulse response (IIR) filter. Its impulse response is easily evaluated and plotted using the commands **ICONV** and **UNITI** found in the PROC menu. Store the following program in a variable named IDEM1. It is used in several examples in this chapter.

IDEM1: « { } [-1.66666666667] R→CL 0 GAIN1 [0] "COMP IMPULSE RESPONSE" 3 DISP 9 UNITI ICONV DUP PLT1 »

Observe our example filter is a highpass filter since the polynomial roots are in z^{-1}, not z. Using the below command **RZINV**, the corresponding root vectors in z are [0] and [-0.6]. Using **R→CL**, these roots in z correspond to the polynomial coefficient lists in z { 0 1 } and { 0.6 1 }. **GAIN1** in this example is used unconventionally to normalize the impulse response at zero to one.

FILTER RESPONSE NORMALIZATION

With the output vector from **FTRV2** at Level 1 of the stack, the program steps DUP n GET ABS / will normalize the output of **FTRV2**. For the lowpass and bandstop cases, the DC gain is usually normalized to 1, so n = 1. For the highpass case, the highest frequency (represented by the N/2 + 1 value where N is the size of the vector) is normalized with n = N/2 + 1. For the bandpass case, the value of the filter at the center frequency is normally set to 1, so n is that index value corresponding to the center frequency. A lowpass Butterworth filter example is given below.

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS

INTERPOLATION OF THE IMPULSE RESPONSE

The plot of the impulse response for the above example filter is not pretty because there are only nine values plotted. Interpolation provides a means of increasing the sampling rate to obtain a better approximation to the underlying analog impulse response. The below two programs interpolate the output of IDEM1 by a factor of eight and compare the resulting impulse responses. The second program, while more difficult conceptually, is both faster and more accurate since it avoids the use of an imperfect 10-pole Butterworth filter.

« IDEM1 → V « "ZERO FILL FOR INTERP" 3 DISP V ZFIL1 8 VINTP 128 TWIDL FFT DUP SPECT PLT1 "BUILD LP FILTER" 3 DISP 10 BPOLE 16 INV L→LP 1 128 0 FTRV2 DUP 1 GET ABS / DUP SPECT PLT3 "APPLY LP FILTER" 3 DISP VABS VECTX 7 × 128 TWIDL IFFT HALF RE DUP PLT1 "OVERLAY PLOTS" 3 DISP V PLT3 V » »

« IDEM1 → V « V ZFIL1 16 TWIDL FFT DUP SPECT PLT1 "ZERO FILL FFT" 3 DISP 128 ZFILN DUP SPECT PLT1 8 × 128 TWIDL IFFT HALF RE DUP PLT1 "OVERLAY PLOTS" 3 DISP V PLT3 1 8 VDEC V 1 8 VSUBS - » »

Application of the decimation command **VDEC** to the interpolated output with first value n = 1 and decimation factor F = 8 reproduces to 10 digits the IDEM1 output with the second program.

DELAY VECTOR	τ**VT1**(Z,F,K,δ)	Z F K∈**N** δ∈[0, 1]	[VECTOR]

τ**VT1** computes the frequency samples of a transfer function's group delay where the transfer function is specified by the root (zero) vector Z.

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS

$K \geq 3$ is the size of the output vector containing the group delay evaluated at the frequencies k F/(K – 1) for $k \in [0, K - 1]$. Thus the frequency scale is linear, and the frequencies are non-negative. The group delay of the empty root vector [ ] symbolized by { } is a vector of all zeros. An example is given below.

When $\delta = 0$, **τVT1** treats the input as the roots of the Laplace transform variable s, which is evaluated as $i\omega$, and thus provides the normal Fourier frequency response. When $\delta = 1$, then the input is treated as a digital filter in the variable z^{-1}, which is evaluated as $e^{-i\omega} = e^{-i2\pi f}$. In the digital case, FS is normally 1 or 0.5 since the period of $e^{-i2\pi f}$ is 1.

DELAY VECTOR	**τVT2**(N,D,F,K,δ)	N D F K ∈ **N** δ	[VECTOR]

τVT2 computes the group delay for a filter specified by the numerator root (zero) vector N and the denominator root (pole) vector D. **τVT2** calls **τVT1** twice and differences the results. The below program computes and plots the group delay of a 6-pole Butterworth filter.

« 6 BPOLE 2 40 0 τVT2 PLT1 »

Observe that the delay at zero is 3.8637 s and the peak delay at a little less than 1 is 6.4897 s.

Similarly, the group delay for our above digital filter example is plotted by the program:

« { } [–1.66666666667] 1 40 1 τVT2 PLT1 ».

This group delay is easily verified by differentiating the phase with the **DER1** command.

Neither of the N, D, or Z inputs to **τVT1** or **τVT2** may be polynomial lists.

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
EVALUATE H(F)	**HOFω**(H,F,δ)	F = FREQUENCY δ ∈[0, 1]	VALUE

Given the complex vector H containing the roots (zeros) of some transfer function (or the corresponding polynomial list) and the frequency F, **HOFω** computes the complex value of H(F). This is the MATHLIB internal command called by **FTRV1** to evaluate filter responses.

EVALUATE H(F)	**τOFω**(H,F,δ)	F = FREQUENCY δ ∈[0, 1]	VALUE

Given the complex vector H containing the roots (zeros) of some transfer function and the frequency F, **τOFω** computes the value of the group delay for that transfer function. This is the MATHLIB internal command used by **τVT1** to evaluate group delay.

HERTZ VERSUS RADIANS PER SECOND

Rigorously speaking, the input frequencies to the above six commands would be in radians per second $\omega = 2\pi f$. In practice in the analog case, if the filter is scaled for Hertz instead of radians, then the above commands can accept the input frequency in Hertz. The below commands **L→LP**, **L→HP**, **L→BP**, and **L→BS** will do this scaling for you. However, the digital filters all involve nonlinear transformations ($e^{i\,2\pi f}$) and thus do not scale. The above six commands and **WARP** below have all been written for normalized Hertz inputs in the digital case where $\delta = 1$. They internally multiply by the 2π factor. By normalized Hertz we mean F = fT in Hertz-seconds where f is the actual frequency in Hertz and T is the sampling period in seconds. $F \in [0, 0.5]$. We recommend that you always work in Hertz.

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
LOG SCALED FREQUENCY RESPONSE	**FTRVL**(N,D,L,H,K)	N D K > 1 0 < L < H	[VECTOR] OF SIZE K

Given the filter numerator and denominator root vectors (polynomial lists) N and D, lower and upper frequency limits L and H, and the number of values K, **FTRVL** constructs a filter response vector like **FTRV2**, except that the frequency scale is logarithmic and the negative frequency response is not computed. Thus, LOG–LOG plots can be viewed. For example, the below program shows the geometric symmetry of lowpass to bandpass transformations using command **L→BP**:

« 6 BPOLE 2 8 L→BP .4 40 40 FTRVL DUP SPECT VLOG 10 × PLT1 VTRUD PLT3 »

POLE PLOT	**POLEP**(V)	POLE OR ZERO VECTOR	SCATTER PLOT

POLEP provides the capability of plotting the zeros and poles of both analog and digital filters represented by their corresponding root vectors.

WARNING: THIS IS THE ONE COMMAND IN THE MATH LIBRARY WHICH USES ΣDAT. If the current directory has a ΣDAT with data you do not wish to lose, then store it in a different variable.

« 6 BPOLE POLEP » plots the left half plane poles of the Butterworth filter.

« 6 BPOLE DUP NEG VCMB POLEP » plots all the poles of $|H(f)|^2$.

z transform responses can also be plotted with **POLEP**. If H(z) is a polynomial in z and is realizable (roots of H(s) lie in the left half plane), then all its roots are in or on the unit circle. However, if H(z) is a realizable polynomial in z^{-1}, then the roots are on or outside the unit circle.

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
REQUIRED FILTER ORDER	**FORDR**(F_c,F_s,a_{max}, a_{min},D?,T)	D?∈[0,1] T = 1 OR 2 OR 3	REQUIRED FILTER ORDER

FORDR can be used to estimate the required filter order. Parameter a_{max} is the maximum passband insertion loss in dB from zero to F_c Hertz, and a_{min} is the minimum attenuation in dB for frequencies greater than or equal to F_s. D? equals 1 if you are doing a digital design and 0 if you are doing an analog design. The three filter types supported are

T = 1: BUTTERWORTH T = 2: CHEBYSHEV T = 3: ELLIPTIC.

FORDR(1 , 1.4 , 0.5 , 50 , 0 , 1) = 21
FORDR(1 , 1.4 , 0.5 , 50 , 0 , 2) = 9
FORDR(1 , 1.4 , 0.5 , 50 , 0 , 3) = 6
FORDR(0.1 , 0.15 , 0.5 , 50 , 1 , 1) = 16
FORDR(0.1 , 0.15 , 0.5 , 50 , 1 , 2) = 8
FORDR(0.1 , 0.15 , 0.5 , 50 , 1 , 3) = 5

FORDR uses the equations given on page 241 of Rabiner and Gold.

ROOT VECTOR VARIABLE INVERT	**RZINV**(N,D)	N = NUMERATOR D = DENOMINATOR	2: NUMERATOR 1: DENOMINATOR

Given the ratio of two polynomials in some variable α, **RZINV** transforms the polynomials defined by the numerator root vector N and the denominator root vector D into root vectors of α^{-1}.

$$\text{RZINV}(\{\},[1\ \ 2\ \ 4]) = \begin{cases} 2\text{: } [0\ \ 0\ \ 0] \\ 1\text{: } [1\ \ .5\ \ .25] \end{cases}$$

Observe that the three zeros at infinity associated with the empty root vector { } become zeros at 0.

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
ROOT VECTORS TO COEFFICIENT LISTS	**R→CL**(N,D)	N = NUMERATOR D = DENOMINATOR	2: NUMERATOR 1: DENOMINATOR

Given the ratio of two polynomials in some variable α, **R→CL** transforms the polynomials defined by the numerator root vector N and the denominator root vector D into the ratio of polynomial lists in the variable α by applying **CLIST** to both N and D.

RZINV({ } , [1 2 4]) = { 2: { 1 }
1: { -8 14 -7 1 }

POLYNOMIAL LIST VARIABLE INVERT	**PZINV**(N,D)	N = NUMERATOR D = DENOMINATOR	2: NUMERATOR 1: DENOMINATOR

Given the ratio of two polynomials in some variable α, **PZINV** transforms the polynomials defined by the numerator polynomial list N and the denominator polynomial list D into polynomial lists of α^{-1}.

RZINV({ 1 } , { -8 14 -7 1 }) = { 2: { 0 0 0 1 }
1: { 1 -7 14 -8 }

BUTTERWORTH FILTER	**BPOLE**(N)	N = # POLES	2: { } 1: [VECTOR]

Returns complex vectors containing the zeros and poles of an Nth order Butterworth filter. The following program computes the zeros and poles of an order 6 Butterworth filter, computes its complex response, and plots the complex response. Push ATTN, and the magnitude of the response in dB will be plotted.

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
Then push ATTN to obtain the phase of that response in degrees. « 6 BPOLE 4 64 0 FTRV2 DUP PLTC DUP VABS VLOG 20 × PLT1 PHASE PLT1 »			
CHEBYSHEV FILTER	**CPOLE**(a_{max},N)	N = # POLES	2: { } 1: [VECTOR]
Returns complex vectors containing the zeros and poles of an Nth order Chebyshev filter. The parameter a_{max} is the maximum value of passband insertion loss in dB. **AX**→ε converts a_{max} to ripple factor, and **AX**←ε converts ripple factor into a_{max} in dB. For example: « 1 6 CPOLE 4 64 0 FTRV2 DUP PLTC DUP VABS VLOG 20 × PLT1 PHASE PLT1 »			
MODULAR ANGLE	**θFCFS**(F_c,F_s)	F_c F_s	θ
Given the highest frequency of the passband F_c and the lowest frequency of the stopband F_s, this command computes the corresponding modular angle θ in degrees. $\theta = 180/\pi \arcsin(F_c/F_s)$.			
E SOLVE	**ESOLV**(a_{max},a_{min},θ, n,F)	a_{max} a_{min} θ n F	M1 M2 n
This rather complicated program converts design specifications into parameters which determine the location of the poles and zeros of the Zolotarev rational function better known as an elliptic filter.			

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS

This class of filter contains *ripple* in both the passband and the stopband. While in theory the desired filter is determined from any three of the parameters (a_{max}, a_{min}, F_c/F_s, and n), it is more practical to specify all four and experiment with the design. A theoretically feasible design is not necessarily a good one. In terms of a lowpass filter, a_{max} is the maximum value of insertion loss in the passband response and should not exceed 1 dB, a_{min} is the minimum value of the stopband attenuation, F_c is the highest frequency of the passband, F_s is the lowest frequency of the stopband, and n is the order of the filter. Flag F should normally be set to zero.

The output is magic parameters M1, M2, and n. In his handbook, Zverev calls M1 and M2: Ω_c and Ω_s, which he also uses for F_c and F_s when there is no relationship. What you need to know about M1 and M2 is that M1 < 1 and M2 > 1 by some margin for a good filter design. For example, let a_{max} = .43 dB and a_{min} = 52 dB. Use θ**FCFS** to compute the modular angle θ and assume it equals 40°. Let n = 5. Then execute **ESOLV**(.43,52,40,5,0). Once the first nonlinear elliptic integral equation is solved, the software checks to see if there is a feasible solution. If none exists, a "Bad Guess(es)" error message is given. If a solution does exist, then the program proceeds to solve the second nonlinear elliptic equation. The outputs are M1=.951193300626, M2=1.4797408173, n=5.

THE NUMERICAL STABILITY OF THE NONLINEAR SOLUTION CAN CRUMBLE! Always plot your filter to be sure it is what you desire. The best plot is upside-down. For example, take the output of **FTRV2** and evaluate:

« VABS VLOG −20 × PLT1 »

If F ≠ 0, then the stack is pushed and a list is added to the bottom containing seven numbers: { Δa_e M1×M2 x_1 $e^{-\Delta a}$ K_1 k_1' K_1'/K_1 } in Zverev's Chapter 4 notation. In this example case:

{ 10.2892576613 1.40757021685 .782150272736 2.6393683224E−3
8.50810580912 8.07304021053E−4 .184623536422 }.

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
The following program provides a nice demo. « .43 52 40 5 0 ESOLV HALT → M1 M2 n « M1 M2 n EZERO M1 M2 n EPOLE HALT 6.4 64 0 FTRV2 VABS VLOG −20 × PLT1 M1 M2 n ΩMIN M1 M2 n ΩMAX » »			
ELLIPTIC	**EZERO**(M1,M2,N)	M1 M2 N ≥ 3	[VECTOR]
Returns a complex vector containing the zeros of an Nth order elliptic filter. For the above example, the zeros are (0, ±1.53805762513) and (0, ±2.31869860758).			
ELLIPTIC	**EPOLE**(M1,M2,N)	M1 M2 N ≥ 2	[VECTOR]
Returns a complex vector containing the poles of an Nth order elliptic filter. For the above example, the poles are (−8.35064686904E−2, ±.967447758756), (−.284661101405, ±.667520621612), and (−.41865463206, 0).			
ELLIPTIC	**ΩMIN**(M1,M2,N)	M1 M2 N ≥ 2	[VECTOR]
Returns a vector of frequencies where the stopband has its minimum attenuation. For the above example, the frequencies are 1.47979408173, 1.75295423683, and 4.26537368353.			
ELLIPTIC	**ΩMAX**(M1,M2,N)	M1 M2 N ≥ 2	[VECTOR]
Returns a vector of frequencies where the passband has its maximum insertion loss. For the above example, the frequencies are .329999273518, .802970315637, and .951193300626.			

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
$a_{max}(\rho)$	$\rho \rightarrow$ **AX**(ρ)	REFLECTION COEF ρ	a_{max} in dB
Computes a_{max} in dB given the reflection coefficient in percent. $a_{max} = -10 \times LOG(1 - SQ(\rho/100))$ $\rho = 100\sqrt{}(1 - ALOG(-a_{max}/10))$ The voltage-standing-wave ratio VSWR = $(1 + \rho/100)/(1 - \rho/100)$.			
$\rho(a_{max})$	$\rho \leftarrow$ **AX**(a_{max})	a_{max} in dB	ρ in %
RIPPLE FACTOR ε	**AX**$\rightarrow \varepsilon(a_{max})$	a_{max} in dB	ε
$\varepsilon = \sqrt{10^{0.1a_{max}} - 1}$ $a_{max} = 10LOG(1 + \varepsilon^2)$			
a_{max}	**AX**$\leftarrow \varepsilon(\varepsilon)$	ε	a_{max} in dB
POLY RESCALE	**PSCAL**(L)	POLYNOMIAL LIST	POLYNOMIAL LIST
Provided the first element in polynomial list L is not zero, **PSCAL** rescales the list such that the first element in the list is 1. For example: PSCAL({ 2 4 6 }) = { 1 2 3 }.			

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS

ANALYSIS OF OTHER FILTER DESIGNS INCLUDING BESSEL FILTERS

The filter tools in this menu can be used for other filter designs also.

Given the filter transfer function, either **PROOT** or **AROOT** below can be used to create the zero and pole vectors used by **FTRV1**, **FTRV2**, **τVT1**, and **τVT2**.

The following program will create the Laplace transform of a Bessel, maximally flat time delay, filter:

« → L « 'Σ(n=0,L,(2×L−n)!/2^(L−n)/n!/(L−n)!×s^n)' EVAL » »

Calling the output of the above program B(s), then the Bessel filter response is B(0)/B(s). Evaluate the above program with L = 7 and then using **COEFL**(B,s,7) to compute the coefficient list results in the output list

{ 135135 135135 62370 17325 3150 378 28 1 }.

Now solve for the roots of this polynomial using **PROOT** to obtain the vector:

[(−2.6856768790, ±5.4206941304) (−4.070139164, ±3.5171740494)
(−4.758290526, ±1.739286058) (−4.971786862, 0)]

with ε = 1E−10. This root vector may be normalized to Hertz by dividing by 2π. The numerator root vector is { }. Use **POLEP** to plot the poles, **FTRV1** to compute the response, and **τVT1** to compute the group delay.

Other filter designs may be analyzed similarly.

COEFFICIENTS	**COEFL**(F,V,N)	F V N	COEFFICIENTS

Reduces function F in variable V to a coefficient list of size N + 1, which is its Maclaurin series expansion. See the MISC menu for examples.

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
POLYNOMIAL ROOT	**PROOT**(L,ε,MAX)	L ε MAX	[VECTOR]

PROOT computes the roots of polynomials with complex coefficients. **PROOT** takes the complex coefficient list L (see **FEVAL** for definition and order) and creates a Frobenius matrix M. **EIGEN** is used to compute the eigenvalues of M, which are the complex roots of L. For example, let $f(x) = x^5 - 0.2x^4 + 7x^3 + x^2 - 3.5x + 2 = 0$. Then list L = { 2 –3.5 1 7 –0.2 1 }. With ε = 1E–10 the roots are (.140377913, ±2.7314490896), (–.908563257, 0), (.413903715, ±.3506447075). MAX is the maximum number of iterations (say 20) **EIGEN** is allowed to converge each eigenvalue. If the solution fails at some eigenvalue, the diagonal matrix is still returned, but only the last values of the vector which converged are actual roots.

POLYNOMIAL ROOT	**AROOT**(L,v,ε, MAX)	L v ε MAX	[VECTOR]

AROOT uses Laguerre's method for solving for all the roots of the complex polynomial defined by the list L. Value v is the initial guess for all of the roots, and ε sets the convergence criterion. MAX (say 100) is the maximum iterations allowed to converge each root. After converging each root, **DEFLT** is used to remove that root from polynomial L. The most difficult polynomial root solutions occur when there are repeated roots. Consider the example: $'x^{12} + 99x^{11} - 377x^{10} - 26395x^9 + 149080x^8 + 1703048x^7 - 15440048x^6 + 8684864x^5 + 302914240x^4 - 1377763200x^3 + 2718976000x^2 - 2620320000x + 1008000000' = (x-2)^5 (x-5) (x+6) (x-7) (x+10) (x-10) (x+15) (x+100)$. With v = 0, ε = 1E–8, and MAX = 100, we only get two roots: (1.98607080, 0) and (1.99562237, –1.32705219E–2). Dropping ε = 1E–4 yields the roots: (1.98607080, 0) (1.99563474, –1.32669871E–2) (1.99563459, .0132665015) (2.01132960, 8.27241365E–3) (2.01133027, –8.27192808E–3) (5, 0) (–6, 0) (7, 0) (–10, 0) (10, 0) (–15, 0) (–100, 0). The average of the first 5 roots is (2, 0) to 11 digits. **LROOT** can be used to verify that (2, 0) is the true root and **DEFLT** can be used to remove it from P giving the reduced polynomial { (–31500000, 0) (3135000, 0) (1619500, 0) (–108650, 0) (–23945, 0) (673, 0) (109, 0) 1 }, which is easily solved since all the roots are unique.

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
$\delta\,\eta$ PARAMETER CONVERSION	$\delta\eta \rightarrow$ **A**(δ,η)	δ AND η IN VOLTS	2: a_{max} dB 1: a_{min} dB

If the passband peak-to-peak response is specified as varying from $1 + \delta$ to $1 - \delta$, and the maximum stopband response is η, then $a_{min} = 20\ \text{LOG}([1+\delta]/\eta)$ dB and $a_{max} = 20\ \text{LOG}([1+\delta]/[1-\delta])$ dB. Rabiner and Gold use this notation where $\delta = \delta_1$, $\eta = \delta_2$.

BESSEL FILTER	**BESLF**(N)	N = ORDER ≥ 0	COEFFICIENT LIST

BESLF computes the denominator coefficient list for an Nth order Bessel filter.

PREWARP FREQ	**WARP**(F)	FREQUENCY	FREQUENCY

$$\text{TAN}\left(\frac{2\pi F}{2}\right) \qquad F \in \left[0, \tfrac{1}{2}\right]$$

WARP provides frequency prewarping that is used with the bilinear transform. The input frequency is normalized, and F = f T where f in Hertz is the actual frequency and T in seconds is the sampling period. When designing digital filters, the frequencies input to the commands **L→LP**, **L→HP**, **L→BP**, and **L→BS** should be prewarped. This will properly set the filter shape after using **BLINT** to perform the bilinear transform. See the below examples.

LOWPASS TO LOWPASS	**L→LP**(N,D,F) FREQUENCY F	N = NUMERATOR D = DENOMINATOR	2: NUMERATOR 1: DENOMINATOR

L→LP is a frequency scaling command which does not change the filter shape.

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS

Design an analog 6-pole Chebyshev filter with a_{max} = 1 dB and cutoff frequency f_c = 10 KHz. Plot it over the range –32 KHz to 32 KHz using a linear scale with amplitude in Watts and phase in degrees: « 1 6 CPOLE 10000 L→LP 64000 64 0 FTRV2 32 VROT DUP SPECT PLT1 PHASE PLT1 »

Design a digital IIR lowpass filter using the bilinear transform based on a 6-pole Butterworth design with cutoff frequency 2 KHz and sampling rate 10 KHz. Plot the poles and zeros as well as the response from –5 KHz to 5 KHz: « 6 BPOLE 2000 .0001 × WARP L→LP .5 BILNT DUP2 RZINV POLEP POLEP 1 64 1 FTRV2 32 VROT DUP SPECT PLT1 PHASE PLT1 »

LOWPASS TO HIGHPASS	**L→HP**(N,D,F) FREQUENCY F	N = NUMERATOR D = DENOMINATOR	2: NUMERATOR 1: DENOMINATOR

L→HP is a general frequency scaling and lowpass-to-highpass transformation command. For a fixed frequency, **L→HP** is its own inverse.

Design an analog highpass 6-pole Chebyshev filter with 1 dB ripple and cutoff at 20 KHz. Plot the poles and the response over the range 0 to 32 KHz:
« 1 6 CPOLE DUP POLEP 20000 L→HP 64000 64 0 FTRV2 HALF1 DUP SPECT PLT1 PHASE PLT1 »

Design a digital IIR highpass filter with cutoff 30 KHz and sampling rate 100 KHz based on a 6-pole 1 dB ripple Chebyshev analog design and plot the complex response: « 1 6 CPOLE 30000 .00001 × WARP L→HP .5 BILNT DUP2 RZINV POLEP POLEP 1 64 1 FTRV2 32 VROT PLTC » Observe that the real part is even and the imaginary part is odd.

NOTE THAT IN SEVERAL OF THE EXAMPLES IN THIS CHAPTER WE USE THE COMMAND **VROT** TO CENTER THE ORIGIN IN THE MIDDLE OF THE PLOT.

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
LOWPASS TO BANDPASS	**L→BP**(N,D,FL,FH) FREQUENCIES FL FH	N = NUMERATOR D = DENOMINATOR	2: NUMERATOR 1: DENOMINATOR

L→BP is a general frequency scaling and lowpass-to-bandpass transformation command.

Design an analog 5 pole Butterworth bandpass filter with cutoff frequencies 10 KHz and 20 KHz, plot the zeros and poles, and plot the response over 1 KHz to 200 KHz on a logarithmic scale:

« 5 BPOLE 10000 20000 L→BP DUP2 POLEP POLEP 1000 200000 64 FTRVL DUP 32 GET ABS / DUP VABS VLOG 20 × PLT1 PHASE PLT1 »

Design a digital IIR bandpass filter based on a 5-pole Butterworth analog design. The sampling rate is 100 KHz and the cutoff frequencies are 20 KHz and 30 KHz.

« 5 BPOLE 20000 .00001 × WARP 30000 .00001 × WARP L→BP .5 BILNT DUP2 RZINV POLEP POLEP 1 64 1 FTRV2 DUP 16 GET ABS / HALF1 DUP SPECT PLT1 PHASE PLT1 »

LOWPASS TO BANDSTOP	**L→BS**(N,D,FL,FH) FREQUENCIES FL FH	N = NUMERATOR D = DENOMINATOR	2: NUMERATOR 1: DENOMINATOR

L→BS is a general frequency scaling and lowpass-to-bandstop transformation command.

Design an analog 5 pole Butterworth bandstop filter with cutoff frequencies 5 KHz and 40 KHz, plot the zeros and poles, and plot the response over 1 KHz to 200 KHz on a logarithmic scale.

« 5 BPOLE 5000 40000 L→BS DUP2 POLEP POLEP 1000 200000 64 FTRVL DUP 1 GET ABS / DUP VABS VLOG 20 × PLT1 PHASE PLT1 »

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
Design a digital IIR bandstop filter based on a 5-pole Butterworth analog design. The sampling rate is 100 KHz and the cutoff frequencies are 20 KHz and 30 KHz. « 5 BPOLE 20000 .00001 × WARP 30000 .00001 × WARP L→BS .5 BILNT DUP2 RZINV POLEP POLEP 1 64 1 FTRV2 DUP 1 GET ABS / HALF1 DUP SPECT PLT1 PHASE PLT1 »			
BILINEAR TRANSFORM	**BLINT**(N,D,Ω)	N = NUMERATOR D = DENOMINATOR	2: NUMERATOR 1: DENOMINATOR

The bilinear transform is defined by the equations

$$s = 2\Omega \frac{z - 1}{z + 1} = 2\Omega \frac{1 - z^{-1}}{1 + z^{-1}} \qquad z = \frac{2\Omega + s}{2\Omega - s},$$

where s is the Laplace transform variable and z is the z transform variable.

BLINT transforms a filter defined by the numerator root vector N and the denominator root vector D. The output numerator and denominator are the corresponding root vectors corresponding to the inverse of the z transform variable z which is z^{-1}. If the root vector of the z transform variable z is desired, **RZINV** will compute it. This Bilinear transformation is very useful for transforming analog filter designs into digital filter designs. In these cases, Ω = 0.5 and **BILNT** is its own inverse.

BILNT([2] , [1 2 1] , 0.5) =

2: [-1 -1 -1/3]
1: [0 -1/3 0]

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS

DIGITAL IIR DESIGN COEFFICIENT EXAMPLES

The following examples provide specific infinite impulse response (IIR) digital coefficient examples of the above six commands using a 2-pole Butterworth filter for the analog design. Since the design is based on root vector calculations, the gain is arbitrary. **GAIN1** is used in the below programs in order to set the filter gain at the desired frequency to 1.

In the below examples the sampling period T = 0.0001.

f_c = 1 KHz and 3.5 KHz for the lowpass and highpass examples.

FL = 1.5 KHz and FH = 3.5 KHz in the bandpass and bandstop examples.

LOWPASS: « 2 BPOLE 1000 .0001 × WARP L→LP .5 BILNT R→CL
PSCAL 1 GAIN1 »
N = { 6.74552738888E-2 .134910547778 6.74552738888E-2 }
D = { 1 -1.14298050254 .412801598095 }

HIGHPASS: « 2 BPOLE 3500 .0001 × WARP L→HP .5 BILNT R→CL
PSCAL -1 GAIN1 »
N = { .131106439916 -.262212879832 .131106439916 }
D = { 1 .747789178264 .272214937926 }

BANDPASS: « 2 BPOLE 1500 .0001 × WARP 3500 .0001 × WARP L→BP
.5 BILNT R→CL PSCAL 10 SRND (0,-1) GAIN1 »
N = { .20657208385 0 -.4131441677 0 .20657208385 }
D = { 1 0 .3695273773 0 .1958157127 }

BANDSTOP: « 2 BPOLE 1500 .0001 × WARP 3500 .0001 × WARP L→BS
.5 BILNT R→CL PSCAL 10 SRND 1 GAIN1 »
N = { .3913357725 0 .782671545 0 .3913357725 }
D = {1 0 .3695273773 0 .1958157127 }

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
DIGITAL HILBERT TRANSFORMER	**DHBRT**(N)	NUMBER OF TAPS IS 2 × IP([N + 1]/2)	COEFFICIENT [VECTOR]

DHBRT computes the tap weights for a digital Hilbert transformer, which produces a 90° phase shift when a signal is convolved with it. The output coefficients should be windowed. A nice demo is created by the below example you can type in and store in variable HBRTD. The sampling period equals 12/40 = 0.3, so that the delay through the Hilbert transformer is 1.8 = 6 × 0.3, and the length of the convolved response is 1.8 + 13.5 = 15.3 = 12 + 11 × 0.3.

« "COMPUTE FILTER" 3 DISP 11 DHBRT 12 HAMM VECTX → h « { { HOME } 'SIN(x)' x 0 12 40 } 0 CSERS "CONVOLVING" 3 DISP h VMPY { { HOME } '–COS(x)' x –1.8 13.5 51 } 0 CSERS SWAP PLT2 » »

SET GAIN TO 1 AT F	**GAIN1**(N,D,F)	N = NUMERATOR D = DENOMINATOR	2: NUMERATOR 1: DENOMINATOR

Given the numerator and denominator coefficient lists N and D, **GAIN1** sets the gain to 1 at the frequency specified by F by evaluating the gain at F and normalizing. See above examples.

In the z transform case F = z = $e^{i\,2\pi f\,T}$, so to set the DC gain of a lowpass filter, F = 1. To set the gain of a highpass filter, f T = 0.5, so F = –1. For a bandpass filter centered at f T = 0.25, F = (0, 1) or z^{-1} = (0, –1). For a bandstop filter, the DC gain is usually set with F = 1. F must be a real or complex number.

Once the gain is set to 1, « N G SMPY » will set it to G at that frequency.

The following examples demonstrate analog filter normalization and plotting with coefficient lists.

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS

« 5 BPOLE 10000 20000 L→BP R→CL PSCAL (0,15000) GAIN1 DUP2 64000 64 0 FTRV2 VABS PLT1 »

« 5 BPOLE 10000 20000 L→BP R→CL PSCAL (0,15000) GAIN1 DUP2 1000 200000 64 FTRVL VABS VLOG 20 × PLT1 »

It is important to note that the order in which the commands **RZINV**, **R→CL**, and **PZINV** are called does result in different filter gains. For example:

« [0] [-0.6] R→CL PZINV » =
2: { 1 }
1: { 1 0.6 }

« [0] [-0.6] RZINV R→CL » =
2: { 1 }
1: { 1.66666666667 1 }

PSCAL({1.66666666667 1 }) = { 1 0.6 }.

SYMBOLIC INVERSE TRANSFORM	**IXFRM**(F,α,D,ε)	F α D ε > 0	SYMBOLIC EQUATION

IXFRM computes symbolic inverse Laplace and z transforms using residue integration. F is the numerator function, α is the variable of integration, D is the denominator root vector, and ε > 0 is used internally by **RESDA** to compute the residues. **UNIQE** internally determines if there are repeated roots in D. The degree of the numerator must be less than that of D; see the examples.

SEE ALSO THE EXAMPLES GIVEN IN CHAPTER 25.

The following simple example of inverse z transformation calls the IDEM1 program on page 399 and then computes the first six values from the inverse z transform for comparison.

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS

« IDEM1 1 6 VSUBS { } [-1.66666666667] RZINV SWAP CLIST z XEQN 'z^(n-1)' × z ROT .000000001 IXFRM → E « E SWAP 0 5 FOR K E n K 2 →LIST | NEXT 6 →ARRY » »

The filter roots are first converted to polynomial lists. Then the numerator is converted to an algebraic equation, which is multiplied by the inverse z transform kernel 'z^(n-1)'. **IXFRM** then computes the symbolic inverse transform, which is evaluated at the same values as the IDEM1 output for comparison.

The following example designs a digital bandpass IIR filter. The resulting numerator and denominator polynomials have the same order, so they must be divided prior to computing the residues. The quotient in this example equals 1, corresponding to a 1 value at zero in the time domain, which is represented by a 1 for the zero value of the response. The fact that 1 is the zero value of the response is alternatively derivable from the initial value theorem for z transforms. After computing the equation for the impulse response, the following program compares it with the impulse response computed by convolution. The plots are overlaid.

« 2 BPOLE .15 WARP .35 WARP L→BP .5 BILNT DUP2 RZINV DUP2 R→CL PDVD DROP z XEQN 'z^(n-1)' × z 4 ROLL 5 ROLL DROP .0000000001 IXFRM 'n>0' SWAP ROT EVAL IFTE { HOME } OVER n 0 15 16 6 →LIST 0 CSERS RE DUP PLT1 4 ROLL 4 ROLL R→CL PSCAL [0 0 0 0] 20 UNITI ICONV RE 1 16 VSUBS DUP PLT3 DUP2 - »

It is interesting to compare the above digital impulse response with the corresponding one for the analog filter from which the design is based. The following program computes the symbolic inverse Laplace transform and overlays the plots of the digital one. Observe that in the analog case, the initial value theorem for Laplace transforms proves that 0 is the value of the analog impulse response at zero. Observe that the 2π is required since the filter is in Hertz, not radians per sec.

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS

« 2 BPOLE .15 .35 L→BP SWAP CLIST s XEQN 2 π →NUM × s × t × EXP × s ROT .0000000001 IXFRM { HOME } OVER t 0 15 50 6 →LIST 0 CSERS RE PLT3 »

Now the form of the output equation for the inverse Laplace transform may look unfamiliar since the value of e has been evaluated in the equation. The following program provides a more familiar equation in terms of exponentials for the inverse Laplace transform. However, it evaluates significantly slower.

« 2 BPOLE .15 .35 L→BP SWAP CLIST s XEQN 'z^(s×t)' × s ROT .0000000001 IXFRM z 'EXP(2×π)' 2 →LIST ↑MATCH DROP { HOME } OVER t 0 15 50 6 →LIST 0 CSERS RE PLT3 »

The above impulse responses are causal. Since it is real, the anti-causal response associated with the right half plane poles is just a time-reversed response. The sum of these two corresponds to a phase linear filter with poles in the left and right half planes, which can be approximated by truncating the response and inserting a delay. FIR filter design is discussed in Chapter 28.

RATIONAL APPROXIMATION	**RATAP**(L,n)	L n	2: P COEF LIST 1: Q COEF LIST

The inverse FIR problem is the same as the rational approximation problem discussed in Chapter 17 and is solved by the MATHLIB command **RATAP**. Given a polynomial (in this case in z^{-1}), find the rational approximation for it (the IIR filter). For example, suppose we have an impulse response given by L = { 1 −1 1/2 −1/6 1/24 −1/120 }, which in the digital processing world is interpreted as $1 - z^{-1} + z^{-2}/2 - z^{-3}/6 + z^{-4}/24 - z^{-5}/120$. Then one choice of IIR filters that corresponds to this impulse response is given by the infinite impulse response filter:

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS

RATAP(L,4) = $\begin{cases} \text{2: } \{\ 1 \quad -0.6 \quad 0.15 \quad -1/60\ \} \\ \text{1: } \{\ 1 \quad 0.4 \quad 0.05\ \} \end{cases} = \dfrac{1 - 0.6z^{-1} + 0.15z^{-2} - z^{-3}/60}{1 + 0.4z^{-1} + 0.05z^{-2}}.$

The inverse of the **RATAP** command is **PLDVD**, discussed in Chapters 19 and 28.

Techniques are discussed in Chapter 28 for obtaining FIR prototypes.

LIST REAL PART	**SRE**(L)	L = { LIST }	REAL PART
LIST IMAG PART	**SIM**(L)	L = { LIST }	IMAGINARY PART
STATE SPACE TO ZEROS AND POLES	**S→ZP**(A,B,C,D,ε)	MATRICES A, B, C, D ∈ ℂ ε ≥ 0	2: NUMERATOR 1: DENOMINATOR

S→ZP computes the zeros and poles (ε > 0), or the numerator and denominator polynomial lists (ε ≤ 0), of the single-input, single-output state space transfer function defined by the A, B, C, and D matrices. When ε > 0, **S→ZP** uses **AROOT** to solve for the roots with an initial guess of 0 and a maximum number of iterations set at 100. If convergence fails, the current estimate of the roots is returned. See the **AROOT** command for more detail. *Matrix A must be 2 x 2 or larger.*

$$\mathbf{A} = \begin{bmatrix} 2 & 4 & 7 \\ 3 & 5 & 9 \\ 0 & 4 & 2 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 4 \\ 6 \\ 8 \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} 3 & 5 & 9 \end{bmatrix} \qquad \mathbf{D} = \begin{bmatrix} 0 \end{bmatrix}$$

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS

With ε = 0, the numerator and denominator coefficients are

N = { 48 168 114 } D = { −8 −24 −9 1 }.

With ε = 1E−10, the numerator and denominator roots are

N = [−.387723706279 −1.08596050425]

D = [−.39413910354 −1.81137610056 11.2055152041].

Observe that **S→ZP** has properly handled the numerator root at infinity. The DC gain is 48/(−8) = −6, and the gain at infinity is zero. Since one of the denominator roots is in the right half plane, the system is unstable. The commands **EROW** and **ECOL** can be used to extend **S→ZP** to multivariable state space systems. This example is continued below.

Programs and examples are given in Chapter 28 to extend these commands to the discrete state space case, including solving matrix difference equations.

TRANSFER FUNCTION TO CONTROLLABLE FORM	**TF→C**(N,D)	N = NUMERATOR D = DENOMINATOR POLY LISTS	4: A MATRIX 3: B MATRIX 2: C MATRIX 1: D MATRIX

TF→C converts the numerator N and denominator D polynomial lists, such as the output of **S→ZP**, and computes the state matrices in controllable form for a single-input, single-output system.

TF→C({ 48 168 114 } , { −8 −24 −9 1 }) =

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 8 & 24 & 9 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} 48 & 168 & 114 \end{bmatrix} \qquad \mathbf{D} = \begin{bmatrix} 0 \end{bmatrix}$$

The commands **S→ZP**, **TF→C**, and **C↑O** collectively offer an easy way to reduce any single-input, single-output state space system to either controllable or observable canonical form. *Polynomial lists N and D must correspond to a state space system with SIZE(A) ≥ {2 2}. Thus, SIZE(D) ≥ 3. Also, SIZE(N) ≤ SIZE(D)* for realizability.

CONTROLLABLE TO AND FROM OBSERVABLE FORM	**C↑O**(A,B,C,D)	4: A MATRIX 3: B MATRIX 2: C MATRIX 1: D MATRIX	4: A MATRIX 3: B MATRIX 2: C MATRIX 1: D MATRIX

C↑O converts between controllable and observable forms. **C↑O** is actually just a command which transposes the input matrices and swaps B and C. For the above example the result is

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 8 \\ 1 & 0 & 24 \\ 0 & 1 & 9 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 48 \\ 168 \\ 114 \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{D} = \begin{bmatrix} 0 \end{bmatrix}$$

C↑O can also be used to switch between controllable and observable Jordan canonical forms. See the below example.

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
C & O MATRICES	**COMT**(A,E)	A E = B OR C	MATRIX

COMT computes either the controllability or observability matrix for a single-input, single-output state space system. The inputs are the square state matrix A *of size 2 x 2, or greater*, and vector E of appropriate dimension.

Given A and E, if E is a column vector, **COMT** assumes that it is the state input vector B and computes the controllability matrix. If E is a row vector, then **COMT** assumes that E is the state output C vector and computes the observability matrix. For the above example, the controllability and observability matrices are

$$U = [B \vdots AB \vdots \cdots \vdots A^{n-1}B] = \begin{bmatrix} 4 & 88 & 912 \\ 6 & 114 & 1194 \\ 8 & 40 & 536 \end{bmatrix}$$

$$V = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix} = \begin{bmatrix} 3 & 5 & 9 \\ 21 & 73 & 84 \\ 261 & 785 & 972 \end{bmatrix}$$

Since the rank of both matrices is 3, the system is both controllable and observable. A linear time-invariant dynamical system is irreducible if and only if it is both controllable and observable. By using the **EROW** and **ECOL** commands, **COMT** can be extended to the multivariable case.

See page 542 of Golub and Van Loan for a Schur decomposition alternative approach to matrix function evaluation which does not require the less numerically stable Jordan decomposition.

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS

JORDAN CANONICAL FORM

Consider the following dynamical system where the A matrix was studied in Chapter 25.

$$A = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -4 & 4 \\ 0 & -1 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 2 \\ 3 \end{bmatrix}$$

$$C = [2 \;\; 0 \;\; 1] \qquad D = [1]$$

Using **S→ZP**, the polynomial lists are N = { 14 21 8 1 } and D = { 4 8 5 1 }. The zeros and poles are Z = [−1 (−3.5,±1.3228756553)] and P = [−1 −2 −2]. Using **COMT**, the controllability and observability matrices for the system are

$$U = \begin{bmatrix} 0 & 0 & 0 \\ 2 & 4 & -24 \\ 3 & -2 & -4 \end{bmatrix} \qquad V = \begin{bmatrix} 2 & 0 & 1 \\ -2 & -1 & 0 \\ 2 & 4 & -4 \end{bmatrix}$$

where the fact that U has a rank of 2 indicates that the system needs reducing. Using **TF→C**, the controllable canonical form is TF→C(N,D).

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -4 & -8 & -5 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$C = [10 \;\; 13 \;\; 3] \qquad D = [1]$$

Using **COMT** and **HRQR** to compute rank, the controllability and observability matrices are

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS

$$U = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & -5 \\ 1 & -5 & 17 \end{bmatrix} \qquad V = \begin{bmatrix} 10 & 13 & 3 \\ -12 & -14 & -2 \\ 8 & 4 & -4 \end{bmatrix}$$

and the rank of U is 3, while that of V is only 2 (with ε = 1E–8). Using **C↑O** to convert to observable form, we have

$$A = \begin{bmatrix} 0 & 0 & -4 \\ 1 & 0 & -8 \\ 0 & 1 & -5 \end{bmatrix} \qquad B = \begin{bmatrix} 10 \\ 13 \\ 3 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \qquad D = \begin{bmatrix} 1 \end{bmatrix}$$

The U and V matrices for this form are the same as above, except swapped and transposed.

Using **HVSDE** to compute partial fraction expansions, we have, where N and D are defined above:

HVSDE(N , D , –1 , 1E–8) = { 0 }
HVSDE(N , D , –2 , 1E–8) = { 4 3 }

so the transfer function for the system is

$$\frac{0}{s+1} + \frac{4}{(s+2)^2} + \frac{3}{s+2} + 1,$$

where the zero coefficient of the s = –1 term again tells us that the system is unreduced. Taking this into account, the controllable Jordan form is

Jordan canonical form for distinct eigenvalue matrices is discussed in Chapter 20.

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
$\mathbf{A} = \begin{bmatrix} -2 & 1 \\ 0 & -2 \end{bmatrix}$ $\quad \mathbf{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ $\quad \mathbf{C} = [4\ \ 3]$ $\quad \mathbf{D} = [1]$ and **C↑O** will convert this to observable Jordan form. Now that we have demonstrated the power of MATHLIB, let us work the problem correctly. Immediately, observing that N and D have a common root, we eliminate it to get an irreducible system and obtain the new numerator and denominator polynomials N_1 = { 14 7 1 } and D_1 = { 4 4 1 }. Applying **HVSDE** to these lists yields the Jordan form given above.			
POWER SPECTRUM	**SPECT**(V)	V = [VECTOR]	$\lvert V_j \rvert^2$
PHASE IN DEGREES	**PHASE**(V)	V = [VECTOR]	ARG(V_j) × 180/π
SPECT([(1,2) (2,3) (3,4) (4,5)]) = [5 13 25 41] PHASE([(1,2) (2,3) (3,4) (4,5)]) = [63.4349 56.3099 53.1301 51.3402]			
VECTOR LOG	**VLOG**(V)	V = [VECTOR]	LOG(V_j)
LIST ROUND	**SRND**(L,N)	L = { LIST }	ROUNDED LIST
SRND applies < N RND > to each element of the list.			
PLOT REAL VECTOR	**PLT1**(V)	V ∈ **R**	PLOT OF V
PLOT VECTOR	**PLTC**(V)	V ∈ **C**	PLOT VR, VI

FILTER DESIGN MENU { FILTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
PLOT VECTOR	**PLT3**(V)	V $\in \mathbf{R}$	PLOT OF V
UP DIRECTORY	**UPDIR**	NONE	PARENT MENU

Bronson, R., *Matrix Methods*, Boston, Academic Press, 1991.

Gardner, W., *Statistical Spectral Analysis*, Englewood Cliffs: NJ, Prentice-Hall, 1988.

Golub, G., and Van Loan, C., *Matrix Computations*, Baltimore, John Hopkins Univ Press, 1989.

Guillemin, E., *Synthesis of Passive Networks*, New York, Wiley, 1957.

Haddad, R., and Parsons, T., *Digital Signal Processing*, New York, Computer Science Press, 1991.

Haykin, S., *Modern Filters*, New York, MacMillan, 1989.

Herstein, I., and Winter, D., *Matrix Theory and Linear Algebra*, New York, Macmillan, 1988.

Oppenheim, A., and Schafer, R., *Digital Signal Processing*, Englewood Cliffs: NJ, Prentice-Hall, 1975.

Oppenheim, A., and Schafer, R. , *Discrete-Time Signal Processing*, Englewood Cliffs: NJ, Prentice-Hall, 1989.

Parks, T., and Burrus, C., *Digital Filter Design*, New York, Wiley, 1987.

Proakis, J., and Manolakis, D., *Introduction to Digital Signal Processing*, New York, MacMillan, 1988.

Rabiner, L., and Gold, B., *Theory and Application of Digital Signal Processing*, Englewood Cliffs: NJ, Prentice-Hall, 1975.

Roberts, R., and Mullis, C., *Digital Signal Processing*, Reading: MA, Addison-Wesley, 1987.

Stearns, S., and David, R., *Signal Processing Algorithms*, Englewood Cliffs: NJ, Prentice-Hall, 1988.

Van Valkenburg, M., *Introduction to Modern Network Synthesis*, New York, Wiley, 1967.

Zverev, A., *Handbook of Filter Synthesis*, New York, John Wiley & Sons, 1967.

28

FIR DESIGN AND DISCRETE COMPUTATIONS

INTRODUCTION

This chapter presents the 36 FIR design and discrete computation commands in the WIND menu. The WIND commands provide most of the data windows used in digital signal processing. Three clipping commands are also provided, in addition to filter parameter conversion and FIR design commands. Discrete Chebyshev polynomial and Wiener–Levinson approximations are also included in this chapter. Solving discrete state space difference equations and transfer functions is covered at the end.

WINDOWING

The WIND menu provides the following windows:

Hamming	**HAMM**	Gaussian	**GAUS**
General Hamming	**GENH**	Parzen	**PARZ**
Hanning	**HANN**	Kaiser	**KAISR**
Bartlett	**BARTL**	Welch	**WELC**
Blackman	**BLAC**	Rectangular	**ONE**

which are my versions of these window functions, obeying the FFT symmetry properties that are fundamental to the application of discrete Fourier transforms to continuous problems.

CLIPPING

CLIPP, **CLIPN**, and **CLIPB** provide positive, negative, and both positive and negative clipping, respectively.

FIR DIGITAL FILTER DESIGN AND APPROXIMATION

Programs and examples of finite impulse response (FIR) digital filter designs are provided. **FIRID** provides designs based on ideal prototypes, and **IXFRM** and **PLDVD** provide designs based on arbitrary prototypes.

DISCRETE CHEBYSHEV POLYNOMIAL APPROXIMATION

FMAT, **CHEBY**, **DOCAP**, and **STRLT** provide discrete orthogonal Chebyshev polynomial approximations to digital impulse responses and digital waveforms.

ADAPTIVE FILTERS USING WIENER–LEVINSON SOLUTIONS

WL1 and **WL2** provide additional least squares design techniques for digital filters.

DISCRETE HILBERT TRANSFORMER DESIGN

DHBRT in the FILTR menu of Chapter 27 designs digital Hilbert transformers.

DISCRETE STATE SPACE DIFFERENCE EQUATIONS

Programs are given demonstrating the computation of discrete state transition matrices for state space difference equations. Calculation of transfer functions and impulse responses is also demonstrated.

WAVELET TRANSFORMS

Perfect reconstruction filter banks and compactly supported wavelet transforms are discussed in Appendix G. Example Haar wavelet matrix programs are also given.

FIR DESIGN AND DISCRETE COMPUTATIONS MENU { WIND }

FUNCTION	COMMAND	INPUTS	OUTPUTS
HAMMING	**HAMM**(N)	N $\in \mathbb{N}$ N EVEN	[WINDOW]
$.54 - .46 \cos(2\pi k/N)$ for $k \in [0, N-1]$ $N \geq 4$			
GENERAL HAMMING	**GENH**(α,N)	N $\in \mathbb{N}$ N EVEN	[WINDOW]
$\alpha - (1 - \alpha) \cos(2\pi k/N)$ for $k \in [0, N-1]$ $\alpha \in [0, 1]$ $N \geq 4$			
HANNING	**HANN**(N)	N $\in \mathbb{N}$ N EVEN	[WINDOW]
$.5 - .5 \cos(2\pi k/N)$ for $k \in [0, N-1]$ $N \geq 4$			
BARTLETT	**BARTL**(N)	N $\in \mathbb{N}$ N EVEN	[WINDOW]
$1 - \text{ABS}(2k - N)/N$ for $k \in [0, N-1]$ $N \geq 4$			
BLACKMAN	**BLAC**(N)	N $\in \mathbb{N}$ N EVEN	[WINDOW]
$.42 - .5 \cos(2\pi k/N) + .08 * \cos(4\pi k/N)$ for $k \in [0, N-1]$ $N \geq 4$			
GAUSSIAN	**GAUS**(α,N)	N $\in \mathbb{N}$ N EVEN	[WINDOW]
$\text{EXP}(-\alpha(k - N/2)\text{^}2)$ for $k \in [0, N-1]$ $\alpha > 0$ $N \geq 4$			

FIR DESIGN AND DISCRETE COMPUTATIONS MENU { WIND }

FUNCTION	COMMAND	INPUTS	OUTPUTS
PARZEN	**PARZ**(N)	N ∈ ℕ N EVEN	[WINDOW]
$1 - \mathrm{ABS}(2k - N)/(N + 1)$ for $k \in [0, N-1]$ $N \geq 4$			
KAISER	**KAISR**(β,N)	N ∈ ℕ N EVEN	[WINDOW]
$\frac{I_0\left[\beta\sqrt{1 - ([k - N/2]/N)^2}\right]}{I_0[\beta]}$ for $k \in [0, N-1]$ $\beta > 0$ $N \geq 4$			
WELCH	**WELC**(N)	N ∈ ℕ N EVEN	[WINDOW]
$1 - [(2k - N)/(N + 1)]^2$ for $k \in [0, N-1]$ $N \geq 4$			

PLOTTING THE WINDOWS

The following program will plot all of the above windows:

« → L W « L EVAL W OBJ→ DUP PLT1 L LREV 1 GET 2 / VROT PLT1 » »

where list L contains the window parameters, and W is a string with the name of the desired window. For example, let L = { 1 64 } and W = "KAISR" and execute the program. First the window will be plotted with the peak in the center. Press ATTN, and the window will be plotted with the peak value at zero.

FIR DESIGN AND DISCRETE COMPUTATIONS MENU { WIND }

FUNCTION	COMMAND	INPUTS	OUTPUTS
CLIP POSITIVE	**CLIPP**(V,CL)	V = [VECTOR] CL = CLIP LEVEL	[WINDOW]
IF V(k) > CL THEN V(k) = CL.			
CLIP BOTH	**CLIPB**(V,CL)	V = [VECTOR] CL = CLIP LEVEL	[WINDOW]
IF |V(k)| > CL THEN V(k) = SIGN(V(k)) CL.			
CLIP NEGATIVE	**CLIPN**(V,CL)	V = [VECTOR] CL = CLIP LEVEL	[WINDOW]
IF V(k) < –CL THEN V(k) = –CL.			
EXAMPLES OF CLIPPING OPERATIONS Let V = [1 8 –2 4 –4 6 3]. Then CLIPP(V,5) = [1 5 –2 4 –4 5 3], CLIPB(V,3) = [1 3 –2 3 –3 3 3], and CLIPN(V,2) = [1 8 –2 4 –2 6 3].			
RECTANGULAR	**ONE**(N)	SIZE = N	[VECTOR]
The rectangular window is simply a vector of all ones. ONE(5) = [1 1 1 1 1].			

FIR DESIGN AND DISCRETE COMPUTATIONS MENU { WIND }

FUNCTION	COMMAND	INPUTS	OUTPUTS
IDEAL FIR DESIGN	**FIRID**(FL,FH,L,C)	FL FH L C FL, FH $\in$ [0, 0.5]	[VECTOR]

FIRID performs FIR filter designs based on ideal prototype filters. FL and FH are the normalized (see page 402) upper and lower frequency limits in Hertz seconds. For lowpass designs, FH is arbitrary, and for highpass designs, FL is arbitrary. L is the filter length. C is the case:

C = 1: LOWPASS C = 2: HIGHPASS C = 3: BANDPASS C = 4: BANDSTOP

The equations for the impulse responses in these cases for n = 0, 1, . . ., L–1 are

$$h_{LP}(n) = 2\ FL\ SINC[2\pi\ FL\ (n-L/2)],$$

$$h_{HP}(n) = SINC[\pi(n-L/2)] - 2\ FH\ SINC[2\pi\ FH\ (n-L/2)],$$

$$h_{BP}(n) = 2\ FH\ SINC[2\pi\ FH\ (n-L/2)] - 2\ FL\ SINC[2\pi\ FL\ (n-L/2)],$$

$$h_{BS}(n) = 2\ FL\ SINC[2\pi\ FL\ (n-L/2)] + SINC[\pi\ (n-L/2)] - 2\ FH\ SINC[2\pi\ FH\ (n-L/2)].$$

For a good discussion of the theory, see Stearns and David.

Their examples in Chapter 8 are computed with the demo program FIR:

« → FL FH L P W C « FL FH L C FIRID DUP PLT1 P OBJ→ DROP W OBJ→ DUP PLT3 VECTX DUP PLT3 DUP 2 L LN 2 LN / CEIL ^ SWAP OVER ZFILN SWAP TWIDL FFT DUP VABS PLT1 DUP VTRUD PLT3 » »

FIR DESIGN AND DISCRETE COMPUTATIONS MENU { WIND }

FUNCTION	COMMAND	INPUTS	OUTPUTS

P is a parameter list to be used by the window command specified by string W.

FIR(0.2 , 0 , 20 , {20} , "HAMM" , 1) = [0 −.00345 −.00393 .00721 .02007 0 −.05163 −.05054 .08533 .29592 .4 .29592 .08533 −.05054 −.05163 0 .02007 .00721 −.00393 −.00345]

FIR(0 , 0.25 , 20 , {20} , "ONE" , 2) = [0 −.03537 0 .04547 0 −.06366 0 .1061 0 −.31831 .5 −.31831 0 .1061 0 −.06366 0 .04547 0 −.03537]

FIR(0.15 , 0.35 , 20 , {20} , "HAMM" , 3) = [0 0 −.0127 0 .02481 0 .06381 0 −.27614 0 .4 0 −.27614 0 .06381 0 .02481 0 −.0127 0]

FIR(0.15 , 0.35 , 20 , {20} , "HANN" , 4) = [0 0 .00723 0 −.02155 0 −.06123 0 .27382 0 .6 0 .27382 0 −.06123 0 −.02155 0 .00723 0]

The filter coefficients are returned to Level 2 of the stack and the FFT of them to Level 1 of the stack. Observe that the filter phase is only linear in the passband.

SYMBOLIC INVERSE TRANSFORM	**IXFRM**(F,α,D,ε)	F α D ε > 0	SYMBOLIC EQUATION

IXFRM computes symbolic inverse Laplace and z transforms using residue integration. F is the numerator function, α is the variable of integration, D is the denominator root vector, and ε > 0 is used internally by **RESDA** to compute the residues. **UNIQE** internally determines if there are repeated roots in D. The degree of the numerator must be less than that of D.

See the examples in Chapters 25 and 27.

FIR DESIGN AND DISCRETE COMPUTATIONS MENU { WIND }

FUNCTION	COMMAND	INPUTS	OUTPUTS

IXFRM EXAMPLE

Given the filter z transform, **IXFRM** is one of the methods for computing the impulse response for FIR approximation. The following program does a complete phase linear FIR filter design based on an IIR filter derived from a Butterworth analog design. Since the numerator and denominator polynomials have the same degree, the constant corresponding to the time equals zero value in the time domain, must be divided out. The quotient is then the value at 0. Given the equation for the impulse response, **CSERS** is then used to generate L (say 32) values of the impulse response, which is windowed with a rectangular window (**ONE**) and FFT'd to the frequency domain for plotting. The original IIR filter plot is then overlaid for comparison. Multiplication with **VECTX** by the ±1 output of **EINDX** performs the phase linear modification of the transfer function so that after it is inverse FFT'd, it is properly rotated in the time domain for phase linear filtering. The output filter coefficients are plotted and returned to the stack.

```
« 32 → L « 6 BPOLE .1 WARP L→LP .5 BILNT → N D « N D RZINV
 DUP2 R→CL PDVD DROP z XEQN 'z^(n-1)' × z 4 ROLL 5 ROLL DROP
 .00000001 IXFRM 'n>0' SWAP ROT EVAL IFTE { HOME } OVER n 0 L 1
 - L 6 →LIST 0 CSERS RE DUP PLT1 DUP L ONE VECTX 2 L LN 2
 LN / CEIL ^ DUP 'L' STO SWAP OVER ZFILN SWAP TWIDL FFT VABS
 DUP HALF1 PLT1 N D 1 L 1 FTRV2 VABS DUP HALF1 PLT3 OVER -1
 L 2 / 1 + EINDX REFLT VECTX L TWIDL IFFT RE DUP PLT1 » » »
```

Hence, while the **FIRID** design is always based on ideal prototype filters, **IXFRM** and **PLDVD** below allow you to base the design on almost any prototype filter with which you wish to begin. More examples on the use of **IXFRM** are given in Chapters 25 and 27.

This and the below **PLDVD** example can be generalized to filter lengths that are not a power of 2, using the DFT and IDFT programs on page 390 of Chapter 26.

FIR DESIGN AND DISCRETE COMPUTATIONS MENU { WIND }

FUNCTION	COMMAND	INPUTS	OUTPUTS
POLYNOMIAL LONG DIVISION	**PLDVD**(L1,L2,n)	COEFFICIENT LISTS L1 & L2 $n \in \mathbb{N}$	3: QUOTIENT 2: REMAINDER 1: DIVISOR

PLDVD provides a direct means of approximating a digital IIR filter function. For more details, see Chapter 19. The following example is like the above one, but uses **PLDVD** instead of **IXFRM**.

« 32 → L « 6 BPOLE .1 WARP L→LP .5 BILNT R→CL PSCAL 1 GAIN1 → N D « N D L PLDVD DROP LREV HALT DROP DUP L ONE VECTX 2 L LN 2 LN / CEIL ^ DUP 'L' STO SWAP OVER ZFILN SWAP TWIDL FFT VABS DUP HALF1 PLT1 N D 1 L 1 FTRV2 VABS DUP HALF1 PLT3 OVER −1 L 2 / 1 + EINDX REFLT VECTX L TWIDL IFFT RE DUP PLT1 » » »

After completing the long division, the program halts so you can examine the magnitude of the remainder. Push CONT, and the program will complete and plot the design. The underlying IIR response is then overlaid for comparison. **EINDX** is used to rotate the FIR coefficients into proper position. **PLDVD** provides a phase linear elliptic filter-based design alternative to a Parks–McClellan iterative optimum design.

ORTHOGONAL POLYNOMIAL APPROXIMATION MATRIX	**FMAT**(P,F,N)	3: PARAM LIST 2: POLYNOMIAL 1: DEGREE	FORWARD TRANSFORM MATRIX

FMAT is like **PMAT** discussed in Chapter 17 except that it is less general, since offsets are not possible, and faster, since the input polynomial function such as **TOFXL** creates polynomial lists directly and not symbolic algebraic expressions which must be reduced to Maclaurin series coefficient lists.

FIR DESIGN AND DISCRETE COMPUTATIONS MENU { WIND }

FUNCTION	COMMAND	INPUTS	OUTPUTS

FMAT combined with **TOFXL** provides a useful set of transformations in the study and design of optimum phase linear FIR filters such as ones designed by the Remez exchange algorithm.

As discussed in Rabiner and Gold, a linear phase FIR filter can be written as

$$H(e^{i\omega}) = H^*(e^{i\omega})\, e^{i(\beta-\alpha\omega)},$$

where $H^*(e^{i\omega})$ is purely real and can be expressed as a polynomial in $\cos(n\omega) = T_n(\cos\omega)$ for $\omega \in [0, \pi]$. Defining $x = \cos\omega$, as discussed in Chapter 17, **PMAT** and **FMAT** provide polynomial coefficient transformations between polynomials in $\cos(n\omega)$ and polynomials in $\cos\omega = x \in [-1, 1]$.

The forward matrix expresses each of the orthogonal polynomials in terms of a power series in x^n where x is some arbitrary variable. The inverse matrix expresses the x^n in terms of the polynomials. The following example begins with a polynomial list in $T_n(z)$ and converts it to a list in z^n. The polynomial in z^n is then converted back to one in $T_n(z)$.

« { 4 2 6 3 8 } → L « { } "TOFXL" L SIZE 1 – FMAT DUP INV → TF TI « L V←L →ROW TF × DUP →VTR V→L z XEQN HALT DROP TI × →VTR V→L 10 SRND "T" PEQN » » »

MATRIX INVERSE	**MINV**(M)	MATRIX M	INVERSE MATRIX
MINV is slower than **INV**, but more accurate. It solves MX = I for X with **LSOVR**.			
CHEBYSHEV $T_n(x)$	**TOFXL**(n)	n = DEGREE n = 0, 1, ...	COEFFICIENT LIST

FIR DESIGN AND DISCRETE COMPUTATIONS MENU { WIND }

FUNCTION	COMMAND	INPUTS	OUTPUTS

TOFXL computes a coefficient list for the Chebyshev polynomial $T_n(x)$. Its equation is given by the **TOFX** command in the POLY menu discussed in Chapter 16.
TOFXL(4) = { 1 0 –8 0 8 }.

DISCRETE CHEBYSHEV POLYNOMIALS	**CHEBY**(N,n)	N + 1 = NUMBER OF DATA POINTS n = DEGREE	COEFFICIENT LIST

The discrete Chebyshev polynomial $f_n(x)$ is an orthogonal polynomial of the discrete variable x taking on the values $x_m = m$ (m = 0, 1, 2, . . ., N) which is defined by:

$$f_n(x) = \sum_{m=0}^{n} (-1)^m \binom{n}{m}\binom{n+m}{m} \frac{x!\,(N-m)!}{(x-m)!\,N!} \qquad n = 0, 1, \ldots, N$$

where integer n is the degree of the polynomial.

CHEBY returns a coefficient list for the polynomials. Use **FMAT** to build a transformation matrix. The coefficients are given by the above equation with the x!/(x – m)! = PERMF(x, m) deleted.

DISCRETE CHEBYSHEV APPROXIMATION	**DOCAP**(y,T,L)	y(n) n = 0,1 . . ., N T = PERIOD L = ORDER	3: b COEF VECT 2: M MATRIX 1: β COEF LIST

DOCAP performs least squares discrete orthogonal Chebyshev polynomial approximations using **FMAT**, **CHEBY**, and **STRLT** below.

FIR DESIGN AND DISCRETE COMPUTATIONS MENU { WIND }

FUNCTION	COMMAND	INPUTS	OUTPUTS

Given data y, sample period T > 0, and the order of the approximation L, $2 \le L \le N$ where N + 1 is the size of y, **DOCAP** constructs the least squares estimate of y:

$$\hat{y} = \sum_{n=0}^{L} b_n f_n(x), \qquad b_n = \frac{\sum_{k=0}^{N} y(k) f_n(k)}{\sum_{k=0}^{N} f_n^2(k)}.$$

It then uses **FMAT** to transform the coefficients into a polynomial in PERMF(x,m). Finally, it uses **STRLT** to transform the coefficients into a polynomial in t where t = T x, T is the sampling period, and β_n are the coefficients.

$$\hat{y} = \sum_{n=0}^{L} \beta_n t^n$$

The variable x in these equations must be an integer, and the sampling must be uniform. The b coefficients and the b to β transformation matrix M are also returned. See Chapter 22 of AMS 55 and Stearns and David for more details.

```
« 50 5 .02 0 → N L T t « 0 N FOR x x T × 't' STO
'10×t^3-3×t^2-5×t-1' →NUM NEXT N 1 + →ARRY T L DOCAP » »
```

The b and β coefficient outputs of this test program are

b = [[−1.96 −.5296 1.95999999998 −.470400000085 5.333E−10 −1.476E−9]],

β = { −1.00000000105 −4.99999995916 −3.00000032266 10.0000009285
−1.10064822985E−6 4.57144676025E−7 }.

FIR DESIGN AND DISCRETE COMPUTATIONS MENU { WIND }

FUNCTION	COMMAND	INPUTS	OUTPUTS

Now it is important to understand that the transformation from the b to the β coefficients is not orthogonal. You therefore must know how many β coefficients are required, or you will introduce unnecessary errors into your β coefficients, which in larger problems can be huge. Since the b coefficients are associated with orthogonal polynomials, the magnitude of the last two coefficients indicates that they should be zero. Replacing the last two b coefficients with 0 and multiplying (b × M = β) yields the β coefficients you will get if you only ask for L = 3 coefficients in the above example. **V→L** will make the β vector into a list.

β = { −1.00000000011 −4.99999999881 −3.0000000028 10.0000000018 0 0 }

These β coefficients are 1 to 2 orders of magnitude more accurate than those above.

STIRLING TRANSFORM	**STRLT(L)** L ≥ 2	L = APPROXIMATION ORDER	COEFFICIENT MATRIX

STRLT sets up a matrix for easily implementing the following relation for Stirling numbers of the first kind, which allows PERMF(x,n) to be expressed as a polynomial in x.

$$x(x-1)(x-2)\ \ldots\ (x-n+1) = \mathrm{PERMF}(x,n) = \sum_{m=0}^{n} S_n^{(m)}\, x^m$$

Using the notation in AMS 55, the elements of the (L + 1) × (L + 1) matrix are

FIR DESIGN AND DISCRETE COMPUTATIONS MENU { WIND }

FUNCTION	COMMAND	INPUTS	OUTPUTS

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & S_2^{(1)} & 1 & 0 & \cdots & 0 \\ 0 & S_3^{(1)} & S_3^{(2)} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & S_L^{(1)} & S_L^{(2)} & S_L^{(3)} & \cdots & 1 \end{bmatrix}$$

$$S_0^{(0)} = 1$$
$$S_n^{(0)} = 0 \qquad n > 0.$$
$$S_n^{(n)} = 1$$

This matrix can be used to transform a polynomial in PERMF(x,n) into one in x.

Stirling numbers of the first kind are discussed in Chapter 18.

WIENER–LEVINSON SOLUTION	**WL1**(V)	V = COVARIANCE VECTOR V ∈ ℂ	2: h VECTOR 1: e VECTOR
WIENER–LEVINSON SOLUTION	**WL2**(RHS,V)	RHS VECTOR V = COVARIANCE VECTOR ∈ ℂ	2: h VECTOR 1: e VECTOR

WL1 and **WL2** provide solutions to Hermitian Toeplitz systems of linear equations: R h = ϕ, where R is an N × N Toeplitz matrix such as an autocovariance matrix, h is the unknown column vector, and ϕ is the known "right-hand side" (RHS) column vector. In statistical applications h represents the minimum mean square error filter associated with the Wiener–Hopt Equation defined by R and ϕ. The V argument in **WL1** and **WL2** is the first column of the R matrix. There are two cases:

FIR DESIGN AND DISCRETE COMPUTATIONS MENU { WIND }

FUNCTION	COMMAND	INPUTS	OUTPUTS

WL1 provides the solution in the case where TRNP(ϕ) = [P_N 0 0 ... 0] and P_N is the resulting mean square error associated with filter solution h.

WL2 provides the solution in the general case where ϕ is equal to the RHS vector.

With both solutions the sequence of errors is returned to Level 1 of the stack in a vector.

The size of V and RHS must be the same and be 2 or greater. **WL1** and **WL2** provide statistical least squares solutions. If you specify a singular (noise-free) deterministic problem, expect to get a *divide-by-zero error* during the calculation!

The program:

« 0 15 FOR K 'EXP(-K^2)' →NUM NEXT 16 →ARRY DUP PLT1 HALT WL1 PLT1 PLT1 »

will create a covariance vector in Level 1 of the stack and HALT. Press CONT and **WL1** will compute the solution and plot it. The errors in the solution will be plotted first. Then the solution is plotted. The first value of e is V(1), which must be real, and the last value of e is the number P_N.

The next program illustrates **WL2**:

« 16 UNITI 0 15 FOR K 'EXP(-K^2)' →NUM NEXT 16 →ARRY DUP PLT1 HALT WL2 PLT1 PLT1 »

FIR DESIGN AND DISCRETE COMPUTATIONS MENU { WIND }

FUNCTION	COMMAND	INPUTS	OUTPUTS

DESIGN OF FIR ADAPTIVE INTERFERENCE CANCELERS

One interesting application of the Wiener–Levinson algorithm is the design of adaptive interference cancelers, which whiten the environment by placing notches where there are interferers. The below programs E3 and E6 create interference environments characterized by their power spectral densities (PSD) and put the power spectrum on the stack. WF2 plots the input PSD, constructs a filter from that PSD, which is also plotted, and plots the filter output response after convolution. In practice the PSD would have been estimated from the received signal by any number of methods discussed in the literature. Many of these techniques can be implemented by MATHLIB.

```
E3: « { 8 } (1,0) CON 3 (80,80) PUT 7 (80,-80) PUT »
```

```
E6: « { 33 } (.035,0) CON 10 (4,4) PUT 12 (6,4) PUT 14 (8,5) PUT 16
    (10,0) PUT 18 (4,8) PUT 20 (4,6) PUT 22 (4,1) PUT REFLT »
```

```
WF2: « "LOG PLOT INPUT PSD" 3 DISP DUP DUP SIZE EVAL 0 → N
PWR « SPECT DUP VLOG 10 × DUP VMIN N 2 / 1 + SWAP R→C
"FREQ" "PWR DB" 3 →LIST PLT1L N TWIDL IFFT DUP 1 GET ABS
'PWR' STO RE DUP SIZE 0 CON 1 1 PUT SWAP WL2 VMIN PWR /
LOG 10 × SWAP N TWIDL FFT VSRT "LOG PLOT FILTER" 3 DISP DUP
SPECT VLOG 10 × DUP VMIN N 2 / 1 + SWAP R→C "FREQ" "PWR
DB" 3 →LIST PLT1L "LOG PLOT OF OUTPUT" 3 DISP ROT OVER VECTX
SPECT VLOG 10 × DUP VMIN N 2 / 1 + SWAP R→C "FREQ" "PWR
DB" 3 →LIST PLT1L » »
```

Now it is very important to observe not only the shape of the plots, but also the variation in dB of the power levels, since while the final output plot is not absolutely flat (white), the peak-to-peak interference power has been reduced by orders of magnitude in dB. Thus, the filter canceled the interference and did an extremely good job, given only its PSD.

FIR DESIGN AND DISCRETE COMPUTATIONS MENU { WIND }

FUNCTION	COMMAND	INPUTS	OUTPUTS
VOLTS TO dB	**V→DB**(V)	VALUE IN VOLTS > 0	VALUE IN dB
V IN dB = 20 LOG(V)			
dB TO VOLTS	**V←DB**(V)	VALUE IN dB	VALUE IN VOLTS
NEPERS TO dB	**N→DB**(V)	VALUE IN NEPERS	VALUE IN dB
V IN dB = 8.68588963808 × V			
dB TO NEPERS	**N←DB**(V)	VALUE IN dB	VALUE IN NEPERS
δ η PARAMETER CONVERSION	δη→**A**(δ,η)	2: δ IN VOLTS > 0 1: η IN VOLTS > 0	2: a_{max} dB 1: a_{min} dB
If the passband peak–to–peak response is specified as varying from 1 + δ to 1 – δ and the maximum stopband response is η, then: a_{min} = 20 LOG([1+δ] / η) and a_{max} = 20 LOG ([1+δ] / [1–δ]). Rabiner and Gold use this notation where δ = δ_1 and η = δ_2.			
δ η PARAMETER CONVERSION	δη←**A**(a_{max},a_{min})	2: a_{max} dB 1: a_{min} dB	2: δ 1: η

FIR DESIGN AND DISCRETE COMPUTATIONS MENU { WIND }

FUNCTION	COMMAND	INPUTS	OUTPUTS
REFLECTION COEFFICIENT CONVERSION	ρ→**AX**(ρ)	REFLECTION COEFFICIENT ρ ρ ∈ [0, 100)	a_{max} IN dB

Computes a_{max} in dB given the reflection coefficient in percent.

$a_{max} = -10 \times LOG(1-SQ(\rho/100))$ $\rho = 100\sqrt{(1 - ALOG(-a_{max}/10))}$

The voltage-standing-wave ratio VSWR = $(1+\rho/100)/(1-\rho/100)$.

REFLECTION COEFFICIENT CONVERSION	ρ←**AX**(a_{max})	a_{max} IN dB $a_{max} \geq 0$	REFLECTION COEFFICIENT ρ
RIPPLE FACTOR CONVERSION	**AX**→ε(a_{max})	a_{max} IN dB $a_{max} \geq 0$	ε

$\varepsilon = \sqrt{10^{0.1a_{max}} - 1}$ $a_{max} = 10LOG(1 + \varepsilon^2)$

RIPPLE FACTOR CONVERSION	**AX**←ε(ε)	ε ≥ 0	a_{max} IN dB
VECTOR MPY	**VECTX**(V1,V2)	V1 V2	$V1_j \times V2_j$

VECTX is useful for applying windows to filters.

VECTX([3 5 4 7 1] , [4 2 3 1 5]) = [12 10 12 7 5]

UP DIRECTORY	**UPDIR**	NONE	PARENT MENU

FIR DESIGN AND DISCRETE COMPUTATIONS MENU { WIND }

FUNCTION	COMMAND	INPUTS	OUTPUTS

DISCRETE STATE SPACE SYSTEMS

See Chapters 25, 26, and 27 for state space conventions and techniques. In z transform notation for A, B, C, D $\in \mathbb{C}$, the matrix state equations are

$$z\,X(z) = A\,X(z) + B\,U(z), \qquad Y(z) = C\,X(z) + D\,U(z).$$

where A is a square matrix and D is a 1×1 matrix.
The below programs assume that the SIZE(A) ≥ {2 2}.

The following program will compute the symbolic state transition matrix $\Phi(n)$ by computing the inverse z transform of $z\,(zI + A)^{-1}$. Note that $z\,z^{(n-1)} = z^n$.

```
ZASOV: « → A « A SIZE → S « A 1 SMI SWAP 0 1E-10 100 AROOT
'V' STO « z XEQN 'z^n' × z V 1E-10 IXFRM »
L1F1 'V' PURGE SOB→ DROP S →SOB » » »
```

The next program uses **IXFRM** to compute the symbolic impulse response from the z transform transfer function which is computed by command **S→TF**.

```
IXFRMD: « → A B C D ε « A B C D ε S→ZP DUP2 R→CL PDVD
DROP z XEQN 'z^(n-1)' × z 4 ROLL 5 ROLL DROP 1E-10 IXFRM IF
OVER SABS 0 > THEN 'n>0' SWAP ROT EVAL IFTE ELSE SWAP DROP
END » » »
```

The theory is given in Chapter 7 of Proakis and Manolakis. For example, let

A = [[0 1] [1 1]], B= [[0] [1]], C = [[1 1]], D = [[1]], ε = 1E–10.

Then the above programs compute the same results as given in Example 7.5.9 of Proakis and Manolakis. While the MATHLIB output symbolic equations are not as pretty as those in the the text, the answers do agree to 10 digits.

FIR DESIGN AND DISCRETE COMPUTATIONS MENU { WIND }

FUNCTION	COMMAND	INPUTS	OUTPUTS

Abramowitz, M., and Stegun, I., *Handbook of Mathematical Functions*, AMS 55, 1964, Chapter 22.

Chen, C., *Introduction to Linear System Theory*, New York, Holt Rinehart and Winston, 1970.

Desoer, C., *Notes for a Second Course on Linear Systems*, New York, Van Nostrand Reinhold, 1970.

Gold, B., and Rader, C., *Digital Processing of Signals*, New York, McGraw–Hill, 1969.

Oppenheim, A., and Schafer, R., *Digital Signal Processing*, Englewood Cliffs: NJ, Prentice-Hall, 1975.

Oppenheim, A., and Schafer, R. , *Discrete-Time Signal Processing*, Englewood Cliffs: NJ, Prentice-Hall, 1989.

Parks, T., and Burrus, C., *Digital Filter Design*, New York, Wiley, 1987.

Proakis, J., and Manolakis, D., *Introduction to Digital Signal Processing*, New York, MacMillan, 1988.

Rabiner, L., and Gold, B., *Theory and Application of Digital Signal Processing*, Englewood Cliffs: NJ, Prentice-Hall, 1975.

Roberts, R., and Mullis, C., *Digital Signal Processing*, Reading: MA, Addison-Wesley, 1987.

Shynk, J., "Frequency-Domain and Multirate Adaptive Filtering," *IEEE ASSP Magazine*, Vol 9, No 1, January, 1992.

Stearns, S., and David, R., *Signal Processing Algorithms*, Englewood Cliffs: NJ,

Zadeh, L., and Desoer, C., *Linear System Theory*, New York, McGraw-Hill, 1963.

29

SPECIAL VECTOR OPERATIONS

INTRODUCTION

This chapter presents the 24 special vector operations in the VECTR menu. The VECTR commands include object conversion, sorting, rearrangement, as well as numerous primitives for extracting and replacing subsets of elements. They allow the user to manipulate the data in vectors without putting the elements on the stack.

OBJECT TYPE CONVERSION

The commands **V→L** and **V←L** provide conversions between vector and list objects. **→VTR** converts both column and row vectors (matrix objects) into vector objects. It is the inverse of the commands **→ROW** and **→COL** found in the MATR menu. If the input to **→VTR** is already a vector, it simply returns that vector.

SORTING

SRT↑ and **SRT↓** sort vectors and put the elements into ascending or descending order. The command **SRTI** combines **SRT↑** with bookkeeping so that an index is output defining all the position interchanges. This is useful for sorting matrices based on vectors using **RORDR** and **CORDR**.

REVERSING

VREV and **LREV** reverse the order of the elements in vectors and lists.

ELEMENT MANIPULATIONS

Seventeen additional commands provide for value and subset inserting, deleting, swapping, copying, and moving, as well as vector splitting and splicing.

ADVANCED APPROXIMATIONS WITH SERIES

The **VSUBS** command affords a number of interesting advanced series approximations. In the below examples, let $f(z) \sim \Sigma\, a_n z^n$ be the N term Maclaurin series approximation for f(z) represented by polynomial list L as defined on page 193.

POWER: The Maclaurin series polynomial list for $[f(z)]^m$ is computed by PTOM:

```
« → L m « L SIZE → N « L 2 m START L PMPY 1 N VSUBS NEXT » »
```

REVERSION OF SERIES: Let $w = \Sigma\, a_n z^n$. Find coefficients b_n such that $z = \Sigma\, b_n w^n$.

```
REVS: « → L « L SIZE → N « N UNITI V→L L 3 N START DUP L PMPY
      1 N VSUBS NEXT N →LIST M←SO MINV 2 EROW V→L » » »
```

Observe that $z = f^{-1}[\, f(z)\,] = f^{-1}[w] \sim \Sigma\, b_n w^n$ is the inverse of the f(z) function.

INVERSE AND RATIO OF SERIES: See the **PLDVD** command on page 204.

COMPOSITION OF TWO SERIES: Let f(z) be defined as above and also define $g[w] \sim \Sigma\, c_n w^n$ to be the first N terms of the Maclaurin series for g[w] represented by list G. Then the first N terms of $g[\, f(z)\,]$ are computed by the below program CFG.

```
« → L G « G SIZE → N « G N N VSUBS N 1 - 1 FOR K L PMPY G K
    K VSUBS PADD 1 N VSUBS -1 STEP » » »
```

For L = { 2 −.5 .01 −.005 .00004 }, f(0.1) = FEVAL(L,0.1) = 1.950095004. FEVAL(PTOM(L,3) , 0.1) = 7.415958858 ~ $(1.950095004)^3$. FEVAL(REVS(L) , 1.950095004) = 0.0999999977. f(f(0.1)) = 1.02647988216. FEVAL(CFG(L,L) , 0.1) = 1.02647988195. CFG is useful for computing the Maclaurin series of $\sqrt{[1+f(z)]}$, for example.

VECTOR OPERATIONS MENU { VECTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
VECTOR TO LIST	**V→L**(V)	VECTOR	LIST
LIST TO VECTOR	**V←L**(L)	LIST	[VECTOR]
V→L([[1 2 3]]) = { 1 2 3 } V←L({ 1 2 3 }) = [1 2 3]			
TO VECTOR	**→VTR**(M)	VECTOR	[VECTOR]
→VTR([[1][2][3]]) = →VTR([[1 2 3]]) = →VTR([1 2 3]) = [1 2 3]			
SORT ↑	**SRT↑**(V)	[VECTOR]	[VECTOR]
↑ = increasing value with increasing position [1 2 3]			
SORT ↓	**SRT↓**(V)	[VECTOR]	[VECTOR]
SORT ↑ WITH SORTED INDEX	**SRTI**(V)	[VECTOR]	2: [VECTOR] 1: [INDEX]
SRT↑([2 5 3 7]) = [2 3 5 7] SRT↓([2 5 3 7]) = [7 5 3 2] SRTI([2 5 3 7]) = 2: [2 3 5 7] 1: [1 3 2 4] Suppose vector V originated as either a row or column in a matrix M. Index vector I is the required vector for sorting matrix M according to row (column) V by using **RORDR** or **CORDR**. See pages 233 and 234. *SIZE(V) ≥ 2 to be sorted.*			
VTR REVERSE	**VREV**(V)	[VECTOR]	[VECTOR]

VECTOR OPERATIONS MENU { VECTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
LIST REVERSE	**LREV**(L)	LIST	LIST
VREV([1 2 3 4 5]) = [5 4 3 2 1] LREV({ 1 2 3 4 5 }) = { 5 4 3 2 1 }			
INSERT VALUE	**IVAL**(V,X,L) V = VECTOR	X INSERTED AT LOCATION L	LARGER VECTOR WITH X AT L
IVAL([1 2 3 4 5] , 9 4) = [1 2 3 9 4 5]			
DELETE VALUE	**DVAL**(V,L) V = VECTOR	DELETE VALUE AT LOCATION L	SMALLER VECTOR WITHOUT X AT L
DVAL([1 2 3 9 4 5] , 4) = [1 2 3 4 5]			
SWAP VALUE	**SVAL**(V,L1,L2) V = VECTOR	SWAP VALUES AT L1 AND L2	
MOVE VALUE	**MVAL**(V,F,T) V = VECTOR	MOVE VALUE AT F TO T	
COPY VALUE	**CVAL**(V,F,T) V = VECTOR	COPY VALUE AT F INTO T	
SVAL([1 2 3 4 5] , 2 , 5) = [1 5 3 4 2] MVAL([1 2 3 4 5] , 2 , 5) = [1 3 4 5 2] CVAL([1 2 3 4 5] , 4 , 1) = [4 2 3 4 5]			

VECTOR OPERATIONS MENU { VECTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
VECTOR INSERT	**VINST**(V,VI,I) V = VECTOR	INSERT VI INTO V STARTING AT I. OUTPUT VECTOR IS LARGER	
VECTOR DELETE	**VDEL**(V,F,L) V = VECTOR	DELETE VALUES IN RANGE F TO L. OUTPUT VECTOR IS SMALLER	
VINST ([1 2 3 4 5] , [0 0 0] , 3) = [1 2 0 0 0 3 4 5] VDEL ([1 2 0 0 0 3 4 5] , 3 , 5) = [1 2 3 4 5]			
VECTOR SUBSET	**VSUBS**(V,F,L) V = VECTOR	EXTRACTS SUBSET OF V FROM F TO L	
VECTOR REPLACE	**VREPL**(V1,V2,F) V = VECTOR	REPLACES THE PART OF V1 WITH V2 STARTING AT POSITION F	
VSUBS([1 2 3 4 5] , 2 , 5) = [2 3 4 5] VREPL([1 2 3 4 5] , [8 9] , 2) = [1 8 9 4 5] Both **VSUBS** and **VREPL** also work for lists (symbolic vectors).			
VECTOR SPLIT	**VSPLT**(V,N)	V N	TWO VECTORS
Given vector V and 0 < N < size of V, **VSPLT** divides V into two vectors. N is the size of the first.			
VECTOR COMBINE	**VCMB**(V1,V2)	TWO VECTORS	ONE VECTOR
Given vectors V1 and V2, **VCMB** splices them into a single vector [V1 V2].			

VECTOR OPERATIONS MENU { VECTR }

FUNCTION	COMMAND	INPUTS	OUTPUTS
LIST ZERO DELETE	**LZDEL**(L)	LIST	LIST
Removes the contiguous zeros at the right of list L. For example: LZDEL({ 1 2 3 0 4 0 0 0 }) = { 1 2 3 0 4 }.			
LIST ZERO-FILL	**LZFIL**(N,D)	LISTS N and D	LISTS N and D
Adds zeros to the right of the smaller of the lists N and D so that they have the same size. **LZFIL({ 1 2 3 4 } , { 5 6 }) =** 2: { 1 2 3 4 } / 1: { 5 6 0 0 }			
VECTOR OF ONES	**ONE**(N)	N = SIZE	[VECTOR]
ONE creates a vector of all ones. ONE(4) = [1 1 1 1]			
UNIT IMPULSE	**UNITI**(N)	N = SIZE	[VECTOR]
UNITI creates a vector of size N whose first value is one and the rest are zero. UNITI(5) = [1 0 0 0 0]			
UP DIRECTORY	**UPDIR**	NONE	PARENT MENU

30

VECTOR SCALAR ALGEBRA

INTRODUCTION

This chapter presents the 30 vector scalar algebra commands in the VSAG menu. The VSAG commands generally treat vectors as a collection of ordered values and provide the common arithmetic functions for those values so that the user can apply the same operation to the data without having to put the individual elements on the stack. Maximum and minimum value commands are also provided along with random vector creation commands.

ARITHMETIC

Commands **VADD**, **VSUB**, **VECTX**, and **VECTD** provide the basic **+**, **−**, **×**, and **/** functions. **VSINV** provides the vector invert function **INV**. Multiplying or dividing all the elements by a constant is provided by the HP 48 **×** and **/** commands. **VECTX** is known as the Hadamard product.

SIMPLE FUNCTIONS

VLOG, **VALOG**, **VLN**, **VEXP**, **VSQ**, **VSRT**, **VABS**, **VARG**, **VMOD**, **VRND**, **VFLOR**, and **VCEIL** apply the associated HP 48 command to each element of the vector. **V1F1** and **V2F1** are available for applying other commands, such as **SIN**, to each element of a vector.

EXTREME VALUE

VMAX and **VMIN** find the maximum and minimum values of vectors.

ARGUMENT UNWRAP

ARGU and **VTRUR** provide argument unwrapping in radians for argument and complex vectors.

RANDOM VECTORS

RNDU and **RNDN** provide random vectors with uniform and normal amplitude statistics. **RNDC** provides two jointly normal vectors with a user-specified correlation coefficient.

LIMITATIONS OF SOFTWARE

Several of the programs in this menu like **VMAX** assume that the SIZE(V) ≥ 2.

VECTOR SCALAR ALGEBRA MENU { VSAG }

FUNCTION	COMMAND	INPUTS	OUTPUTS
VECTOR MAX	**VMAX**(V)	V = [VECTOR]	MAX VALUE
VECTOR MIN	**VMIN**(V)	V = [VECTOR]	MIN VALUE
VMAX([1 3 2 5 –1]) = 5 VMIN([1 3 2 5 –1]) = –1			
VECTOR SCALAR ADD	**VADD**(V,s)	V = [VECTOR] s = SCALAR	[VECTOR] $V_j + s$
VECTOR SCALAR SUBTRACT	**VSUB**(V,s)	V = [VECTOR] s = SCALAR	[VECTOR] $V_j - s$
VADD([1 3 2 5 –1] , 2) = [3 5 4 7 1] VSUB([1 3 2 5 –1] , 2) = [–1 1 0 3 –3]			
VECTOR LOG	**VLOG**(V)	V = [VECTOR]	LOG(V_j)
VECTOR ALOG	**VALOG**(V)	V = [VECTOR]	ALOG(V_j)
VLOG([3 5 4 7 1]) = [.4771 .699 .6021 .8451 0] VALOG([3 5 4 7 1]) = [1000 100000 10000 10000000 10]			
VECTOR LN	**VLN**(V)	V = [VECTOR]	LN(V_j)
VECTOR EXP	**VEXP**(V)	V = [VECTOR]	EXP(V_j)
VLN ([3 5 4 7 1]) = [1.0986 1.6094 1.3863 1.9459 0] VEXP([3 5 4 7 1]) = [20.09 148.41 54.6 1096.63 2.72]			

VECTOR SCALAR ALGEBRA MENU { VSAG }

FUNCTION	COMMAND	INPUTS	OUTPUTS
VECTOR SQ	**VSQ**(V)	V = [VECTOR]	$SQ(V_j)$
VECTOR √	**VSRT**(V)	V = [VECTOR]	$\sqrt{}(V_j)$
VSQ([3 5 4 7 1]) = [9 25 16 49 1] VSRT([3 5 4 7 1]) = [1.732 2.236 2 2.646 1]			
VECTOR MPY	**VECTX**(V1,V2)	V1 V2	$V1_j \times V2_j$
VECTOR DIVIDE	**VECTD**(V1,V2)	V1 V2	$V1_j \div V2_j$
VECTX([3 5 4 7 1] , [4 2 3 1 5]) = [12 10 12 7 5] VECTD([12 10 12 7 5] , { 3 5 4 7 1 }) = { 4 2 3 1 5 } V1 and V2 may be HP vectors or symbolic vectors (lists). If either V1 or V2 is a list, then the output vector is also a symbolic vector (list).			
VECTOR ABS	**VABS**(V)	V = [VECTOR]	$ABS(V_j)$
VECTOR ARG	**VARG**(V)	V = [VECTOR]	$ARG(V_j)$
The outputs of **VARG**, **ARGU**, and **VTRUR** are in radians. If degrees is desired, then use **PHASE**, **PHASU**, or **VTRUD** in the { PROC } menu. VABS([(1,2) (2,3) (3,4)] = [2.2361 3.6056 5] VABS([(1,2) (2,3) (3,4)] = [1.1071 .9828 .9273]			
VECTOR UNWRAP IN RADIANS	**ARGU**(V)	V = REAL [VECTOR]	UNWRAPPED ARGUMENT VECTOR

VECTOR SCALAR ALGEBRA MENU { VSAG }

FUNCTION	COMMAND	INPUTS	OUTPUTS
ARGU provides argument unwrapping where the input vector contains arguments in radians, such as the output of **VARG**. The values must be in the range $[-\pi, \pi]$. **VTRUR** uses **VARG** to compute the argument and then calls **ARGU** to do the unwrapping.			
VECTOR UNWRAP IN RADIANS	**VTRUR**(V)	V = COMPLEX [VECTOR]	UNWRAPPED ARGUMENT VECTOR
VTRUR([(1,0) (1,2) (2,–3) (3,–8) (–2,–5) (–1,5) (1,1)]) = [0 1.1071 –.9828 –1.212 –1.9513 –4.515 –5.4978]			
VECTOR INV	**VSINV**(V)	V = [VECTOR]	V_j INV
VSINV([(1,2) (0,0) (3,4)]) = [(.2,–.4) (0,0) (.12,–.16)] **VSINV** has an interesting feature. It inverts zeros by returning zeros. Also, VSINV({ }) = { }.			
VECTOR MOD	**VMOD**(V,n)	V = [VECTOR] $n \in \mathbb{N}$	MOD(V_j,n)
VECTOR ROUND	**VRND**(V,n)	V = [VECTOR] $n \in \mathbb{I}$	V n RND
VECTOR FLOOR	**VFLOR**(V)	V = [VECTOR]	FLOOR(V_j)
VECTOR CEILING	**VCEIL**(V)	V = [VECTOR]	CEIL(V_j)

VECTOR SCALAR ALGEBRA MENU { VSAG }

FUNCTION	COMMAND	INPUTS	OUTPUTS
See the HP 48 owner's manual for the commands MOD, RND, FLOOR, and CEIL.			
RANDOM VECTOR WITH UNIFORM AMPLITUDE STATISTICS	**RNDU**(μ,σ,s,N)	μ = MEAN σ = STAND DEV s = SEED N = SIZE	RANDOM [VECTOR]
RANDOM VECTOR WITH NORMAL AMPLITUDE STATISTICS	**RNDN**(μ,σ,s,N)	μ = MEAN σ = STAND DEV s = SEED N = SIZE	RANDOM [VECTOR]
RNDN(1,2,11,8) creates an eight dimensional vector with elements that are normally distributed with mean value of 1 and standard deviation of 2.			
RANDOM VECTOR WITH NORMAL AMPLITUDE STATISTICS	**RNDC**(μ,σ,ρ,s,N) s = SEED N = VECTOR SIZE	μ = COMPLEX MEAN σ = COMPLEX σ ρ = CORRELATION COEFFICIENT	RANDOM [VECTOR]
RNDC creates two normally distributed vectors and stores them as a complex vector. The real part of the mean and standard deviation is associated with the real part of the resulting vector. The correlation coefficient ρ is real, and $\rho \in [-1, 1]$. The command **C→R** will separate the vectors.			

VECTOR SCALAR ALGEBRA MENU { VSAG }

FUNCTION	COMMAND	INPUTS	OUTPUTS
RNDC((1,2) , (3,4) , .5 , 1 , 8) creates a complex, normally distributed random vector of size 8. The real part has mean 1 and standard deviation 3, while the imaginary part has mean 2 and standard deviation 4. The correlation between the real and imaginary parts is 0.5.			
EXPONENTIAL INDEX	**EINDX**(a,N)	$a \in \mathbb{C}$ $N \in \mathbb{N}$	[VECTOR]
EINDX creates an exponential index vector whose first element equals 1 and the others equal a^n for n = 1, 2, . . ., N – 1. EINDX(2 , 7) = [1 2 4 8 16 32 64]			
VECTOR CONSTANT DELETE	**VCDEL**(V,C)	V = [VECTOR] C = CONSTANT	[VECTOR]
VCDEL deletes all the elements equal to constant C in V and returns what is left. { } is returned if all the elements of V are equal to C, symbolizing an empty vector [ ]. VCDEL([1 2 0 4] , 0) = [1 2 4] VCDEL([1 1 1] , 1) = { }			

VECTOR SCALAR ALGEBRA MENU { VSAG }

FUNCTION	COMMAND	INPUTS	OUTPUTS
VECTOR ELEMENT OPERATIONS	**V1F1**(V,P)	V = [VECTOR] P = PROGRAM	V_j < P > EVAL
V1F1([1 2 3] , < COS >) = [COS(1) COS(2) COS(3)]			
VECTOR ELEMENT OPERATIONS	**V2F1**(V1,V2,P)	VECTORS V1 V2 P = PROGRAM	$V1_j$ $V2_j$ < P > EVAL
V2F1([1 2 3] , [4 5 6] , < ATN2 >) = [ATN2(1,4) ATN2(2,5) ATN2(3,6)] V1 and V2 may be HP vectors or symbolic vectors (lists). If either V1 or V2 is a list, then the output vector is also a symbolic vector (list). The elements of V1 and V2 may be symbolic.			
VECTOR OF ZEROS	**ZERO**(N)	N = SIZE	[VECTOR]
ZERO creates a vector of all zeros. ZERO(4) = [0 0 0 0]			
UP DIRECTORY	**UPDIR**	NONE	PARENT MENU

31

MATRIX SCALAR ALGEBRA

INTRODUCTION

This chapter presents the 32 matrix scalar algebra commands in the MSAG menu. The MSAG commands generally treat matrices as a collection of ordered values and provide the common arithmetic functions for those values so that the user can apply the same operation to the data without having to put the individual elements on the stack. Maximum and minimum value commands are also provided along with random matrix creation commands.

ARITHMETIC

Commands **MADD**, **MSUB**, **MATX**, and **MATD** provide the basic **+**, **−**, **×**, and **/** functions. Multiplying or dividing all the elements by a constant is provided by the HP 48 **×** and **/** commands. **MATX** is known as the Hadamard product.

SIMPLE FUNCTIONS

MLOG, **MALOG**, **MLN**, **MEXP**, **MSQ**, **MSRT**, **MABS**, **MARG**, **MMOD**, **MRND**, **MFLR**, and **MCEIL** apply the associated HP 48 command to each element of the vector. **M1F1** and **M2F1** are available for applying other commands, such as **SIN**, to each element of a matrix.

EXTREME VALUE

MMAX and **MMIN** find the maximum and minimum values of matrices.

MATRIX NORMS

RSUM, CSUM, RΣ0D, CΣ0D, RABS2, and **CABS2** provide various matrix norms.

RANDOM MATRICES

MRDU and **MRDN** provide random matrices with uniform and normal statistics. **MRDC** provides two jointly normal matrices with a user-specified correlation. **MRDC** can also be used to create random complex matrices. **MRDS** and **MRDH** create random symmetric and Hermitian matrices.

LIMITATIONS OF SOFTWARE

Several of the programs in this menu like **RSUM** assume that there are at least two columns. Similarly, **MMAX** assumes that M has at least two elements.

MATRIX SCALAR ALGEBRA
{ MSAG }

FUNCTION	COMMAND	INPUTS	OUTPUTS
MATRIX MAX	**MMAX**(M)	M = MATRIX	MAX VALUE
MATRIX MIN	**MMIN**(M)	M = MATRIX	MIN VALUE

$$\mathbf{MMAX}\left(\begin{bmatrix}1 & 8\\ 4 & 6\\ 2 & 9\end{bmatrix}\right) = 9 \qquad \mathbf{MMIN}\left(\begin{bmatrix}1 & 8\\ 4 & 6\\ 2 & 9\end{bmatrix}\right) = 1$$

MATRIX SCALAR ADD	**MADD**(M,s)	M = MATRIX s = SCALAR	$M_{jk} = M_{jk} + s$
MATRIX SCALAR SUBTRACT	**MSUB**(M,s)	M = MATRIX s = SCALAR	$M_{jk} = M_{jk} - s$

$$\mathbf{MADD}\left(\begin{bmatrix}1 & 8\\ 4 & 6\\ 2 & 9\end{bmatrix}, 2\right) = \begin{bmatrix}3 & 10\\ 6 & 8\\ 4 & 11\end{bmatrix} \qquad \mathbf{MSUB}\left(\begin{bmatrix}1 & 8\\ 4 & 6\\ 2 & 9\end{bmatrix}, 2\right) = \begin{bmatrix}-1 & 6\\ 2 & 4\\ 0 & 7\end{bmatrix}$$

MATRIX LOG	**MLOG**(M)	M = MATRIX	$LOG(M_{jk})$
MATRIX ALOG	**MALOG**(M)	M = MATRIX	$ALOG(M_{jk})$

$$\mathbf{MLOG}\left(\begin{bmatrix}1 & 8\\ 4 & 6\\ 2 & 9\end{bmatrix}\right) = \begin{bmatrix}0 & .903\\ .602 & .778\\ .301 & .954\end{bmatrix} \qquad \mathbf{MALOG}\left(\begin{bmatrix}1 & 3\\ 4 & 6\\ 2 & 4\end{bmatrix}\right) = \begin{bmatrix}10 & 1000\\ 10000 & 1000000\\ 100 & 10000\end{bmatrix}$$

MATRIX SCALAR ALGEBRA
{ MSAG }

FUNCTION	COMMAND	INPUTS	OUTPUTS
MATRIX LN	**MLN**(M)	M = MATRIX	LN(M_{jk})
MATRIX EXP	**MEXP**(M)	M = MATRIX	EXP(M_{jk})

$$\mathbf{MLN}\left(\begin{bmatrix}1 & 3\\ 4 & 6\\ 2 & 4\end{bmatrix}\right) = \begin{bmatrix}0 & 1.099\\ 1.386 & 1.792\\ .693 & 1.386\end{bmatrix} \qquad \mathbf{MEXP}\left(\begin{bmatrix}1 & 3\\ 4 & 6\\ 2 & 4\end{bmatrix}\right) = \begin{bmatrix}2.718 & 20.086\\ 54.598 & 403.429\\ 7.389 & 54.598\end{bmatrix}$$

MATRIX SQ	**MSQ**(M)	M = MATRIX	SQ(M_{jk})
MATRIX √	**MSRT**(M)	M = MATRIX	√(M_{jk})

$$\mathbf{MSQ}\left(\begin{bmatrix}1 & 3\\ 4 & 6\\ 2 & 4\end{bmatrix}\right) = \begin{bmatrix}1 & 9\\ 16 & 36\\ 4 & 16\end{bmatrix} \qquad \mathbf{MSRT}\left(\begin{bmatrix}1 & 9\\ 16 & 36\\ 4 & 16\end{bmatrix}\right) = \begin{bmatrix}1 & 3\\ 4 & 6\\ 2 & 4\end{bmatrix}$$

MATRIX MPY	**MATX**(M1,M2)	M1, M2 = MATRIX	$M1_{jk} \times M2_{jk}$
MATRIX DIVIDE	**MATD**(M1,M2)	M1, M2 = MATRIX	$M1_{jk} \div M2_{jk}$

$$\mathbf{MATX}\left(\begin{bmatrix}1 & 3\\ 4 & 6\\ 2 & 4\end{bmatrix}, \begin{bmatrix}4 & 2\\ 3 & 5\\ 1 & 6\end{bmatrix}\right) = \begin{bmatrix}4 & 6\\ 12 & 30\\ 2 & 24\end{bmatrix} \qquad \mathbf{MATD}\left(\begin{bmatrix}4 & 6\\ 12 & 30\\ 2 & 24\end{bmatrix}, \begin{bmatrix}4 & 2\\ 3 & 5\\ 1 & 6\end{bmatrix}\right) = \begin{bmatrix}1 & 3\\ 4 & 6\\ 2 & 4\end{bmatrix}$$

ROW SUM	**RSUM**(M)	M = MATRIX	[VECTOR]

MATRIX SCALAR ALGEBRA
{ MSAG }

FUNCTION	COMMAND	INPUTS	OUTPUTS
COLUMN SUM	**CSUM**(M)	M = MATRIX	[VECTOR]

$$\text{RSUM}\begin{bmatrix}1 & 3\\ 4 & 6\\ 2 & 4\end{bmatrix} = [4 \;\; 10 \;\; 6] \qquad \text{CSUM}\begin{bmatrix}1 & 3\\ 4 & 6\\ 2 & 4\end{bmatrix} = [7 \;\; 13]$$

ROW SUM PLUS DEL ROW IF Σ=0	**RΣ0D**(M)	M = MATRIX	2: [VECTOR] 1: M

Output is a vector of row sums and matrix M after removing the rows having a sum of zero.

$$\text{RΣ0D}\begin{bmatrix}1 & 0\\ 4 & -4\\ 2 & 4\end{bmatrix} = \begin{cases}2: [\,1 \;\; 6\,]\\ 1: \begin{bmatrix}1 & 0\\ 2 & 4\end{bmatrix}\end{cases} \qquad \text{CΣ0D}\begin{bmatrix}1 & 0\\ 4 & -4\\ 2 & 4\end{bmatrix} = \begin{cases}2: [\,7\,]\\ 1: \begin{bmatrix}1\\ 4\\ 2\end{bmatrix}\end{cases}$$

If all rows (columns) are deleted, an →ARRY error occurs.

COL SUM PLUS DEL ROW IF Σ=0	**CΣ0D**(M)	M = MATRIX	2: [VECTOR] 1: M
MATRIX ABS	**MABS**(M)	M = MATRIX	ABS(M_{jk})

MATRIX SCALAR ALGEBRA
{ MSAG }

FUNCTION	COMMAND	INPUTS	OUTPUTS
MABS $\begin{bmatrix} (1,2) & (2,3) \\ (3,4) & (4,5) \end{bmatrix} = \begin{bmatrix} 2.236 & 3.606 \\ 5 & 6.403 \end{bmatrix}$		**MARG** $\begin{bmatrix} (1,2) & (2,3) \\ (3,4) & (4,5) \end{bmatrix} = \begin{bmatrix} 1.107 & .983 \\ .927 & .896 \end{bmatrix}$	
MATRIX ARG	**MARG**(M)	M = MATRIX	ARG(M_{jk})
MATRIX MOD	**MMOD**(M,n)	M = MATRIX	MOD(M_{jk},n)
MATRIX ROUND	**MRND**(M,n)	M = MATRIX	M n RND
MATRIX FLOOR	**MFLR**(M)	M = MATRIX	FLOOR(M_{jk})
MATRIX CEILING	**MCEIL**(M)	M = MATRIX	CEIL(M_{jk})
See the HP 48 owner's manual for the commands **MOD**, **RND**, **FLOOR**, and **CEIL**.			
ROW NORM	**RABS2**(M)	MATRIX M	[VECTOR]
COLUMN NORM	**CABS2**(M)	MATRIX M	[VECTOR]
These sum the squares of the absolute values of the elements in each row (column) and return the values in a vector. This is the square of the Frobenius (Euclidean) norm of each row (column). **RABS2** and **CABS2** are like **RABS** and **CABS** except that the sum is not square-rooted.			
RANDOM MATRIX WITH UNIFORM AMPLITUDE STATISTICS	**MRDU**(μ,σ,s,L)	μ = MEAN σ = STAND DEV s = SEED L = { R C }	RANDOM R × C MATRIX

MATRIX SCALAR ALGEBRA { MSAG }

FUNCTION	COMMAND	INPUTS	OUTPUTS
RANDOM MATRIX WITH NORMAL AMPLITUDE STATISTICS	**MRDN**(μ,σ,s,L)	μ = MEAN σ = STAND DEV s = SEED L = { R C }	RANDOM R × C MATRIX
RANDOM MATRIX WITH NORMAL AMPLITUDE STATISTICS	**MRDC**(μ,σ,ρ,s,L) s = SEED L = {R C}	μ = COMPLEX MEAN σ = COMPLEX σ ρ = CORRELATION COEFFICIENT	RANDOM R × C MATRIX

MRDC creates two normally distributed matrices and stores them as a complex matrix. The real part of the mean and standard deviation is associated with the real part of the resulting matrix. The correlation coefficient ρ is real, and $\rho \in [-1, 1]$. The command **C→R** will separate the matrices.

MRDU, **MRDN**, and **MRDC** call **RNDU**, **RNDN**, and **RNDC**, respectively, to obtain the random values. Then **RDM** is used to create the matrix.

RANDOM SYMMETRIC MATRIX WITH NORMAL STATISTICS	**MRDS**(μ,σ,s,L)	μ = MEAN σ = STAND DEV s = SEED L = {R C} C = R	SQUARE RANDOM R × R MATRIX
RANDOM HERMITIAN MATRIX WITH NORMAL STATISTICS	**MRDH**(μ,σ,s,L)	μ = MEAN σ = STAND DEV s = SEED L = {R C} C = R	SQUARE RANDOM R × R MATRIX

MATRIX SCALAR ALGEBRA
{ MSAG }

FUNCTION	COMMAND	INPUTS	OUTPUTS
MRDS calls **MRDN** to obtain a random normal matrix M of mean μ and standard deviation $\sqrt{2}\times\sigma$. Then **MRDS** computes $(M + M^T)/2$. **MRDH** calls **MRDC** to obtain a random complex normal matrix M of mean μ, standard deviation $\sqrt{2}\times\sigma$, and correlation coefficient 0. Then **MRDH** computes $(M + M^H)/2$.			
MATRIX ELEMENT OPERATIONS	**M1F1**(M,P)	M = MATRIX P = PROGRAM	M_{jk} < P > EVAL
M1F1([[1 2][3 4]] , < COS >) = [[COS(1) COS(2)][COS(3) COS(4)]]			
MATRIX ELEMENT OPERATIONS	**M2F1**(M1,M2,P)	M1 M2 = MATRIX P = PROGRAM	M_{jk} M_{jk} < P > EVAL
M2F1([[1 2][3 4]] , [[5 6][7 8]] , < ATN2 >) = [[ATN2(1,5) ATN2(2,6)][ATN2(3,6) ATN2(4,8)]]			
UP DIRECTORY	**UPDIR**	NONE	PARENT MENU

APPENDIX A

AVAILABILITY OF OVER 700 APPLICATION, TEST, AND SYMBOLIC FUNCTION PROGRAMS

INTRODUCTION

There are over 500 MATHLIB application, test, and example programs that are available. They include the nontrivial examples in the manual. Also available are over 200 symbolic function and symbolic function defined derivative programs as described at the end of Chapter 19. This software is available on a ROM card from which you can transfer the source code programs to your VAR directory to edit and use in your particular applications. The 11 directories are each small enough that you do not need to buy a RAM card to store them. By distributing this software on a ROM card, you are saved the expense of buying a 128 K RAM card and the PC interface kit. This software comes with program summaries and directory menu maps. Since the 17 Weierstrass elliptic function programs are not covered in this manual, they are discussed in the application software manual. Additional application programs are currently under development. If you are interested in these programs, please write the author at the user support address on page 473 or call the author at (703) 938–0832.

ADVANCED NONPARAMETRIC TESTS

Three additional statistics nonparametric test programs are given below. They involve advanced MATHLIB programming and are out of the spirit of trying to keep the material in Chapter 22 palatable to beginning students. The examples and theory are discussed in Chapter 20 of Pfaffenberger and Patterson, which is referenced in Chapter 22 of this manual. Program RANK is given on page 295 of Chapter 22.

WILCOXON SIGNED RANK TEST

Given two data vectors of equal length, WILCX computes the r statistic for nonparametrically testing if the means are the same.

```
« → V1 V2 « V1 V2 - 0 VCDEL DUP SIZE EVAL SWAP DUP VABS SRTI
SWAP RANK ROT →COL ROT RORDR →VTR « SIGN » V1F1 VECTX
SSORT DUP 1 GET VΣ SWAP 3 GET VΣ NEG MIN » »
```

For example, if V1 = [200 150 300 400 120 250 320 175 100 500] and V2 = [225 375 650 380 100 250 410 180 130 60], then r = 5.

KRUSKAL–WALLIS TEST

The Kruskal–Wallis test is an extension of the Mann–Whitney test discussed in Chapter 22 where there are more than two populations. Each population is a column of input matrix M, and program KWT computes the t statistic for the test.

```
« → M « M SIZE DUP EVAL → S R C « M R C × 1 2 →LIST RDM
→VTR SRTI SWAP DUP RANK SWAP DUP SIZE EVAL SWAP 0 VCDEL
SIZE EVAL - VSUB →COL SWAP IORDR RORDR S RDM DUP CNUM DUP
ROT CSUM OVER DUP VΣ 1 + × 2 / - VSQ OVER VECTD SWAP VΣ DUP
1 + × / 12 × VΣ » » »
```

For M = [[25 18 26] [22 23 28] [31 21 24] [26 0 25] [20 24 32]], the t statistic equals 5.24642857144, which is more accurate than the 5.233 text answer.

SPEARMAN RANK CORRELATION TEST

Program SRHO is a more general version of **SRCTT** in that it allows repeated data values and applies midranking to them. For the **SRCTT** example given in Chapter 22, SRHO computes the same t statistic since rho = r.

```
« → V1 V2 « V1 DUP SIZE EVAL SWAP SRTI SWAP RANK →COL SWAP
IORDR RORDR →VTR V2 SRTI SWAP RANK →COL SWAP IORDR RORDR
→VTR - DUP DOT OVER 1 OVER SQ - × / 6 × 1 + » »
```

For V1 = [90 100 75 80 60 75 85 40 95 65] and V2 = [70 60 60 80 75 90 100 75 85 65], rho = .024242424244.

APPENDIX B

WARRANTY AND USER SUPPORT

USER SUPPORT

You can get answers to your questions concerning the operation of the MATHLIB commands from the author. If you do not find the information in this manual or the HP 48 owner's manual, contact the author at (703) 938-0832. I can provide technical assistance only about the operation of the MATHLIB commands. User support does not include consultation about your particular applications nor tutoring on your mathematics and engineering problems. For questions on the calculator, programming, or HP 48 commands, please call Hewlett-Packard at (503) 757-2004.

Numerous programs have been provided in this manual to demonstrate the use of the commands and programming techniques. You may type in these programs or obtain them from the author. Each owner of this book is granted a one-user individual license to use these programs in his or her applications. *However, the distribution of copies of these programs to others, by any means and in any form, is strictly prohibited.*

If you believe you have found a software bug which is not simply a lack of accuracy due to the computational limitations of the HP 48 or the noted limitations of the MATHLIB software, please write the author at:

Dr. John F. Holland
P.O. Box 3008
Oakton, VA 22124 USA

Please include a complete description of the bug and the related circumstances or application program it occurs in, plus your name, address, and phone number.

WARRANTY

Academic Press, Inc. ("AP") and the author cannot and do not warrant the performance or results that may be obtained by using this software and manual ("the product"). This includes the examples. The product is sold "as is" without warranty of any kind (except as hereafter described), either expressed or implied, including, but not limited to, any warranty of performance or any implied warranty of merchantability or fitness for any particular purpose. AP and the author shall not be responsible under any circumstances for providing information on the corrections to errors and omissions discovered in the product at any time. AP warrants only that the ROM card on which the software program is recorded is free from defects in material and faulty workmanship under normal use and service for a period of ninety (90) days from the date the product is delivered. The purchaser's sole and exclusive remedy in the event of a defect is expressly limited to either replacement of the ROM card or refund of the purchase price, at AP's sole discretion. This warranty does not apply if the product has been damaged by accident or misuse or as the result of service or modification. ROM card environmental limits are given on page 660 of the HP 48 owner's manual.

In no event, whether as a result of breach of contract, warranty or tort (including negligence), will AP or the author be liable to purchaser for any damages, including any lost profits, lost savings, or other incidental or consequential damages arising out of the use or inability to use the product or any modifications thereof, or due to the contents of the software programs, even if AP or the author has been advised of the possibility of such damages, or for any claim by any other party.

Any request for replacement of a defective ROM card must be postage prepaid and must be accompanied by the original defective ROM card, your mailing address and phone number, and proof of date of purchase and purchase price. Send such requests to the author at the above address. AP and the author shall have no obligation to refund the purchase price or to replace a ROM card based on claims of defects in the nature or operation of the product.

Some states do not allow limitation on how long an implied warranty lasts, nor exclusions or limitations of incidental or consequential damages, so the above limitations and exclusions may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary with jurisdiction.

The re-export of United States origin software is subject to the United States laws under the Export Administration Act of 1969 as amended. Any further sale of the product shall be in compliance with the United States Department of Commerce administration regulations. Compliance is your responsibility and not that of AP.

APPENDIX C

LIMITS, DERIVATIVES, AND FORMULAS

INTRODUCTION

This appendix presents some of the common relationships and formulas used in differential calculus. In addition, we review the concept of limits. This discussion is limited to real functions of a real variable denoted by $x \in \mathbf{R}$ and $f(x) \in \mathbf{R}$. Extensions to complex functions of a complex variable are given in Chapter 3.

DEFINITIONS

FUNCTION: A *function* f(x) is a relationship or mapping which defines the numbers f(x) in terms of the numbers x. The set of values for which x is defined is called the *domain* of f(x), and the corresponding set of values for which f(x) is defined is called the *range*. $f(x) = c$ where $c \in \mathbf{R}$ is the *constant* function, $f(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$ is a *polynomial* function,

$$f(x) = \frac{x + 2}{x\sqrt{x^2 + 1}}$$

is an *algebraic* function, and $f(x) = \sin(x)$ is a *transcendental* function. When the domain of f(x) is the integers $\mathbf{I}$ or the natural numbers $\mathbf{N}$, say $n \in \mathbf{N}$, then f(n) is called a *sequence*. $f(n) = (2n - 1)/n = 1, 3/2, 5/3, 7/4, 9/5, \ldots$, for $n \in \mathbf{N}$ is a sequence.

LIMITS: When a sequence for large n tends toward a single number, then that number is called the limit of the sequence. The limit of $f(n) = (2n - 1)/n$ as $n \to \infty$ is 2. The limit of $g(n) = (-1)^n$ as $n \to \infty$ does not exist. Define the functions $x(n) = (2n - 1)/n$ and $g(x) = x^2$. Then as $n \to \infty$, $x \to 2$. In this context we may define the limit of g(x) as x goes to 2 as $g(x) \to 4$ as $x \to 2$. Since the sequence $x(n) < 2$ for all $n \in \mathbf{N}$, we say it is a sequence from below or from the left. Observe that the sequence never takes on the value of its limit. The sequence $y(n) = (2n + 1)/n$ also has 2 as its limit, but $y(n) > 2$ for all $n \in \mathbf{N}$. This is an example of a limit from above or from the right. The statement that the limit of f(x) as $x \to a$ exists implies both the limit from the left and that from the right exist and are equal.

CONTINUITY: f(x) is said to be continuous at $x = x_0$ if

1) $f(x_0)$ is defined 2) $\lim_{x \to x_0} f(x)$ exists 3) $\lim_{x \to x_0} f(x) = f(x_0)$.

More rigorously for $h > 0$, f(x) is left continuous at x_0 if

1) $f(x_0)$ is defined 2) $\lim_{h \to 0} f(x_0 - h)$ exists 3) $\lim_{h \to 0} f(x_0 - h) = f(x_0)$.

and f(x) is right-continuous at x_0 if

1) $f(x_0)$ is defined 2) $\lim_{h \to 0} f(x_0 + h)$ exists 3) $\lim_{h \to 0} f(x_0 + h) = f(x_0)$.

When these limits are equal, then we say that f(x) is continuous at $x = x_0$.

NEIGHBORHOOD: For each real number c, we refer to an open interval containing c as a neighborhood. Thus, if $a < c < b$, then (a, b) is a neighborhood of c. A punctured or deleted neighborhood of c is the interval (a, b) with the point c removed.

DERIVATIVE: The derivative of f(x) from the left is the limit of the ratio

$$\frac{df(x)^-}{dx} = \lim_{h \to 0} \frac{f(x - h) - f(x)}{-h}$$

where $h > 0$. Similarly, the derivative from the right is the limit

$$\frac{df(x)^+}{dx} = \lim_{h \to 0} \frac{f(x + h) - f(x)}{h}.$$

When these limits are equal, then we say f(x) is differentiable at x.

$$\frac{df(x)}{dx} = \frac{df(x)^+}{dx} = \frac{df(x)^-}{dx} = f'(x)$$

If f(x) is differentiable at x, then f(x) must also be continuous at x. The geometric interpretation of f '(x) is as the slope of f(x). Higher-order derivatives are denoted by

$$\frac{d}{dx}\left[\frac{df(x)}{dx}\right] = \frac{d^2f(x)}{dx^2} = f''(x) = f^{(2)}(x), \qquad \frac{d^n(x)}{dx^n} = f^{(n)}(x).$$

LIMIT THEOREMS

The following relationships are available for evaluating limits. Let $c \in \mathbb{R}$, and

$$\lim_{x \to a} f(x) = A, \qquad \lim_{x \to a} g(x) = B.$$

Then if $h(x) = c$ = a constant for all x, $\lim_{x \to a} h(x) = c$

$$\lim_{x \to a}[c \cdot f(x)] = cA, \qquad \lim_{x \to a}[f(x) \pm g(x)] = A \pm B,$$

$$\lim_{x \to a}[f(x) \cdot g(x)] = A \cdot B, \qquad \lim_{x \to a} \frac{f(x)}{g(x)} = \frac{A}{B} \quad B \neq 0,$$

$$\lim_{x \to a} \sqrt[n]{f(x)} = \sqrt[n]{A} \qquad \text{for } \sqrt[n]{A} \in \mathbb{R}.$$

If g(x) is continuous at A, then

$$\lim_{x \to a} g(f(x)) = g(\lim_{x \to a} f(x)) = g(A).$$

L'HOSPITAL'S RULE

L'Hospital's rule handles the cases where the limit of the ratio of two functions cannot be evaluated as the ratio of the limits because they both go to either 0 or infinity. Let the functions f(x) and g(x) be differentiable at every number other than a in some deleted neighborhood of a with $g'(x) \neq 0$ in that neighborhood. If

$$\lim_{x \to a} f(x) = \lim_{x \to a} g(x) = 0,$$

or if

$$\lim_{x \to a} f(x) = \pm\infty \quad \text{AND} \quad \lim_{x \to a} g(x) = \pm\infty,$$

then

$$\lim_{x \to a} \frac{f(x)}{g(x)} = \lim_{x \to a} \frac{f'(x)}{g'(x)}.$$

The above two cases are the 0/0 and ∞/∞ cases. The $0 \cdot \infty$ case is handled by

$$\lim_{x \to a} f(x) \cdot g(x) = \lim_{x \to a} \frac{f(x)}{1/g(x)} = \lim_{x \to a} \frac{g(x)}{1/f(x)}$$

the $\infty - \infty$ case by

$$\lim_{x \to a} [1/f(x)] - [1/g(x)] = \lim_{x \to a} \frac{g(x) - f(x)}{g(x)\ f(x)}$$

and the 0^0, ∞^0, and 1^∞ cases by

$$\lim_{x \to a} f(x)^{g(x)} = \text{EXP}[\lim_{x \to a} g(x)\ \text{LN}\ f(x)].$$

With the 1E±499 range of the HP 48, it is very easy to evaluate limits numerically.

CHAIN RULE

The chain rule handles the cases of derivatives of functions of functions. Let f(y) and y(x) be differentiable functions. Then

$$\frac{d\ f(y)}{dx} = \frac{d\ f(y)}{dy} \bullet \frac{d\ y(x)}{dx}.$$

DERIVATIVE FORMULAS

Let $c \in \mathbb{R}$ but $c \neq 0$, and u(x), v(x), and w(x) be differentiable functions of x. Then

$$\frac{d}{dx}[c] = 0, \qquad \frac{d}{dx}[\,x^c\,] = cx^{c-1},$$

$$\frac{d}{dx}[x] = 1, \qquad \frac{d}{dx}[u^c(x)] = c\,u^{c-1}(x)\,\frac{d}{dx}[u(x)],$$

$$\frac{d}{dx}[cu(x)] = c \bullet \frac{d}{dx}u(x), \qquad \frac{d}{dx}[u(x)v(x)] = v(x) \bullet \frac{d}{dx}u(x) + u(x) \bullet \frac{d}{dx}v(x).$$

It is common not to explicitly show the argument x. Additional formulas are

$$\frac{d}{dx}[uvw] = uv\,\frac{dw}{dx} + uw\,\frac{dv}{dx} + vw\,\frac{du}{dx},$$

$$\frac{d}{dx}\left(\frac{u}{v}\right) = \frac{v\,\dfrac{du}{dx} - u\,\dfrac{dv}{dx}}{v^2} \qquad v \neq 0,$$

$$\frac{d}{dx}\left[\frac{c}{u}\right] = c \bullet \frac{d}{dx}\left[\frac{1}{u}\right] = c\,(-1)\,u^{-2} \bullet \frac{du}{dx} = -\frac{c}{u^2} \bullet \frac{du}{dx}.$$

ELEMENTARY TRANSCENDENTAL FUNCTIONS

The elementary transcendental functions are defined in Chapter 3. They are

- Exponential function
- Six trigonometric functions
- Six hyperbolic functions

Also defined in Chapter 3 are the inverses of these functions:

- Logarithmic function
- Six inverse trigonometric functions
- Six inverse hyperbolic functions

Derivatives of these functions are listed below.

$$\frac{d}{dx} e^x = e^x \qquad \frac{d}{dx} e^u = e^u \frac{du}{dx}$$

$$\frac{d}{dx} \ln x = \frac{1}{x} \qquad \frac{d}{dx} \ln u = \frac{1}{u}\frac{du}{dx}$$

$$\frac{d}{dx} \log_a(u) = \frac{1}{u} \log_a(e) \frac{du}{dx} \quad a>0 \text{ and } a\neq 1 \qquad \frac{d}{dx} a^u = a^u \ln(a) \frac{du}{dx} \quad a > 0$$

where the last two follow from the identities:

$$\log_a(u) = \frac{\ln(u)}{\ln(a)}, \qquad \log_a(e) = \frac{\ln(e)}{\ln(a)} = \frac{1}{\ln(a)} \qquad a^u = e^{u \ln(a)}.$$

$\ln(ab) = \ln a + \ln b \quad \ln(a/b) = \ln a - \ln b \quad \ln(a^b) = b \ln a \quad \ln e = 1 \quad e^{a+b} = e^a e^b$

$$\frac{d}{dx}[\sin u] = \cos u \frac{du}{dx} \qquad \frac{d}{dx}[\cot u] = -\csc^2 u \frac{du}{dx}$$

$$\frac{d}{dx}[\cos u] = -\sin u \frac{du}{dx} \qquad \frac{d}{dx}[\sec u] = \sec u \tan u \frac{du}{dx}$$

$$\frac{d}{dx}[\tan u] = \sec^2 u \frac{du}{dx} \qquad \frac{d}{dx}[\csc u] = -\csc u \cot u \frac{du}{dx}$$

$$\frac{d}{dx}[\arcsin u] = \frac{1}{\sqrt{1-u^2}}\frac{du}{dx} \qquad \frac{d}{dx}[\operatorname{arccot} u] = -\frac{1}{1+u^2}\frac{du}{dx}$$

$$\frac{d}{dx}[\arccos u] = -\frac{1}{\sqrt{1-u^2}}\frac{du}{dx} \qquad \frac{d}{dx}[\operatorname{arcsec} u] = \frac{1}{u\sqrt{u^2-1}}\frac{du}{dx}$$

$$\frac{d}{dx}[\arctan u] = \frac{1}{1+u^2}\frac{du}{dx} \qquad \frac{d}{dx}[\operatorname{arccsc} u] = -\frac{1}{u\sqrt{u^2-1}}\frac{du}{dx}$$

$$\frac{d}{dx}[\sinh u] = \cosh u\,\frac{du}{dx} \qquad \frac{d}{dx}[\coth u] = -\operatorname{csch}^2 u\,\frac{du}{dx}$$

$$\frac{d}{dx}[\cosh u] = \sinh u\,\frac{du}{dx} \qquad \frac{d}{dx}[\operatorname{sech} u] = -\operatorname{sech} u \tanh u\,\frac{du}{dx}$$

$$\frac{d}{dx}[\tanh u] = \operatorname{sech}^2 u\,\frac{du}{dx} \qquad \frac{d}{dx}[\operatorname{csch} u] = -\operatorname{csch} u \coth u\,\frac{du}{dx}$$

$$\frac{d}{dx}[\operatorname{arcsinh} u] = \frac{1}{\sqrt{1+u^2}}\frac{du}{dx} \qquad \frac{d}{dx}[\operatorname{arccoth} u] = \frac{1}{1-u^2}\frac{du}{dx}$$

$$\frac{d}{dx}[\operatorname{arccosh} u] = \frac{1}{\sqrt{u^2-1}}\frac{du}{dx} \qquad \frac{d}{dx}[\operatorname{arcsech} u] = \mp\frac{1}{u\sqrt{1-u^2}}\frac{du}{dx}$$

$$\frac{d}{dx}[\operatorname{arctanh} u] = \frac{1}{1-u^2}\frac{du}{dx} \qquad \frac{d}{dx}[\operatorname{arccsch} u] = \mp\frac{1}{u\sqrt{1+u^2}}\frac{du}{dx}$$

where the upper sign of $\mp$ is used for RE(u) > 0, and the lower sign is used for RE(u) < 0. All of these formulas are also valid for the general complex case.

$$\frac{d^n}{dx^n}\sin x = \sin\left(x + \frac{n\pi}{2}\right) \qquad \frac{d^n}{dx^n}\cos x = \cos\left(x + \frac{n\pi}{2}\right)$$

MULTIVARIATE CALCULUS

Differentiation for functions of several variables is the natural extension of that for a single variable. We simply define the limits one variable at a time and use the symbol ∂ instead of d to point out that it is a *partial derivative*, not a total derivative as defined below. Assuming that the right and left derivatives with respect to each variable are equal, for two variables the partial derivative limits are

$$\frac{\partial f(x, y)}{\partial x} = \lim_{h \to 0} \frac{f(x + h, y) - f(x, y)}{h},$$

$$\frac{\partial f(x, y)}{\partial y} = \lim_{h \to 0} \frac{f(x, y + h) - f(x, y)}{h}.$$

Now suppose w = f(x, y, z, t) where x, y, and z are also functions of time t. Then the total derivative of w with respect to t is

$$\frac{dw}{dt} = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt} + \frac{\partial f}{\partial z}\frac{dz}{dt} + \frac{\partial f}{\partial t}.$$

This can also be written in terms of the differentials dx, dy, dz, and dt.

$$dw = \frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial y}dy + \frac{\partial f}{\partial z}dz + \frac{\partial f}{\partial t}dt$$

where we have simply multiplied both sides of the equation by dt and treated the differentials by the usual rules of algebra.

VECTOR AND MATRIX CALCULUS

Definitions for vectors and matrices are given in Chapter 20. Commands for symbolic vectors and matrices (arrays) are given in Chapter 25. In general, the derivative of an array is simply the derivative of each element using the above definitions and rules. Command **SDERV** provides this. However, there are several special vector derivative operations which are useful in solving various problems in physics.

Define the vector operator ∇ by the equation

$$\nabla = \frac{\partial}{\partial x}\mathbf{i} + \frac{\partial}{\partial y}\mathbf{j} + \frac{\partial}{\partial z}\mathbf{k}$$

where **i**, **j**, and **k** denote the unit vectors in the standard rectangular coordinate system. Also define the three-dimensional vector differential $\mathbf{dr} = \mathbf{i}\,dx + \mathbf{j}\,dy + \mathbf{k}\,dz$. We can now very compactly represent the above total differentials by

$$dw = [dr \bullet \nabla]\, w + \frac{\partial w}{\partial t} dt \qquad \frac{dw}{dt} = \left[\frac{dr}{dt} \bullet \nabla\right] w + \frac{\partial w}{\partial t}$$

where the symbol • denotes dot product. If you see Maxwell's equations in their original form before the *nabla* notation ∇, you will quickly appreciate the notation.

GRADIENT

When the ∇ operator is applied to a scalar function ϕ, the result is called the *gradient* of ϕ. In general, a gradient can be N-dimensional and is provided by command **SGRD**.

$$\nabla \phi = \frac{\partial \phi}{\partial x} \mathrm{i} + \frac{\partial \phi}{\partial y} \mathrm{j} + \frac{\partial \phi}{\partial z} \mathrm{k}$$

$\nabla\phi \bullet \mathbf{a}$, where **a** is any unit vector, is called the *directional derivative* of ϕ in the direction **a**.

DIVERGENCE

The *divergence* is in general the dot product of ∇ with a three-dimensional vector $\mathbf{V}(x, y, z) = V_x \mathbf{i} + V_y \mathbf{j} + V_z \mathbf{k}$.

$$\nabla \bullet \mathbf{V} = \frac{\partial V_x}{\partial x} + \frac{\partial V_y}{\partial y} + \frac{\partial V_z}{\partial z}$$

By convention, ∇ operates on that to the right so that $\nabla \bullet V \neq V \bullet \nabla$. The command **SDIV** provides the divergence operation.

CURL

The *curl* or *rotation* of $\mathbf{V}(x, y, z) = V_x \mathbf{i} + V_y \mathbf{j} + V_z \mathbf{k}$ is the cross-product $\nabla \times V$.

$$\nabla \times \mathbf{V} = \left(\frac{\partial V_z}{\partial y} - \frac{\partial V_y}{\partial z}\right) \mathbf{i} + \left(\frac{\partial V_x}{\partial z} - \frac{\partial V_z}{\partial x}\right) \mathbf{j} + \left(\frac{\partial V_y}{\partial x} - \frac{\partial V_x}{\partial y}\right) \mathbf{k}$$

The command **SCURL** provides the curl operation.

LAPLACIAN

The *Laplacian* is the composition of the operations $\nabla \cdot (\nabla \phi) = \nabla^2 \phi$ and equals

$$\nabla^2 = \nabla \cdot \nabla = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}.$$

FORMULAS INVOLVING ∇

If $\mathbf{F}$ and $\mathbf{G}$ are differentiable vectors, Φ and Ψ are differentiable functions, and $\alpha \in \mathbb{C}$:

$$(\mathbf{G} \cdot \nabla)\mathbf{F} = G_x \frac{\partial \mathbf{F}}{\partial x} + G_y \frac{\partial \mathbf{F}}{\partial y} + G_z \frac{\partial \mathbf{F}}{\partial z} = \mathbf{i}\,(\mathbf{G} \cdot \nabla F_x) + \mathbf{j}\,(\mathbf{G} \cdot \nabla F_y) + \mathbf{k}\,(\mathbf{G} \cdot \nabla F_z)$$

$$\nabla(\Phi + \Psi) = \nabla\Phi + \nabla\psi$$
$$\nabla \cdot (\mathbf{F} + \mathbf{G}) = \nabla \cdot \mathbf{F} + \nabla \cdot \mathbf{G}$$
$$\nabla \times (\mathbf{F} + \mathbf{G}) = \nabla \times \mathbf{F} + \nabla \times \mathbf{G}$$
$$\nabla(\nabla \cdot \mathbf{F}) = \nabla^2 \mathbf{F} + \nabla \times (\nabla \times \mathbf{F})$$
$$\nabla \times (\nabla \times \mathbf{F}) = \nabla(\nabla \cdot \mathbf{F}) - \nabla^2 \mathbf{F}$$

$$\nabla(\alpha\Phi) = \alpha\,\nabla\Phi$$
$$\nabla \cdot (\alpha\mathbf{F}) = \alpha\,\nabla \cdot \mathbf{F}$$
$$\nabla \times (\alpha\mathbf{F}) = \alpha\,\nabla \times \mathbf{F}$$
$$\nabla \times (\nabla\Phi) = 0$$
$$\nabla \cdot (\nabla \times \mathbf{F}) = 0$$

$$\nabla(\Phi\Psi) = \Psi\,\nabla\,\Phi + \Phi\,\nabla\,\Psi$$
$$\nabla(\mathbf{F} \cdot \mathbf{G}) = (\mathbf{F} \cdot \nabla)\mathbf{G} + (\mathbf{G} \cdot \nabla)\mathbf{F} + \mathbf{F} \times (\nabla \times \mathbf{G}) + \mathbf{G} \times (\nabla \times \mathbf{F})$$
$$\nabla \cdot (\Phi\,\mathbf{F}) = \Phi\,(\nabla \cdot \mathbf{F}) + (\nabla\Phi) \cdot \mathbf{F}$$
$$\nabla \cdot (\mathbf{F} \times \mathbf{G}) = \mathbf{G} \cdot (\nabla \times \mathbf{F}) - \mathbf{F} \cdot (\nabla \times \mathbf{G})$$
$$(\mathbf{G} \cdot \nabla)\Phi\mathbf{F} = \mathbf{F}(\mathbf{G} \cdot \nabla\Phi) + \Phi(\mathbf{G} \cdot \nabla)\mathbf{F}$$
$$\nabla \times (\Phi\mathbf{F}) = \Phi\,(\nabla \times \mathbf{F}) + (\nabla\Phi) \times \mathbf{F}$$
$$\nabla \times (\mathbf{F} \times \mathbf{G}) = (\mathbf{G} \cdot \nabla)\mathbf{F} - (\mathbf{F} \cdot \nabla)\mathbf{G} + \mathbf{F}\,(\nabla \cdot \mathbf{G}) - \mathbf{G}(\nabla \cdot \mathbf{F})$$
$$(\mathbf{G} \cdot \nabla)\mathbf{F} = \tfrac{1}{2}[\nabla \times (\mathbf{F} \times \mathbf{G}) + \nabla(\mathbf{F} \cdot \mathbf{G}) - \mathbf{F}(\nabla \cdot \mathbf{G}) + \mathbf{G}(\nabla \cdot \mathbf{F}) - \mathbf{F} \times (\nabla \times \mathbf{G}) - \mathbf{G} \times (\nabla \times \mathbf{F})]$$

Also note that if $\mathbf{r} = x\,\mathbf{i} + y\,\mathbf{j} + z\,\mathbf{k}$, then $[\,\mathbf{F}(\mathbf{r}) \cdot \nabla]\mathbf{r} = \mathbf{F}(\mathbf{r})$.

With MATHLIB all these operations can be performed on the HP 48 in eight orthogonal curvilinear coordinate systems.

APPENDIX D

INTEGRAL CALCULUS AND TABLES

INTRODUCTION

This appendix briefly discusses the relationships and formulas used in integration. In addition, we briefly survey several types of integrals. The discussion is limited to real functions of a real variable. Extensions to complex functions of a complex variable are discussed in Chapter 3 and defined for these tables at the end of this appendix.

The HP 48 does not have the vast megabytes of memory required to hold large numbers of integral formulas. Consequently, we provide the below collection. An example of advanced techniques for working with integral formulas is given on page 212.

SUMMATION NOTATION

The symbol Σ is commonly used to denote summation. In the equation

$$a_0 + a_1x + a_2x^2 + \ldots + a_{N-1}x^{N-1} + a_nx^N = \sum_{n=0}^{N} a_n\, x^n,$$

n is the summation index, 0 is the lower limit, and N is the upper limit. In general, both summation limits can be infinite. Finite summations are implemented on the HP 48 by the command Σ.

RIEMANN OR COMMON INTEGRATION

A Riemann integral is the limit of a special sum which converges to the area under the function. Let f(x) be defined on interval [a, b] and x_n be numbers such that: $x_0 = a$, $x_n = b$, and $x_0 < x_1 < x_2 < \ldots < x_{n-1} < x_n$. Let the numbers $z_n \in [x_{n-1}, x_n]$ for n = 1, 2, . . . n. In particular, choose $x_n - x_{n-1} = 1/N$ for positive integer N and define $\Delta x = (b - a)/N$. Then the sum

$$\sum_{n=1}^{N} f(z_n)\ \Delta x$$

is with increasing N a better and better rectangular approximation to the area under f(x) over the range (a, b). The limit as $N \to \infty$ is in fact the exact area under f(x) which defines the Riemann or common integral.

$$\lim_{N \to \infty} \sum_{n=1}^{N} f(z_n)\ \Delta x = \int_a^b f(x)\ dx = F(b) - F(a)$$

When the integration limits are indefinite, we use the notation

$$F(x) = \int f(x)\ dx.$$

Given the definition, one simply applies the formulas to perform integration.

GENERAL INTEGRATION FORMULAS

Let f(z) and g(z) be functions and a and b be constants. Then we have

LINEARITY: $\int [a\ f(z) + b\ g(z)]\ dz = a \int f(z)\ dz + b \int g(z)\ dz$.

ANTIDERIVATIVE: Integration is the inverse of differentiation.

$$\frac{d}{dz} \int f(z)\ dz = f(z) \quad \text{and} \quad \int f'(z)\ dz = f(z).$$

INTEGRATION BY PARTS: $\int f'(z)\ g(z)\ dz = f(z)\ g(z) - \int f(z)\ g'(z)\ dz$.

CHANGE OF VARIABLE: $\int f(z)\ dz = \int f[g(w)]\ g'(w)\ dw$ where $z = g(w)$.

CHAIN RULE: $\int [g(z)]^n g'(z)\, dz = [g(z)]^{n+1}/(n+1)$ for $n \neq -1$, but $\int [g(z)]^{-1} g'(z)\, dz = \ln [g(z)]$.

OTHER INTEGRAL DEFINITIONS

The above integrals use the differential notation dz. This is not unique. As with differential calculus we can also write

$$F(z) = \int \frac{d\, F(z)}{dz}\, dz = \int d\, F(z).$$

and the change of variable formula as

$$F(z) = \int f[g(z)]\, g'(z)\, dz = \int f[g(z)]\, dg(z).$$

These are commonly called Stieltjes integrals. The notion of dF(z) and dg(z) can be made quite abstract where now we use the notation $d\mu(z)$ and we call $\mu(z)$ the measure defined on some abstract space. These abstract integrals are generally called Lebesgue integrals. Lebesgue-integrable in the sense of ordinary Riemann-like integrals means that the integrand is absolutely integrable, that is, f(z) is absolutely integrable in the interval (a, b) if the integral $\int_a^b |f(z)|\, dz < \infty$. Absolute integrability (summability) justifies limit interchanges. The difficult part of calculus lies in the understanding of in what sense the limits exist and in what sense they may be interchanged.

BASIC FORMULAS

$$\int \frac{dz}{z} = \ln z \qquad \int z^n\, dz = \frac{z^{n+1}}{n+1} \quad n \neq -1$$

$$\int e^{az}\, dz = e^{az}/a \qquad \int b^{az}\, dz = \frac{b^{az}}{a \ln b}$$

$$\int \ln z\, dz = z \ln z - z \qquad \int a^z \ln a\, dz = a^z$$

$$\int [f(z)]^n f'(z)\,dz = \frac{[f(z)]^{n+1}}{n+1} \quad n \neq -1$$

$$\int [a\,f(z)+b]^n f'(z)\,dz = \frac{[a\,f(z)+b]^{n+1}}{a\,(n+1)}$$

$$\int \frac{f'(z)\,dz}{f(z)} = \ln f(z)$$

$$\int \frac{f'(z)\,dz}{\sqrt{a\,f(z)+b}} = \frac{2}{a}\sqrt{a\,f(z)+b}$$

$$\int \sin z\,dz = -\cos z$$

$$\int \cos z\,dz = \sin z$$

$$\int \frac{dz}{\sin^2 z} = -\cot z$$

$$\int \frac{dz}{\cos^2 z} = \tan z$$

$$\int \frac{\sin z}{\cos^2 z}dz = \sec z$$

$$\int \frac{\cos z}{\sin^2 z}dz = -\csc z$$

$$\int \tan z\,dz = -\ln \cos z$$

$$\int \cot z\,dz = \ln \sin z$$

$$\int \frac{dz}{a^2+z^2} = \frac{1}{a}\arctan\frac{z}{a} = -\frac{1}{a}\operatorname{arccot}\frac{z}{a}$$

$$\int \frac{dz}{a^2-z^2} = \frac{1}{a}\operatorname{arctanh}\frac{z}{a} = -\frac{1}{a}\operatorname{arccoth}\frac{z}{a}$$

$$\int \frac{dz}{\sqrt{z^2 \pm a^2}} = \ln\left(z + \sqrt{z^2 \pm a^2}\right)$$

$$\int \frac{dz}{\sqrt{a^2-z^2}} = \arcsin\frac{z}{|a|} = -\arccos\frac{z}{|a|}$$

$$\int \frac{dz}{\sqrt{z^2+1}} = \operatorname{arcsinh} z$$

$$\int \frac{dz}{\sqrt{z^2-1}} = \operatorname{arccosh} z$$

$$\int \sinh z\, dz = \cosh z \qquad \int \cosh z\, dz = \sinh z$$

$$\int \frac{dz}{\sinh^2 z} = -\coth z \qquad \int \frac{dz}{\cosh^2 z} = \tanh z$$

$$\int \tanh z\, dz = \ln \cosh z \qquad \int \coth z\, dz = \ln \sinh z$$

$$\int \frac{dz}{\sinh z} = \ln \tanh \frac{z}{2} \qquad \int_{z_1}^{z_2} f(z)\, dz = \int_{-z_2}^{-z_1} f(-z)\, dz$$

RATIONAL FUNCTION INTEGRATION

A *rational function* is the ratio of two polynomials with no common roots. To integrate an arbitrary rational function f(z)/g(z), the polynomials must first be divided into the form $f(z)/g(z) = f_1(z) + f_2(z)/g(z)$, where $f_1(z)$ is a polynomial in z and the degree of $f_2(z)$ is less than that of g(z). The command **PDVD** will do this division for you. The HP 48 command ∫ will then integrate $f_1(z)$ directly. Next use **AROOT** or **PROOT** to find the roots of g(z). Given these roots, use **HVSDE** to perform a partial fraction expansion of $f_2(z)/g(z)$. The resulting terms can then be integrated using these tables.

$$\int (a + bz)^n\, dz = \frac{(a + bz)^{n+1}}{(n+1)b} \qquad n \neq -1$$

$$\int z(a + bz)^n\, dz = \frac{(a + bz)^{n+2}}{b^2 (n+2)} - \frac{a(a + bz)^{n+1}}{b^2 (n+1)} \qquad n \neq -1, -2$$

$$\int z^2 (a + bz)^n\, dz = \frac{1}{b^3}\left[\frac{(a + bz)^{n+3}}{n+3} - 2a\,\frac{(a + bz)^{n+2}}{n+2} + a^2\,\frac{(a + bz)^{n+1}}{n+1}\right]$$

$$n \neq -1, -2, -3$$

$$\int \frac{dz}{(a + bz)} = \frac{\ln(a + bz)}{b}$$

$$\int \frac{dz}{(a + bz)^3} = -\frac{1}{2b(a + bz)^2}$$

$$\int \frac{dz}{(a + bz)^2} = -\frac{1}{b(a + bz)}$$

$$\int \frac{z\,dz}{a + bz} = \frac{a + bz - a\ln(a + bz)}{b^2}$$

$$\int \frac{z\,dz}{(a + bz)^2} = \frac{1}{b^2}\left[\ln(a + bz) + \frac{a}{a + bz}\right]$$

$$\int \frac{z\,dz}{(a + bz)^n} = \frac{1}{b^2}\left[\frac{-1}{(n - 2)(a + bz)^{n-2}} + \frac{a}{(n - 1)(a + bz)^{n-1}}\right] \quad n \neq 1, 2$$

$$\int \frac{z^2\,dz}{a + bz} = \frac{1}{b^3}\left[\frac{1}{2}(a + bz)^2 - 2a(a + bz) + a^2\ln(a + bz)\right]$$

$$\int \frac{z^2\,dz}{(a + bz)^2} = \frac{1}{b^3}\left[a + bz - 2a\ln(a + bz) - \frac{a^2}{a + bz}\right]$$

$$\int \frac{z^2\,dz}{(a + bz)^3} = \frac{1}{b^3}\left[\ln(a + bz) + \frac{2a}{a + bz} - \frac{a^2}{2(a + bz)^2}\right]$$

$$\int \frac{z^2\,dz}{(a + bz)^n} = \frac{1}{b^3}\left[\frac{-1}{(n - 3)(a + bz)^{n-3}} + \frac{2a}{(n - 2)(a + bz)^{n-2}} - \frac{a^2}{(n - 1)(a + bz)^{n-1}}\right] \quad n \neq 1, 2, 3$$

$$\int \frac{dz}{z(a + bz)} = -\frac{1}{a} \ln\left(\frac{a + bz}{z}\right)$$

$$\int \frac{dz}{z(a + bz)^2} = \frac{1}{a(a + bz)} - \frac{1}{a^2} \ln\left(\frac{a + bz}{z}\right)$$

$$\int \frac{dz}{z(a + bz)^3} = \frac{1}{a^3}\left[\frac{1}{2}\left(\frac{2a + bz}{a + bz}\right)^2 + \ln\left(\frac{z}{a + bz}\right)\right]$$

$$\int \frac{dz}{z^2(a + bz)} = -\frac{1}{az} + \frac{b}{a^2} \ln\left(\frac{a + bz}{z}\right)$$

$$\int \frac{dz}{z^3(a + bz)} = \frac{2bz - a}{2a^2x^2} + \frac{b^2}{a^3} \ln\left(\frac{z}{a + bz}\right)$$

$$\int \frac{dz}{z^2(a + bz)^2} = -\frac{a + 2bz}{a^2z(a + bz)} + \frac{2b}{a^3} \ln\left(\frac{a + bz}{z}\right)$$

FORMS CONTAINING $c^2 \pm z^2$ OR $z^2 - c^2$

$$\int \frac{dz}{c^2 + z^2} = \frac{1}{c} \arctan \frac{z}{c}$$

$$\int \frac{z\,dz}{c^2 \pm z^2} = \pm\frac{1}{2} \ln(c^2 \pm z^2)$$

$$\int \frac{dz}{c^2 - z^2} = \frac{1}{2c} \ln\left(\frac{c + z}{c - z}\right) \quad c^2 > z^2$$

$$\int \frac{z\,dz}{(c^2 \pm z^2)^{n+1}} = \mp\frac{1}{2n(c^2 \pm z^2)^n}$$

$$\int \frac{dz}{z^2 - c^2} = \frac{1}{2c} \ln\left(\frac{z - c}{z + c}\right) \quad z^2 > c^2$$

$$\int \frac{dz}{(c^2 \pm z^2)^n} = \frac{1}{2c^2(n-1)}\left[\frac{z}{(c^2 \pm z^2)^{n-1}} + (2n-3)\int \frac{dz}{(c^2 \pm z^2)^{n-1}}\right]$$

$$\int \frac{dz}{(z^2 - c^2)^n} = \frac{1}{2c^2(n-1)}\left[-\frac{z}{(z^2 - c^2)^{n-1}} - (2n-3)\int \frac{dz}{(z^2 - c^2)^{n-1}}\right]$$

$$\int \frac{z\,dz}{z^2 - c^2} = \frac{1}{2}\ln(z^2 - c^2) \qquad \int \frac{z\,dz}{(z^2 - c^2)^{n+1}} = \frac{-1}{2n(z^2 - c^2)^n} \qquad n \neq 0$$

FORMS CONTAINING a + bz AND c + ez

$$\int \frac{dz}{(a + bz)(c + ez)} = \frac{1}{ae - bc}\ln\left(\frac{c + ez}{a + bz}\right)$$

$$\int \frac{z\,dz}{(a + bz)(c + ez)} = \frac{1}{ae - bc}\left[\frac{a}{b}\ln(a + bz) - \frac{c}{e}\ln(c + ez)\right]$$

$$\int \frac{dz}{(a + bz)^2(c + ez)} = \frac{1}{ae - bc}\left(\frac{1}{a + bz} + \frac{e}{ae - bc}\ln\frac{c + ez}{a + bz}\right)$$

$$\int \frac{z\,dz}{(a + bz)^2(c + ez)} = \frac{-a}{b(ae - bc)(a + bz)} - \frac{c}{(ae - bc)^2}\ln\frac{c + ez}{a + bz}$$

$$\int \frac{z^2\,dz}{(a + bz)^2(c + ez)} = \frac{a^2}{b^2(ae - bc)(a + bz)} + \frac{1}{(ae - bc)^2}\left[\frac{c^2}{e}\ln(c+ez) + \frac{a(ae - 2bc)}{b^2}\ln(a + bz)\right]$$

$$\int \frac{a + bz}{c + ez}\,dz = \frac{bz}{e} + \frac{ae - bc}{e^2}\ln(c + ez)$$

FORMS CONTAINING $a + bz^n$

$$\int \frac{dz}{a + bz^2} = \frac{1}{\sqrt{ab}}\arctan\frac{z\sqrt{ab}}{a} \qquad ab > 0$$

$$\int \frac{dz}{a + bz^2} = \frac{1}{2\sqrt{-ab}}\ln\frac{a + z\sqrt{-ab}}{a - z\sqrt{-ab}} = \frac{1}{\sqrt{-ab}}\operatorname{arctanh}\frac{z\sqrt{-ab}}{a} \qquad ab < 0$$

$$\int \frac{dz}{a^2 + b^2z^2} = \frac{1}{ab}\arctan\frac{bz}{a}$$

$$\int \frac{z^2\,dz}{a + bz^2} = \frac{z}{b} - \frac{a}{b}\int \frac{dz}{a + bz^2}$$

$$\int \frac{z\,dz}{a + bz^2} = \frac{1}{2b}\ln(a + bz^2)$$

$$\int \frac{dz}{a^2 - b^2z^2} = \frac{1}{2ab}\ln\frac{a + bz}{a - bz}$$

$$\int \frac{dz}{(a + bz^2)^2} = \frac{z}{2a(a + bz^2)} + \frac{1}{2a}\int \frac{dz}{a + bz^2}$$

$$\int \frac{dz}{(a + bz^2)^{m+1}} = \frac{z}{2ma(a + bz^2)^m} + \frac{2m - 1}{2ma}\int \frac{dz}{(a + bz^2)^m}$$

$$\int \frac{z\,dz}{(a + bz^2)^{m+1}} = -\frac{1}{2bm(a + bz^2)^m}$$

$$\int \frac{z^2\,dz}{(a + bz^2)^{m+1}} = \frac{-z}{2mb(a + bz^2)^m} + \frac{1}{2mb}\int \frac{dz}{(a + bz^2)^m}$$

$$\int \frac{dz}{z(a + bz^2)} = \frac{1}{2a}\ln\frac{z^2}{a + bz^2} \qquad \int \frac{dz}{z^2(a + bz^2)} = -\frac{1}{az} - \frac{b}{a}\int \frac{dz}{a + bz^2}$$

$$\int \frac{dz}{a + bz^3} = \frac{\sqrt[3]{a/b}}{3a}\left[\frac{1}{2}\ln\frac{(\sqrt[3]{a/b} + z)^3}{a + bz^3} + \sqrt{3}\arctan\frac{2z - \sqrt[3]{a/b}}{\sqrt{3}\sqrt[3]{a/b}}\right]$$

$$\int \frac{z\,dz}{a + bz^3} = \frac{1}{3b\sqrt[3]{a/b}}\left[\frac{1}{2}\ln\frac{a + bz^3}{(\sqrt[3]{a/b} + z)^3} + \sqrt{3}\arctan\frac{2z - \sqrt[3]{a/b}}{\sqrt{3}\sqrt[3]{a/b}}\right]$$

$$\int \frac{z^2\,dz}{a + bz^3} = \frac{1}{3b}\ln(a + bz^3)$$

$$\int \frac{dz}{a + bz^4} = \frac{\sqrt[4]{a/(4b)}}{2a}\left[\frac{1}{2}\ln\frac{z^2 + 2z\sqrt[4]{a/(4b)} + 2\sqrt{a/(4b)}}{z^2 - 2z\sqrt[4]{a/(4b)} + 2\sqrt{a/(4b)}} + \arctan\frac{2z\sqrt[4]{a/(4b)}}{2\sqrt{a/(4b)} - z^2}\right] \qquad ab>0$$

$$\int \frac{dz}{a + bz^4} = \frac{\sqrt[4]{-a/b}}{2a}\left[\frac{1}{2}\ln\frac{z + \sqrt[4]{-a/b}}{z - \sqrt[4]{-a/b}} + \arctan\frac{z}{\sqrt[4]{-a/b}}\right] \qquad ab < 0$$

$$\int \frac{z\,dz}{a + bz^4} = \frac{1}{2b\sqrt{a/b}}\arctan\frac{z^2}{\sqrt{a/b}} \qquad ab > 0$$

$$\int \frac{z\,dz}{a + bz^4} = \frac{1}{4b\sqrt{-a/b}} \ln \frac{z^2 - \sqrt{-a/b}}{z^2 + \sqrt{-a/b}} \qquad ab < 0$$

$$\int \frac{z^2\,dz}{a + bz^4} = \frac{1}{4b\sqrt[4]{a/(4b)}} \left[\frac{1}{2} \ln \frac{z^2 - 2z\sqrt[4]{a/(4b)} + 2\sqrt{a/(4b)}}{z^2 + 2z\sqrt[4]{a/(4b)} + 2\sqrt{a/(4b)}} + \arctan \frac{2z\sqrt[4]{a/(4b)}}{2\sqrt{a/(4b)} - z^2} \right] \qquad ab > 0$$

$$\int \frac{z^2\,dz}{a + bz^4} = \frac{1}{4b\sqrt[4]{-a/b}} \left[\ln \frac{z - \sqrt[4]{-a/b}}{z + \sqrt[4]{-a/b}} + 2 \arctan \frac{z}{\sqrt[4]{-a/b}} \right] \qquad ab < 0$$

$$\int \frac{z^3\,dz}{a + bz^4} = \frac{1}{4b} \ln (a + bz^4) \qquad \int \frac{dz}{x(a + bz^n)} = \frac{1}{an} \ln \frac{z^n}{a + bz^n}$$

FORMS CONTAINING $c^3 \pm z^3$

$$\int \frac{dz}{c^3 \pm z^3} = \pm \frac{1}{6c^2} \ln \frac{(c \pm z)^3}{c^3 \pm z^3} + \frac{1}{c^2\sqrt{3}} \arctan \frac{2z \mp c}{c\sqrt{3}}$$

$$\int \frac{dz}{(c^3 \pm z^3)^2} = \frac{z}{3c^3(c^3 \pm z^3)} + \frac{2}{3c^3} \int \frac{dz}{c^3 \pm z^3}$$

$$\int \frac{dz}{(c^3 \pm z^3)^{n+1}} = \frac{1}{3nc^3} \left[\frac{z}{(c^3 \pm z^3)^n} + (3n - 1) \int \frac{dz}{(c^3 \pm z^3)^n} \right]$$

$$\int \frac{z\,dz}{c^3 \pm z^3} = \frac{1}{6c} \ln \frac{c^3 \pm z^3}{(c \pm z)^3} \pm \frac{1}{c\sqrt{3}} \arctan \frac{2z \mp c}{c\sqrt{3}}$$

$$\int \frac{z\,dz}{(c^3 \pm z^3)^2} = \frac{z^2}{3c^3(c^3 \pm z^3)} + \frac{1}{3c^3} \int \frac{z\,dz}{c^3 \pm z^3}$$

$$\int \frac{z\,dz}{(c^3 \pm z^3)^{n+1}} = \frac{1}{3nc^3} \left[\frac{z^2}{(c^3 \pm z^3)^n} + (3n - 2) \int \frac{z\,dz}{(c^3 \pm z^3)^n} \right]$$

$$\int \frac{z^2\,dz}{c^3 \pm z^3} = \pm \frac{1}{3} \ln(c^3 \pm z^3) \qquad \int \frac{z^2\,dz}{(c^3 \pm z^3)^{n+1}} = \mp \frac{1}{3n(c^3 \pm z^3)^n}$$

$$\int \frac{dz}{z(c^3 \pm z^3)} = \frac{1}{3c^3} \ln \frac{z^3}{c^3 \pm z^3}$$

$$\int \frac{dz}{z(c^3 \pm z^3)^2} = \frac{1}{3c^3(c^3 \pm z^3)} + \frac{1}{3c^6} \ln \frac{z^3}{c^3 \pm z^3}$$

$$\int \frac{dz}{z(c^3 \pm z^3)^{n+1}} = \frac{1}{3nc^3(c^3 \pm z^3)^n} + \frac{1}{c^3} \int \frac{dz}{z(c^3 \pm z^3)^n}$$

$$\int \frac{dz}{z^2(c^3 \pm z^3)} = -\frac{1}{c^3 z} \mp \frac{1}{c^3} \int \frac{z\,dz}{c^3 \pm z^3}$$

$$\int \frac{dz}{z^2(c^3 \pm z^3)^{n+1}} = \frac{1}{c^3} \int \frac{dz}{z^2(c^3 \pm z^3)^n} \mp \frac{1}{c^3} \int \frac{z\,dz}{(c^3 \pm z^3)^{n+1}}$$

FORMS CONTAINING $c^4 \pm z^4$

$$\int \frac{dz}{c^4 + z^4} = \frac{1}{2c^3\sqrt{2}}\left[\frac{1}{2}\ln\frac{z^2 + cz\sqrt{2} + c^2}{z^2 - cz\sqrt{2} + c^2} + \arctan\frac{cz\sqrt{2}}{c^2 - z^2}\right]$$

$$\int \frac{dz}{c^4 - z^4} = \frac{1}{2c^3}\left[\frac{1}{2}\ln\frac{c + z}{c - z} + \arctan\frac{z}{c}\right]$$

$$\int \frac{z\,dz}{c^4 + z^4} = \frac{1}{2c^2}\arctan\frac{z^2}{c^2} \qquad \int \frac{z\,dz}{c^4 - z^4} = \frac{1}{4c^2}\ln\frac{c^2 + z^2}{c^2 - z^2}$$

$$\int \frac{z^2\,dz}{c^4 + z^4} = \frac{1}{2c\sqrt{2}}\left[\frac{1}{2}\ln\frac{z^2 - cz\sqrt{2} + c^2}{z^2 + cz\sqrt{2} + c^2} + \arctan\frac{cz\sqrt{2}}{c^2 - z^2}\right]$$

$$\int \frac{z^2 dz}{c^4 - z^4} = \frac{1}{2c}\left[\frac{1}{2}\ln\frac{c + z}{c - z} - \arctan\frac{z}{c}\right] \qquad \int \frac{z^3\,dz}{c^4 \pm z^4} = \pm\frac{1}{4}\ln(c^4 \pm z^4)$$

FORMS CONTAINING $a + bz + cz^2$

$$Z = a + bz + cz^2 \qquad q = 4ac - b^2$$

$$\int \frac{dz}{Z} = \frac{2}{\sqrt{q}}\arctan\frac{2cz + b}{\sqrt{q}} \quad q > 0 \qquad \int \frac{dz}{Z} = \frac{-2}{\sqrt{-q}}\operatorname{arctanh}\frac{2cz + b}{\sqrt{-q}} \quad q < 0$$

$$\int \frac{dz}{Z^2} = \frac{2cz + b}{qZ} + \frac{2c}{q}\int \frac{dz}{Z} \qquad \int \frac{dz}{Z^3} = \frac{2cz + b}{q}\left[\frac{1}{2Z^2} + \frac{3c}{qZ}\right] + \frac{6c^2}{q^2}\int \frac{dz}{Z}$$

$$\int \frac{z\,dz}{Z} = \frac{1}{2c}\ln Z - \frac{b}{2c}\int \frac{dz}{Z} \qquad \int \frac{z\,dz}{Z^2} = -\frac{bz + 2a}{qZ} - \frac{b}{q}\int \frac{dz}{Z}$$

$$\int \frac{z^2\,dz}{Z} = \frac{z}{c} - \frac{b}{2c^2}\ln Z + \frac{b^2 - 2ac}{2c^2}\int \frac{dz}{Z}$$

$$\int \frac{z^2\,dz}{Z^2} = \frac{(b^2 - 2ac)z + ab}{c\,q\,Z} + \frac{2a}{q}\int \frac{dz}{Z}$$

$$\int \frac{dz}{z\,Z} = \frac{1}{2a}\ln \frac{z^2}{Z} - \frac{b}{2a}\int \frac{dz}{Z}$$

$$\int \frac{dz}{z^2\,Z} = \frac{b}{2a^2}\ln \frac{Z}{z^2} - \frac{1}{az} + \left[\frac{b^2}{2a^2} - \frac{c}{a}\right]\int \frac{dz}{Z}$$

TRIGONOMETRIC FUNCTION INTEGRALS

The substitution w = 2 arctan z will replace any rational function of sin z and cos z by a rational function of w.

$$\sin z = \frac{2w}{1+w^2} \qquad \cos z = \frac{1-w^2}{1+w^2} \qquad dz = \frac{2\,dw}{1+w^2}$$

After integrating, return to the original variable with z = tan(w/2).

$$\int \sin az\,dz = -\frac{1}{a}\cos az \qquad \int \cos az\,dz = \frac{1}{a}\sin az$$

$$\int (\tan az)\, dz = -\frac{1}{a} \ln \cos az \qquad \int (\cot az)\, dz = \frac{1}{a} \ln \sin az$$

$$\int (\sec az)\, dz = \frac{1}{a} \ln(\sec az + \tan az) = \frac{1}{a} \ln \tan\left(\frac{\pi}{4} + \frac{az}{2}\right)$$

$$\int (\csc az)\, dz = \frac{1}{a} \ln(\csc az - \cot az) = \frac{1}{a} \ln \tan \frac{az}{2}$$

$$\int (\sin^2 az)\, dz = -\frac{1}{2a} \cos az \sin az + \frac{1}{2}z = \frac{1}{2}z - \frac{1}{4a} \sin 2az$$

$$\int (\sin^3 az)\, dz = -\frac{1}{3a} (\cos az)(\sin^2 az + 2)$$

$$\int (\sin^4 az)\, dz = \frac{3z}{8} - \frac{\sin 2az}{4a} + \frac{\sin 4az}{32a}$$

$$\int (\cos^2 az)\, dz = \frac{1}{2a} \sin az \cos az + \frac{1}{2}z = \frac{1}{2}z + \frac{1}{4a} \sin 2az$$

$$\int (\cos^3 az)\, dz = \frac{1}{3a} (\sin az)(\cos^2 az + 2) \qquad \int (\cos^4 az)\, dz = \frac{3z}{8} + \frac{\sin 2az}{4a} + \frac{\sin 4az}{32a}$$

$$\int \frac{dz}{\sin^2 az} = \int (\csc^2 az)\, dz = -\frac{1}{a} \cot az \qquad \int \frac{dz}{\cos^2 az} = \int (\sec^2 az)\, dz = \frac{1}{a} \tan az$$

$$\int (\sin mz)(\sin nz)\,dz = \frac{\sin(m-n)z}{2(m-n)} - \frac{\sin(m+n)z}{2(m+n)} \qquad m^2 \neq n^2$$

$$\int (\cos mz)(\cos nz)\,dz = \frac{\sin(m-n)z}{2(m-n)} + \frac{\sin(m+n)z}{2(m+n)} \qquad m^2 \neq n^2$$

$$\int (\sin az)(\cos az)\,dz = \frac{1}{2a}\sin^2 az$$

$$\int (\sin mz)(\cos nz)\,dz = -\frac{\cos(m-n)z}{2(m-n)} - \frac{\cos(m+n)z}{2(m+n)} \qquad m^2 \neq n^2$$

$$\int (\sin^2 az)(\cos^2 az)\,dz = \frac{-1}{32a}\sin 4az + \frac{z}{8} \qquad \int \frac{\sin az}{\cos^2 az}\,dz = \frac{1}{a\cos az} = \frac{\sec az}{a}$$

$$\int (\sin az)(\cos^m az)\,dz = -\frac{\cos^{m+1} az}{(m+1)a} \qquad \int (\sin^m az)(\cos az)\,dz = \frac{\sin^{m+1} az}{(m+1)a}$$

$$\int \frac{\sin^2 az}{\cos az}\,dz = -\frac{1}{a}\sin az + \frac{1}{a}\ln\tan\left(\frac{\pi}{4} + \frac{az}{2}\right)$$

$$\int \frac{\cos az}{\sin^2 az}\,dz = \frac{-1}{a\sin az} = -\frac{\csc az}{a} \qquad \int \frac{dz}{(\sin az)(\cos az)} = \frac{1}{a}\ln\tan az$$

$$\int \frac{dz}{(\sin az)(\cos^2 az)} = \frac{1}{a}\left(\sec az + \ln\tan\frac{az}{2}\right)$$

$$\int \frac{dz}{(\sin^2 az)(\cos az)} = -\frac{1}{a}\csc az + \frac{1}{a}\ln \tan\left(\frac{\pi}{4} + \frac{az}{2}\right)$$

$$\int \frac{dz}{(\sin^2 az)(\cos^2 az)} = -\frac{2}{a}\cot 2az$$

$$\int \sin(a + bz)\, dz = -\frac{1}{b}\cos(a + bz)$$

$$\int \cos(a + bz)\, dz = \frac{1}{b}\sin(a + bz)$$

$$\int \frac{dz}{1 \pm \sin az} = \mp \frac{1}{a}\tan\left(\frac{\pi}{4} \mp \frac{az}{2}\right)$$

$$\int \frac{dz}{1 + \cos az} = \frac{1}{a}\tan\frac{az}{2}$$

$$\int \frac{dz}{1 - \cos az} = -\frac{1}{a}\cot\frac{az}{2}$$

$$\int \frac{\sin az}{1 \pm \sin az}\, dz = \pm z + \frac{1}{a}\tan\left(\frac{\pi}{4} \mp \frac{az}{2}\right)$$

$$\int \frac{dz}{(\sin az)(1 \pm \sin az)} = \frac{1}{a}\tan\left(\frac{\pi}{4} \mp \frac{az}{2}\right) + \ln \tan\frac{az}{2}$$

$$\int \frac{dz}{(1 + \sin az)^2} = -\frac{1}{2a}\tan\left(\frac{\pi}{4} - \frac{az}{2}\right) - \frac{1}{6a}\tan^3\left(\frac{\pi}{4} - \frac{az}{2}\right)$$

$$\int \frac{dz}{(1 - \sin az)^2} = \frac{1}{2a}\cot\left(\frac{\pi}{4} - \frac{az}{2}\right) + \frac{1}{6a}\cot^3\left(\frac{\pi}{4} - \frac{az}{2}\right)$$

$$\int \frac{\sin az}{(1 + \sin az)^2}\, dz = -\frac{1}{2a}\tan\left(\frac{\pi}{4} - \frac{az}{2}\right) + \frac{1}{6a}\tan^3\left(\frac{\pi}{4} - \frac{az}{2}\right)$$

$$\int \frac{\sin az}{(1 - \sin az)^2}\,dz = -\frac{1}{2a}\cot\left(\frac{\pi}{4} - \frac{az}{2}\right) + \frac{1}{6a}\cot^3\left(\frac{\pi}{4} - \frac{az}{2}\right)$$

$$\int \frac{\sin z\,dz}{a + b\sin z} = \frac{z}{b} - \frac{a}{b}\int \frac{dz}{a + b\sin z}$$

$$\int \frac{dz}{(\sin z)(a + b\sin z)} = \frac{1}{a}\ln\tan\frac{z}{2} - \frac{b}{a}\int \frac{dz}{a + b\sin z}$$

$$\int \frac{dz}{(a + b\sin z)^2} = \frac{b\cos z}{(a^2 - b^2)(a + b\sin z)} + \frac{a}{a^2 - b^2}\int \frac{dz}{a + b\sin z}$$

$$\int \frac{\sin z\,dz}{(a + b\sin z)^2} = \frac{a\cos z}{(b^2 - a^2)(a + b\sin z)} + \frac{b}{b^2 - a^2}\int \frac{dz}{a + b\sin z}$$

$$\int \frac{\cos az}{a + \cos az}\,dz = z - \frac{1}{a}\tan\frac{az}{2} \qquad \int \frac{\cos az}{1 - \cos az}\,dz = -z - \frac{1}{a}\cot\frac{az}{2}$$

$$\int \frac{dz}{(\cos az)(1 + \cos az)} = \frac{1}{a}\ln\tan\left(\frac{\pi}{4} + \frac{az}{2}\right) - \frac{1}{a}\tan\frac{az}{2}$$

$$\int \frac{dz}{(\cos az)(1 - \cos az)} = \frac{1}{a}\ln\tan\left(\frac{\pi}{4} + \frac{az}{2}\right) - \frac{1}{a}\cot\frac{az}{2}$$

$$\int \frac{dz}{(1 + \cos az)^2} = \frac{1}{2a}\tan\frac{az}{2} + \frac{1}{6a}\tan^3\frac{az}{2}$$

$$\int \frac{dz}{(1 - \cos az)^2} = -\frac{1}{2a}\cot\frac{az}{2} - \frac{1}{6a}\cot^3\frac{az}{2}$$

$$\int \frac{\cos az}{(1 + \cos az)^2}\,dz = \frac{1}{2a}\tan\frac{az}{2} - \frac{1}{6a}\tan^3\frac{az}{2}$$

$$\int \frac{\cos az}{(1 - \cos az)^2}\,dz = \frac{1}{2a}\cot\frac{az}{2} - \frac{1}{6a}\cot^3\frac{az}{2}$$

$$\int \frac{\cos z\,dz}{a + b\cos z} = \frac{z}{b} - \frac{a}{b}\int \frac{dz}{a + b\cos z}$$

$$\int \frac{dz}{(\cos z)(a + b\cos z)} = \frac{1}{a}\ln\tan\left(\frac{z}{2} + \frac{\pi}{4}\right) - \frac{b}{a}\int \frac{dz}{a + b\cos z}$$

$$\int \frac{dz}{(a + b\cos z)^2} = \frac{b\sin z}{(b^2 - a^2)(a + b\cos z)} - \frac{a}{b^2 - a^2}\int \frac{dz}{a + b\cos z}$$

$$\int \frac{\cos z\,dz}{(a + b\cos z)^2} = \frac{a\sin z}{(a^2 - b^2)(a + b\cos z)} - \frac{b}{a^2 - b^2}\int \frac{dz}{a + b\cos z}$$

$$\int \frac{\sin az}{1 \pm \cos az}\,dz = \mp\frac{1}{a}\ln(1 \pm \cos az) \qquad \int \frac{\cos az}{1 \pm \sin az}\,dz = \pm\frac{1}{a}\ln(1 \pm \sin az)$$

$$\int \frac{dz}{(\sin az)(1 \pm \cos az)} = \pm\frac{1}{2a(1 \pm \cos az)} + \frac{1}{2a}\ln\tan\frac{az}{2}$$

$$\int \frac{dz}{(\cos az)(1 \pm \sin az)} = \mp \frac{1}{2a(1 \pm \sin az)} + \frac{1}{2a} \ln \tan\left(\frac{\pi}{4} + \frac{az}{2}\right)$$

$$\int \frac{\sin az}{(\cos az)(1 \pm \cos az)} dz = \frac{1}{a} \ln(\sec az \pm 1)$$

$$\int \frac{\cos az}{(\sin az)(1 \pm \sin az)} dz = -\frac{1}{a} \ln(\csc az \pm 1)$$

$$\int \frac{\sin az}{(\cos az)(1 \pm \sin az)} dz = \frac{1}{2a(1 \pm \sin az)} \pm \frac{1}{2a} \ln \tan\left(\frac{\pi}{4} + \frac{az}{2}\right)$$

$$\int \frac{\cos az}{(\sin az)(1 \pm \cos az)} dz = -\frac{1}{2a(1 \pm \cos az)} \pm \frac{1}{2a} \ln \tan \frac{az}{2}$$

$$\int \frac{dz}{\sin az \pm \cos az} = \frac{1}{a\sqrt{2}} \ln \tan\left(\frac{az}{2} \pm \frac{\pi}{8}\right)$$

$$\int \frac{dz}{(\sin az \pm \cos az)^2} = \frac{1}{2a} \tan\left(az \mp \frac{\pi}{4}\right)$$

$$\int \frac{dz}{1 + \cos az \pm \sin az} = \pm \frac{1}{a} \ln\left(1 \pm \tan \frac{az}{2}\right)$$

$$\int \frac{dz}{a^2 \cos^2 cz - b^2 \sin^2 cz} = \frac{1}{2abc} \ln \frac{b \tan cz + a}{b \tan cz - a}$$

$$\int z (\sin az)\, dz = \frac{1}{a^2} \sin az - \frac{z}{a} \cos az$$

$$\int z^2 (\sin az)\, dz = \frac{2z}{a^2} \sin az - \frac{a^2z^2 - 2}{a^3} \cos az$$

$$\int z^3 (\sin az)\, dz = \frac{3a^2z^2 - 6}{a^4} \sin az - \frac{a^2z^3 - 6z}{a^3} \cos az$$

$$\int z (\cos az)\, dz = \frac{1}{a^2} \cos az + \frac{z}{a} \sin az$$

$$\int z^2 (\cos az)\, dz = \frac{2z \cos az}{a^2} + \frac{a^2z^2 - 2}{a^3} \sin az$$

$$\int z^3 (\cos az)\, dz = \frac{3a^2z^2 - 6}{a^4} \cos az + \frac{a^2z^3 - 6z}{a^3} \sin az$$

$$\int z^n (\sin az)\, dz = -\sum_{k=0}^{n} k\,! \binom{n}{k} \frac{z^{n-k}}{a^{k+1}} \cos\left(az + \frac{1}{2} k\pi\right)$$

$$\int z^n (\cos az)\, dz = \sum_{k=0}^{n} k\,! \binom{n}{k} \frac{z^{n-k}}{a^{k+1}} \sin\left(az + \frac{1}{2} k\pi\right)$$

$$\int \frac{z}{1 \pm \sin az}\, dz = \mp \frac{z \cos az}{a(1 \pm \sin az)} + \frac{1}{a^2} \ln(1 \pm \sin az)$$

$$\int \frac{z}{1 + \cos az}\, dz = \frac{z}{a} \tan \frac{az}{2} + \frac{2}{a^2} \ln \cos \frac{az}{2}$$

$$\int \frac{z}{1 - \cos az}\, dz = -\frac{z}{a} \cot \frac{az}{2} + \frac{2}{a^2} \ln \sin \frac{az}{2}$$

$$\int \frac{z + \sin z}{1 + \cos z}\, dz = z \tan \frac{z}{2} \qquad \int \frac{z - \sin z}{1 - \cos z}\, dz = -z \cot \frac{z}{2}$$

$$\int \sqrt{1 - \cos az}\, dz = -\frac{2 \sin az}{a\sqrt{1 - \cos az}} = -\frac{2\sqrt{2} \cos\left(\frac{az}{2}\right)}{a}$$

$$\int \sqrt{1 + \cos az}\, dz = \frac{2 \sin az}{a\sqrt{1 + \cos az}} = \frac{2\sqrt{2} \sin\left(\frac{az}{2}\right)}{a}$$

$$\int (\tan^2 az)\, dz = \frac{1}{a} \tan az - z \qquad \int (\cot^2 az)\, dz = -\frac{1}{a} \cot az - z$$

$$\int (\tan^3 az)\, dz = \frac{1}{2a} \tan^2 az + \frac{1}{a} \ln \cos az$$

$$\int (\tan^4 az)\, dz = \frac{\tan^3 az}{3a} - \frac{1}{a} \tan z + z$$

$$\int (\tan^n az)\, dz = \frac{\tan^{n-1} az}{a(n - 1)} - \int (\tan^{n-2} az)\, dz$$

$$\int (\cot^3 az)\, dz = -\frac{1}{2a}\cot^2 az - \frac{1}{a}\ln \sin az$$

$$\int (\cot^4 az)\, dz = -\frac{1}{3a}\cot^3 az + \frac{1}{a}\cot az + z$$

$$\int \frac{\sin az}{\sqrt{1 + b^2\sin^2 az}}\, dz = -\frac{1}{ab}\arcsin\frac{b\cos az}{\sqrt{1 + b^2}}$$

$$\int \frac{\sin az}{\sqrt{1 - b^2\sin az}}\, dz = -\frac{1}{ab}\ln\left(b\cos az + \sqrt{1 - b^2\sin^2 az}\right)$$

$$\int \frac{\cos az}{\sqrt{1 + b^2\sin^2 az}}\, dz = \frac{1}{ab}\ln\left(b\sin az + \sqrt{1 + b^2\sin^2 az}\right)$$

$$\int \frac{\cos az}{\sqrt{1 - b^2\sin^2 az}}\, dz = \frac{1}{ab}\arcsin(b\sin az)$$

INVERSE TRIGONOMETRIC FUNCTION INTEGRALS

$$\int (\arcsin az)\, dz = z\arcsin az + \frac{\sqrt{1 - a^2z^2}}{a}$$

$$\int (\arccos az)\, dz = z\arccos az - \frac{\sqrt{1 - a^2z^2}}{a}$$

$$\int (\arctan az)\, dz = z \arctan az - \frac{1}{2a} \ln(1 + a^2z^2)$$

$$\int (\operatorname{arccot} az)\, dz = z \operatorname{arccot} az + \frac{1}{2a} \ln(1 + a^2z^2)$$

$$\int (\operatorname{arcsec} az)\, dz = z \operatorname{arcsec} az - \frac{1}{a} \ln\left(az + \sqrt{a^2z^2 - 1} \right)$$

$$\int (\operatorname{arccsc} az)\, dz = z \operatorname{arccsc} az + \frac{1}{a} \ln\left(az + \sqrt{a^2z^2 - 1} \right)$$

$$\int z (\arcsin az)\, dz = \frac{1}{4a^2} \left[(2a^2z^2 - 1) \arcsin(az) + az \sqrt{1 - a^2z^2} \right]$$

$$\int z (\arccos az)\, dz = \frac{1}{4a^2} \left[(2a^2z^2 - 1) \arccos(az) - az \sqrt{1 - a^2z^2} \right]$$

$$\int z (\arctan az)\, dz = \frac{1 + a^2z^2}{2a^2} \arctan az - \frac{z}{2a}$$

$$\int z (\operatorname{arccot} az)\, dz = \frac{1 + a^2z^2}{2a^2} \operatorname{arccot} az + \frac{z}{2a}$$

$$\int \frac{\arcsin az}{z^2}\, dz = a \ln\left(\frac{1 - \sqrt{1 - a^2z^2}}{z}\right) - \frac{\arcsin az}{z}$$

$$\int \frac{\arccos az\, dz}{z^2} = -\frac{1}{z} \arccos az + a \ln \frac{1 + \sqrt{1 - a^2z^2}}{z}$$

$$\int \frac{\arctan az\, dz}{z^2} = -\frac{1}{z}\arctan az - \frac{a}{2}\ln\frac{1+a^2z^2}{z^2}$$

$$\int \frac{\operatorname{arccot} az}{z^2}\, dz = -\frac{1}{z}\operatorname{arccot} az - \frac{a}{2}\ln\frac{z^2}{a^2z^2+1}$$

$$\int (\arcsin az)^2\, dz = z(\arcsin az)^2 - 2z + \frac{2\sqrt{1-a^2z^2}}{a}\arcsin az$$

$$\int (\arccos az)^2\, dz = z(\arccos az)^2 - 2z - \frac{2\sqrt{1-a^2z^2}}{a}\arccos az$$

$$\int \frac{\arcsin az}{\sqrt{1-a^2z^2}}\, dz = \frac{1}{2a}(\arcsin az)^2 \qquad \int \frac{\arccos az}{\sqrt{1-a^2z^2}}\, dz = -\frac{1}{2a}(\arccos az)^2$$

$$\int \frac{\arctan az}{a^2z^2+1}\, dz = \frac{1}{2a}(\arctan az)^2 \qquad \int \frac{\operatorname{arccot} az}{a^2z^2+1}\, dz = -\frac{1}{2a}(\operatorname{arccot} az)^2$$

$$\int z(\operatorname{arcsec} az)\, dz = \frac{z^2}{2}\operatorname{arcsec} az - \frac{1}{2a^2}\sqrt{a^2z^2-1}$$

$$\int z(\operatorname{arccsc} az)\, dz = \frac{z^2}{2}\operatorname{arccsc} az + \frac{1}{2a^2}\sqrt{a^2z^2-1}$$

EXPONENTIAL FUNCTION INTEGRALS

$$\int e^z \, dz = e^z$$

$$\int e^{az} \, dz = \frac{e^{az}}{a}$$

$$\int e^{-z} \, dz = -e^{-z}$$

$$\int z \, e^{az} \, dz = \frac{e^{az}}{a^2}(az - 1)$$

$$\int z^m e^{az} \, dz = e^{az} \sum_{r=0}^{m} (-1)^r \frac{m! \, z^{m-r}}{(m - r)! \, a^{r+1}}$$

$$\int e^{az} \ln z \, dz = \frac{e^{az} \ln z}{a} - \frac{1}{a} \int \frac{e^{az}}{z} \, dz$$

$$\int \frac{dz}{ae^{mz} + be^{-mz}} = \frac{1}{m\sqrt{ab}} \arctan\left(e^{mz} \sqrt{\frac{a}{b}}\right) \qquad a > 0 \qquad b > 0$$

$$\int \frac{dz}{ae^{mz} - be^{-mz}} = \frac{-1}{m\sqrt{ab}} \operatorname{arctanh}\left(e^{mz} \sqrt{\frac{a}{b}}\right) \qquad a > 0 \qquad b > 0$$

$$\int (a^z - a^{-z}) \, dz = \frac{a^z + a^{-z}}{\ln a}$$

$$\int \frac{e^{az}}{b + ce^{az}} \, dz = \frac{1}{ac} \ln(b + ce^{az})$$

$$\int \frac{z \, e^{az}}{(1 + az^2)} \, dz = \frac{e^{az}}{a^2(1 + az)}$$

$$\int z \, e^{-z^2} \, dz = -\frac{1}{2} e^{-x^2}$$

$$\int e^{az}\,[\sin(bz)]\,dz = \frac{e^{az}\,[a\sin(bz) - b\cos(bz)]}{a^2 + b^2}$$

$$\int e^{az}\,[\sin(bz)]\,[\sin(cz)]\,dz = \frac{e^{az}\,[(b-c)\sin(b-c)z + a\cos(b-c)z]}{2[a^2 + (b-c)^2]} - \frac{e^{az}\,[(b+c)\sin(b+c)z + a\cos(b+c)z]}{2[a^b + (b+c)^2]}$$

$$\int e^{az}\,[\sin(bz)]\,[\cos(cz)]\,dz = \frac{e^{az}\,[a\sin(b-c)z - (b-c)\cos(b-c)z]}{2[a^2 + (b-c)^2]} + \frac{e^{az}\,[a\sin(b+c)z - (b+c)\cos(b+c)z]}{2[a^b + (b+c)^2]}$$

$$\int e^{az}\,[\sin(bz)][\sin(bz+c)]\,dz = \frac{e^{az}\cos c}{2a} - \frac{e^{az}[a\cos(2bz+c) + 2b\sin(2bz+c)]}{2(a^2 + 4b^2)}$$

$$\int e^{az}\,[\sin(bz)][\cos(bz+c)]\,dz = \frac{-e^{az}\sin c}{2a} + \frac{e^{az}\,[a\sin(2bz+c) - 2b\cos(2bz+c)]}{2(a^2 + 4b^2)}$$

$$\int e^{az}\,[\cos(bz)]\,dz = \frac{e^{az}}{a^2 + b^2}\,[a\cos(bz) + b\sin(bz)]$$

$$\int e^{az}\,[\cos(bz)][\cos(cz)]\,dz = \frac{e^{az}[(b-c)\sin(b-c)z + a\cos(b-c)z]}{2[a^2 + (b-c)^2]} + \frac{e^{az}\,[(b+c)\sin(b+c)z + a\cos(b+c)z]}{2[a^2 + (b+c)^2]}$$

$$\int e^{az}\,[\cos(bz)][\cos(bz+c)]\,dz = \frac{e^{az}\cos c}{2a} + \frac{e^{az}\,[a\cos(2bz+c) + 2b\sin(2bz+c)]}{2(a^2 + 4b^2)}$$

$$\int e^{az} [\cos(bz)][\sin(bz + c)]\, dz = \frac{e^{az} \sin c}{2a} + \frac{e^{az}[a \sin(2bz+c) - 2b \cos(2bz+c)]}{2(a^2 + 4b^2)}$$

$$\int z\, e^{az} (\sin bz)\, dz = \frac{z\, e^{az}}{a^2 + b^2} (a \sin bz - b \cos bz) - \frac{e^{az}}{(a^2 + b^2)^2} [(a^2 - b^2) \sin bz - 2ab \cos bz]$$

$$\int z\, e^{az} (\cos bz)\, dz = \frac{z\, e^{az}}{a^2 + b^2} (a \cos bz + b \sin bz) - \frac{e^{az}}{(a^2 + b^2)^2} [(a^2 - b^2) \cos bz + 2ab \sin bz]$$

LOGARITHMIC FUNCTION INTEGRALS

$$\int (\ln z)\, dz = z \ln z - z$$

$$\int z (\ln z)\, dz = \frac{z^2}{2} \ln z - \frac{z^2}{4}$$

$$\int z^2 (\ln z)\, dz = \frac{z^3}{3} \ln z - \frac{z^3}{9}$$

$$\int (\ln z)^n\, dz = (-1)^n\, n\, !\, z \sum_{r=0}^{n} \frac{(-\ln z)^r}{r!}$$

$$\int z^n (\ln az)\, dz = \frac{z^{n+1}}{n + 1} \ln az - \frac{z^{n+1}}{(n + 1)^2}$$

$$\int (\ln z)^2\, dz = z (\ln z)^2 - 2z \ln z + 2z$$

$$\int \frac{(\ln z)^n}{z}\, dz = \frac{1}{n + 1} (\ln z)^{n+1}$$

$$\int \frac{dz}{z \ln z} = \ln(\ln z) \qquad \int \frac{dz}{z (\ln z)^n} = -\frac{1}{(n-1)(\ln z)^{n-1}}$$

$$\int z^m (\ln z)^n \, dz = (-1)^n \frac{n!}{m+1} z^{m+1} \sum_{r=0}^{n} \frac{(-\ln z)^r}{r!(m+1)^{n-r}}$$

$$\int [\ln (az + b)] \, dz = \frac{az + b}{a} \ln(az + b) - z$$

$$\int \frac{\ln (az + b)}{z^2} \, dz = \frac{a}{b} \ln z - \frac{az + b}{bz} \ln(az + b)$$

$$\int \left[\ln \frac{z + a}{z - a}\right] dz = (z + a) \ln(z + a) - (z - a) \ln(z - a)$$

$$\int \frac{1}{z^2} \left[\ln \frac{z + a}{z - a}\right] dz = \frac{1}{z} \ln \frac{z - a}{z + a} - \frac{1}{a} \ln \frac{z^2 - a^2}{z^2}$$

$$\int [\ln (z^2 + a^2)] \, dz = z \ln(z^2 + a^2) - 2z + 2a \arctan \frac{z}{a}$$

$$\int [\ln(z^2 - a^2)] \, dz = z \ln(z^2 - a^2) - 2z + a \ln \frac{z + a}{z - a}$$

$$\int z \, [\ln(z^2 \pm a^2)] \, dz = \frac{1}{2} (z^2 \pm a^2) \ln(z^2 \pm a^2) - \frac{1}{2} z^2$$

$$\int \left[\ln\left(z + \sqrt{z^2 \pm a^2}\right)\right] dz = z \ln\left(z + \sqrt{z^2 \pm a^2}\right) - \sqrt{z^2 \pm a^2}$$

$$\int z\left[\ln\left(z + \sqrt{z^2 \pm a^2}\right)\right] dz = \left(\frac{z^2}{2} \pm \frac{a^2}{4}\right) \ln\left(z + \sqrt{z^2 \pm a^2}\right) - \frac{z\sqrt{z^2 \pm a^2}}{4}$$

$$\int \frac{\ln\left(z + \sqrt{z^2 + a^2}\right)}{z^2} dz = -\frac{\ln\left(z + \sqrt{z^2 + a^2}\right)}{z} - \frac{1}{a} \ln \frac{a + \sqrt{z^2 + a^2}}{z}$$

$$\int \frac{\ln\left(z + \sqrt{z^2 - a^2}\right)}{z^2} dz = -\frac{\ln\left(z + \sqrt{z^2 - a^2}\right)}{z} + \frac{1}{|a|} \operatorname{arcsec} \frac{x}{a}$$

HYPERBOLIC FUNCTION INTEGRALS

$$\int \sinh z\, dz = \cosh z \qquad \int \cosh z\, dz = \sinh z$$

$$\int \tanh z\, dz = \ln \cosh z \qquad \int \coth z\, dz = \ln \sinh z$$

$$\int \operatorname{sech} z\, dz = \arctan(\sinh z) \qquad \int \operatorname{csch} z\, dz = \ln \tanh(z/2)$$

$$\int z (\sinh z)\, dz = z \cosh z - \sinh z \qquad \int z (\cosh z)\, dz = z \sinh z - \cosh z$$

$$\int (\operatorname{sech} z)(\tanh z)\, dz = -\operatorname{sech} z \qquad \int (\operatorname{csch} z)(\coth z)\, dz = -\operatorname{csch} z$$

$$\int (\sinh^2 z)\, dz = \frac{\sinh 2z}{4} - \frac{z}{2} \qquad \int (\cosh^2 z)\, dz = \frac{\sinh 2z}{4} + \frac{z}{2}$$

$$\int (\tanh^2 z)\, dz = z - \tanh z \qquad \int (\coth^2 z)\, dz = z - \coth z$$

$$\int (\operatorname{sech}^2 z)\, dz = \tanh z \qquad \int (\operatorname{csch}^2 z)\, dz = -\coth z$$

$$\int (\sinh mz)(\sinh nz)\, dz = \frac{\sinh(m+n)z}{2(m+n)} - \frac{\sinh(m-n)z}{2(m-n)} \qquad m^2 \neq n^2$$

$$\int (\cosh mz)(\cosh nz)\, dz = \frac{\sinh(m+n)z}{2(m+n)} + \frac{\sinh(m-n)z}{2(m-n)} \qquad m^2 \neq n^2$$

$$\int (\sinh mz)(\cosh nz)\, dz = \frac{\cosh(m+n)z}{2(m+n)} + \frac{\cosh(m-n)z}{2(m-n)} \qquad m^2 \neq n^2$$

INVERSE HYPERBOLIC FUNCTION INTEGRALS

$$\int \left(\operatorname{arcsinh} \frac{z}{a}\right) dz = z \operatorname{arcsinh} \frac{z}{a} - \sqrt{z^2 + a^2} \qquad a > 0$$

$$\int \left(\operatorname{arccosh} \frac{z}{a}\right) dz = \begin{cases} z \operatorname{arccosh} \dfrac{z}{a} - \sqrt{z^2 - a^2} & \arccos \dfrac{z}{a} > 0 \\ z \operatorname{arccosh} \dfrac{z}{a} + \sqrt{z^2 - a^2} & \arccos \dfrac{z}{a} < 0 \end{cases} \qquad a > 0$$

$$\int z \left(\operatorname{arcsinh} \frac{z}{a}\right) dz = \left(\frac{z^2}{2} + \frac{a^2}{4}\right) \operatorname{arcsinh} \frac{z}{a} - \frac{z}{4} \sqrt{z^2 + a^2} \qquad a > 0$$

$$\int z \left(\operatorname{arccosh} \frac{z}{a}\right) dz = \left(\frac{z^2}{2} - \frac{a^2}{4}\right) \operatorname{arccosh} \frac{z}{a} - \frac{z}{4} \sqrt{z^2 - a^2} \qquad a > 0$$

$$\int \left(\operatorname{arctanh} \frac{z}{a}\right) dz = z \operatorname{arctanh} \frac{z}{a} + \frac{a}{2} \ln(a^2 - z^2) \qquad \left|\frac{z}{a}\right| < 1$$

$$\int \left(\operatorname{arccoth} \frac{z}{a}\right) dz = z \operatorname{arccoth} \frac{z}{a} + \frac{a}{2} \ln(z^2 - a^2) \qquad \left|\frac{z}{a}\right| > 1$$

$$\int z \left(\operatorname{arctanh} \frac{z}{a}\right) dz = \frac{z^2 - a^2}{2} \operatorname{arctanh} \frac{z}{a} + \frac{az}{2} \qquad \left|\frac{z}{a}\right| < 1$$

$$\int z \left(\operatorname{arccoth} \frac{z}{a}\right) dz = \frac{z^2 - a^2}{2} \operatorname{arccoth} \frac{z}{a} + \frac{az}{2} \qquad \left|\frac{z}{a}\right| > 1$$

$$\int (\operatorname{arcsech} z)\, dz = z \operatorname{arcsech} z + \arcsin z$$

$$\int z (\operatorname{arcsech} z)\, dz = \frac{z^2}{2} \operatorname{arcsech} z - \frac{1}{2}\sqrt{1 - z^2}$$

$$\int (\operatorname{arccsch} z)\, dz = z \operatorname{arccsch} z + \frac{z}{|z|} \operatorname{arcsinh} z$$

$$\int z (\operatorname{arccsch} z)\, dz = \frac{z^2}{2} \operatorname{arccsch} z + \frac{1}{2} \frac{z}{|z|} \sqrt{1 + z^2}$$

IRRATIONAL FUNCTION INTEGRATION

Many irrational integrals can be reduced to rational function integrals by the correct substitution. If the integrand is rational except for a radical of the form:

1. $\sqrt[n]{az+b}$, the substitution $az + b = w^n$ will replace it with a rational integrand. For example, let $1 - z = w^2$. Then $z = 1 - w^2$, $dz = -2w\,dw$, and the below integral is

$$\int \frac{dz}{z\sqrt{1-z}} = \int \frac{-2w\,dw}{w(1-w^2)} = -2\int \frac{dw}{1-w^2} = -\ln\left|\frac{1+w}{1-w}\right| = \ln\left|\frac{1-\sqrt{1-z}}{1+\sqrt{1-z}}\right|.$$

2. $\sqrt{a+bz+z^2}$, the substitution $a + bz + z^2 = (w - z)^2$ will replace it with a rational integrand. For example, let $z^2 + z + 2 = (w - z)^2$. Then the below integral is

$$z = \frac{w^2-2}{1+2w} \qquad dz = \frac{2(w^2+w+2)\,dw}{(1+2w)^2} \qquad \sqrt{z^2+z+2} = \frac{w^2+w+2}{1+2w}$$

$$\int \frac{dz}{z\sqrt{z^2+z+2}} = 2\int \frac{dw}{w^2-2} = \frac{1}{\sqrt{2}}\ln\left|\frac{w-\sqrt{2}}{w+\sqrt{2}}\right| = \frac{1}{\sqrt{2}}\ln\left|\frac{\sqrt{z^2+z+2}+z-\sqrt{2}}{\sqrt{z^2+z+2}+z+\sqrt{2}}\right|.$$

3. $\sqrt{a+bz-z^2} = \sqrt{(\alpha+z)(\beta-z)}$, the substitution $a + bz - z^2 = (\alpha + z)^2 w^2$ or $a + bz - z^2 = (\beta - z)^2 w^2$ will replace it with a rational integrand. Let $5 - 4z - z^2 = (5 + z)(1 - z) = (1 - z)^2 z^2$. Then the below integral is

$$z = \frac{w^2-5}{1+w^2} \qquad dz = \frac{12w\,dw}{(1+w^2)^2} \qquad \sqrt{5-4z-z^2} = (1-z)w = \frac{6w}{1+w^2}$$

$$\int \frac{z\,dz}{(5-4z-z^2)^{3/2}} = \frac{1}{18}\int\left[1 - \frac{5}{w^2}\right]dw = \frac{1}{18}\left[w + \frac{5}{w}\right] = \frac{5-2z}{9\sqrt{5-4z-z^2}}.$$

4. $\sqrt{a^2 - b^2z^2}$, the substitution $z = (a/b) \sin w$ gives $a\sqrt{1 - \sin^2 w} = a \cos w$.

5. $\sqrt{a^2 + b^2z^2}$, the substitution $z = (a/b) \tan w$ gives $a\sqrt{1 + \tan^2 w} = a \sec w$.

6. $\sqrt{b^2z^2 - a^2}$, the substitution $z = (a/b) \sec w$ gives $a\sqrt{\sec^2 w - 1} = a \tan w$.

FORMS CONTAINING $\sqrt{a + bz}$

$$\int \sqrt{a + bz}\, dz = \frac{2}{3b}\sqrt{(a + bz)^3}$$

$$\int z\sqrt{a + bz}\, dz = -\frac{2(2a - 3bz)\sqrt{(a + bz)^3}}{15b^2}$$

$$\int z^2\sqrt{a + bz}\, dz = \frac{2(8a^2 - 12abz + 15b^2z^2)\sqrt{(a + bz)^3}}{105b^3}$$

$$\int z^m\sqrt{a + bz}\, dz = \frac{2\sqrt{a + bz}}{b^{m+1}} \sum_{k=0}^{m} \frac{m!\,(-a)^{m-k}}{k!\,(m-k)!\,(2k+3)} (a + bz)^{k+1}$$

$$\int \frac{\sqrt{a + bz}}{z}\, dz = 2\sqrt{a + bz} + a\int \frac{dz}{z\sqrt{a + bz}}$$

$$\int \frac{\sqrt{a + bz}}{z^2}\, dz = -\frac{\sqrt{a + bz}}{z} + \frac{b}{2}\int \frac{dz}{z\sqrt{a + bz}}$$

$$\int \frac{dz}{\sqrt{a + bz}} = \frac{2\sqrt{a + bz}}{b}$$

$$\int \frac{z\, dz}{\sqrt{a + bz}} = -\frac{2(2a - bz)}{3b^2}\sqrt{a + bz}$$

$$\int \frac{z^2\,dz}{\sqrt{a + bz}} = \frac{2(8a^2 - 4abz + 3b^2z^2)}{15b^3}\sqrt{a + bz}$$

$$\int \frac{dz}{z\sqrt{a + bz}} = \frac{1}{\sqrt{a}} \ln\left(\frac{\sqrt{a + bz} - \sqrt{a}}{\sqrt{a + bz} + \sqrt{a}}\right) \qquad a > 0$$

$$\int \frac{dz}{x\sqrt{a + bz}} = \frac{2}{\sqrt{-a}} \arctan\sqrt{\frac{a + bz}{-a}} \qquad a < 0$$

$$\int \frac{dz}{z^2\sqrt{a + bz}} = -\frac{\sqrt{a + bz}}{az} - \frac{b}{2a}\int \frac{dz}{z\sqrt{a + bz}}$$

$$\int (a + bz)^{\pm\frac{n}{2}}\,dz = \frac{2(a + bz)^{\frac{2 \pm n}{2}}}{b(2 \pm n)}$$

$$\int z\,(a + bz)^{\pm\frac{n}{2}}\,dz = \frac{2}{b^2}\left[\frac{(a + bz)^{\frac{4 \pm n}{2}}}{4 \pm n} - \frac{a(a + bz)^{\frac{2 \pm n}{2}}}{2 \pm n}\right]$$

$$\int f\left(z, \sqrt{a + bz}\right) dz = \frac{2}{b}\int f\left(\frac{w^2 - a}{b}, w\right) w\,dw \qquad w = \sqrt{a + bz}$$

FORMS CONTAINING $\sqrt{a + bz}$ AND $\sqrt{c + ez}$

$$\int \frac{dz}{\sqrt{(a + bz)(c + ez)}} = \frac{1}{\sqrt{be}} \ln \frac{\left[b(c + ez) + \sqrt{be(a + bz)(c + ez)}\right]^2}{c + ez} \qquad be > 0$$

$$\int \frac{dz}{\sqrt{(a + bz)(c + ez)}} = \frac{2}{\sqrt{-be}} \arctan \frac{\sqrt{-be(a + bz)(c + ez)}}{b(c + ez)} \qquad be < 0$$

$$\int \sqrt{(a + bz)(c + ez)}\, dz = \frac{(ae - bc) + 2b(c + ez)}{4be} \sqrt{(a + bz)(c + ez)} - \frac{(ae - bc)^2}{8be} \int \frac{dz}{\sqrt{(a + bz)(c + ez)}}$$

$$\int \frac{dz}{(c + ez)\sqrt{a + bz}} = \frac{1}{\sqrt{(ae - bc)e}} \ln \frac{e\sqrt{a + bz} - \sqrt{(ae - bc)e}}{e\sqrt{a + bz} + \sqrt{(ae - bc)e}} \qquad (ae - bc)e > 0$$

$$\int \frac{dz}{(c + ez)\sqrt{a + bz}} = \frac{2}{\sqrt{-(ae - bc)e}} \arctan \frac{e\sqrt{a + bz}}{\sqrt{-(ae - bc)e}} \qquad (ae - bc)e < 0$$

$$\int \frac{z\, dz}{\sqrt{(a + bz)(c + ez)}} = \frac{\sqrt{(a + bz)(c + ez)}}{be} - \frac{ae + bc}{2be} \int \frac{dz}{\sqrt{(a + bz)(c + ez)}}$$

$$\int \frac{dz}{(c + ez)\sqrt{(a + bz)(c + ez)}} = \frac{-2\sqrt{(a + bz)(c + ez)}}{(ae - bc)(c + ez)}$$

$$\int \frac{(c + ez)\,dz}{\sqrt{(a + bz)(c + ez)}} = \frac{\sqrt{(a + bz)(c + ez)}}{b} - \frac{ae - bc}{2b} \int \frac{dz}{\sqrt{(a + bz)(c + ez)}}$$

$$\int \sqrt{\frac{c + ez}{a + bz}}\,dz = \frac{c + ez}{|c + ez|} \int \frac{(c + ez)\,dz}{\sqrt{(a + bz)(c + ez)}}$$

$$\int \frac{(c + ez)^m\,dz}{\sqrt{(a + bz)}} = \frac{2(m!)^2\sqrt{a + bz}}{b(2m + 1)!} \sum_{r=0}^{m} \left(-\frac{4(ae - bc)}{b}\right)^{m-r} \frac{(2r)!}{(r!)^2} (c + ez)^r$$

FORMS CONTAINING $\sqrt{z^2 \pm a^2}$

$$\int \sqrt{z^2 \pm a^2}\,dz = \frac{1}{2}\left[z\sqrt{z^2 \pm a^2} \pm a^2 \ln\left(z + \sqrt{z^2 \pm a^2}\right)\right]$$

$$\int \frac{dz}{\sqrt{z^2 \pm a^2}} = \ln\left(z + \sqrt{z^2 \pm a^2}\right) \qquad \int \frac{dz}{z\sqrt{x^2 - a^2}} = \frac{1}{|a|} \operatorname{arcsec} \frac{z}{a}$$

$$\int \frac{dz}{z\sqrt{z^2 + a^2}} = -\frac{1}{a} \ln\left(\frac{a + \sqrt{z^2 + a^2}}{z}\right)$$

$$\int \frac{\sqrt{z^2 + a^2}}{z}\,dz = \sqrt{z^2 + a^2} - a \ln\left(\frac{a + \sqrt{z^2 + a^2}}{z}\right)$$

$$\int \frac{\sqrt{x^2 - a^2}}{z}\,dz = \sqrt{z^2 - a^2} - |a| \operatorname{arcsec} \frac{z}{a}$$

$$\int \frac{z\,dz}{\sqrt{z^2 \pm a^2}} = \sqrt{z^2 \pm a^2} \qquad \int z\sqrt{x^2 \pm a^2}\,dz = \frac{1}{3}\sqrt{(z^2 \pm a^2)^3}$$

$$\int \sqrt{(z^2 \pm a^2)^3}\,dz = \frac{1}{4}\left[z\sqrt{(z^2 \pm a^2)^3} \pm \frac{3a^2 z}{2}\sqrt{z^2 \pm a^2} + \frac{3a^4}{2}\ln(z + \sqrt{z^2 \pm a^2})\right]$$

$$\int \frac{dz}{\sqrt{(x^2 \pm a^2)^3}} = \frac{\pm z}{a^2\sqrt{z^2 \pm a^2}} \qquad \int \frac{z\,dz}{\sqrt{(z^2 \pm a^2)^3}} = \frac{-1}{\sqrt{z^2 \pm a^2}}$$

$$\int z\sqrt{(z^2 \pm a^2)^3}\,dz = \frac{1}{5}\sqrt{z^2 \pm a^2)^5}$$

$$\int z^2\sqrt{z^2 \pm a^2}\,dz = \frac{z}{4}\sqrt{(z^2 \pm a^2)^3} \mp \frac{a^2}{8} z\sqrt{z^2 \pm a^2} - \frac{a^4}{8}\ln\left(z + \sqrt{a^2 \pm a^2}\right)$$

$$\int z^3\sqrt{z^2 + a^2}\,dz = \left(\frac{1}{5}z^2 - \frac{2}{15}a^2\right)\sqrt{(a^2 + z^2)^3}$$

$$\int z^3\sqrt{z^2 - a^2}\,dz = \frac{1}{5}\sqrt{(z^2 - a^2)^5} + \frac{a^2}{3}\sqrt{(z^2 - a^2)^3}$$

$$\int \frac{z^2\,dz}{\sqrt{(x^2 \pm a^2}} = \frac{z}{2}\sqrt{(z^2 \pm a^2)} \mp \frac{a^2}{2}\ln\left(z + \sqrt{z^2 \pm a^2}\right)$$

$$\int \frac{z^3\,dz}{\sqrt{z^2 \pm a^2}} = \frac{1}{3}\sqrt{(z^2 \pm a^2)^3} \mp a^2\sqrt{z^2 \pm a^2}$$

$$\int \frac{dz}{z^2\sqrt{(x^2 \pm a^2}} = \mp \frac{\sqrt{z^2 \pm a^2}}{a^2 z}$$

$$\int \frac{dz}{z^3\sqrt{z^2 + a^2}} = -\frac{\sqrt{z^2 + a^2}}{2a^2z^2} + \frac{1}{2a^3}\ln\frac{a + \sqrt{z^2 + a^2}}{z}$$

$$\int \frac{dz}{z^3\sqrt{x^2 - a^2}} = \frac{\sqrt{z^2 - a^2}}{2a^2z^2} + \frac{1}{2|a^3|}\operatorname{arcsec}\frac{z}{a}$$

$$\int z^2\sqrt{(z^2 \pm a^2)^3}\,dz = \frac{z}{6}\sqrt{(z^2 \pm a^2)^5} \mp \frac{a^2 z}{24}\sqrt{(z^2 \pm a^2)^3} - \frac{a^4 z}{16}\sqrt{z^2 \pm a^2} \mp \frac{a^6}{16}\ln\left(z + \sqrt{z^2 \pm a^2}\right)$$

$$\int z^3\sqrt{(z^2 \pm a^2)^3}\,dz = \frac{1}{7}\sqrt{(z^2 \pm a^2)^7} \mp \frac{a^2}{5}\sqrt{(z^2 \pm a^2)^5}$$

$$\int \frac{\sqrt{z^2 \pm a^2}\,dz}{z^2} = -\frac{\sqrt{z^2 \pm a^2}}{z} + \ln\left(z + \sqrt{z^2 \pm a^2}\right)$$

$$\int \frac{\sqrt{z^2 + a^2}}{z^3}\,dz = -\frac{\sqrt{z^2 + a^2}}{2z^2} - \frac{1}{2a}\ln\frac{a + \sqrt{z^2 + a^2}}{z}$$

$$\int \frac{\sqrt{z^2 - a^2}}{z^3}\,dz = -\frac{\sqrt{z^2 - a^2}}{2z^2} + \frac{1}{2|a|}\operatorname{arcsec}\frac{z}{a}$$

$$\int \frac{\sqrt{z^2 \pm a^2}}{z^4}\,dz = \mp\frac{\sqrt{(z^2 \pm a^2)^3}}{3a^2z^3}$$

$$\int \frac{z^2\,dz}{\sqrt{(z^2 \pm a^2)^3}} = \frac{-z}{\sqrt{z^2 \pm a^2}} + \ln(z + \sqrt{z^2 \pm a^2})$$

$$\int \frac{z^3\,dz}{\sqrt{(z^2 \pm a^2)^3}} = \sqrt{z^2 \pm a^2} \pm \frac{a^2}{\sqrt{z^2 \pm a^2}}$$

$$\int \frac{dz}{z\sqrt{(z^2 + a^2)^3}} = \frac{1}{a^2\sqrt{z^2 + a^2}} - \frac{1}{a^3}\ln\frac{a + \sqrt{z^2 + a^2}}{z}$$

$$\int \frac{dz}{z\sqrt{(z^2 - a^2)^3}} = \frac{1}{a^2\sqrt{z^2 - a^2}} - \frac{1}{|a^3|}\operatorname{arcsec}\frac{z}{a}$$

$$\int \frac{dz}{z^2\sqrt{(z^2 \pm a^2)^3}} = -\frac{1}{a^4}\left[\frac{\sqrt{z^2 \pm a^2}}{z} + \frac{z}{\sqrt{z^2 \pm a^2}}\right]$$

$$\int \frac{dz}{z^3\sqrt{(z^2+a^2)^3}} = -\frac{1}{2a^2z^2\sqrt{z^2+a^2}} - \frac{3}{2a^4\sqrt{z^2+a^2}} + \frac{3}{2a^5}\ln\frac{a+\sqrt{z^2+a^2}}{z}$$

$$\int \frac{dz}{z^3\sqrt{(z^2-a^2)^3}} = \frac{1}{2a^2z^2\sqrt{z^2-a^2}} - \frac{3}{2a^4\sqrt{z^2-a^2}} - \frac{3}{2|a^5|}\operatorname{arcsec}\frac{z}{a}$$

$$\int \frac{z^m}{\sqrt{z^2\pm a^2}}\,dz = \frac{1}{m}z^{m-1}\sqrt{z^2\pm a^2} \mp \frac{m-1}{m}a^2\int \frac{z^{m-2}}{\sqrt{z^2\pm a^2}}\,dz$$

$$\int \frac{z^{2m}}{\sqrt{z^2\pm a^2}}\,dz = \frac{(2m)!}{2^{2m}(m!)^2}\left[\sqrt{z^2\pm a^2}\sum_{r=1}^{m}\frac{r!(r-1)!}{(2r)!}(\mp a^2)^{m-r}(2z)^{2r-1} + (\mp a^2)^m\ln\left(z+\sqrt{z^2\pm a^2}\right)\right]$$

$$\int \frac{z^{2m+1}}{\sqrt{z^2\pm a^2}}\,dz = \sqrt{z^2\pm a^2}\sum_{r=0}^{m}\frac{(2r)!\,(m!)^2}{(2m+1)!\,(r!)^2}(\mp 4a^2)^{m-r}z^{2r}$$

$$\int \frac{dz}{z^{2m}\sqrt{z^2\pm a^2}} = \sqrt{z^2\pm a^2}\sum_{r=0}^{m-1}\frac{(m-1)!\,m!\,(2r)!\,2^{2m-2r-1}}{(r!)^2(2m)!\,(\mp a^2)^{m-r}z^{2r+1}}$$

$$\int \frac{dz}{z^{2m+1}\sqrt{z^2+a^2}} = \frac{(2m)!}{(m!)^2}\left[\frac{\sqrt{z^2+a^2}}{a^2}\sum_{r=1}^{m}(-1)^{m-r+1}\frac{r!(r-1)!}{2(2r)!(4a^2)^{m-r}z^{2r}} + \frac{(-1)^{m+1}}{2^{2m}2^{2m+1}}\ln\frac{\sqrt{z^2+a^2}+a}{z}\right]$$

$$\int \frac{dz}{z^{2m+1}\sqrt{z^2 - a^2}} = \frac{(2m)!}{(m!)^2}\left[\frac{\sqrt{z^2 - a^2}}{a^2}\sum_{r=1}^{m}\frac{r!(r-1)!}{2(2r)!\,(4a^2)^{m-r}z^{2r}} + \frac{1}{2^{2m}|a|^{2m+1}}\operatorname{arcsec}\frac{z}{a}\right]$$

$$\int \frac{dz}{(z - a)\sqrt{z^2 - a^2}} = -\frac{\sqrt{z^2 - a^2}}{a(z - a)} \qquad \int \frac{dz}{(z + a)\sqrt{z^2 - a^2}} = \frac{\sqrt{z^2 - a^2}}{a(z + a)}$$

$$\int f\left(z, \sqrt{z^2 + a^2}\right) dz = a\int f(a\tan u, a\sec u)\sec^2 u\,du \qquad u = \arctan\frac{z}{a} \qquad a > 0$$

$$\int f\left(z, \sqrt{z^2 - a^2}\right) dz = a\int f(a\sec u, a\tan u)\sec u\tan u\,du \qquad u = \operatorname{arcsec}\frac{z}{a} \qquad a > 0$$

FORMS CONTAINING $\sqrt{a^2 - z^2}$

$$\int \sqrt{a^2 - z^2}\,dz = \frac{1}{2}\left[z\sqrt{a^2 - z^2} + a^2\arcsin\frac{z}{|a|}\right]$$

$$\int \frac{dz}{\sqrt{a^2 - z^2}} = \arcsin\frac{z}{|a|} = -\arccos\frac{z}{|a|}$$

$$\int \frac{dz}{z\sqrt{a^2 - z^2}} = -\frac{1}{a}\ln\left(\frac{a + \sqrt{a^2 - x^2}}{z}\right)$$

$$\int \frac{\sqrt{a^2 - z^2}}{z}\,dz = \sqrt{a^2 - z^2} - a \ln\left(\frac{a + \sqrt{a^2 - z^2}}{z}\right)$$

$$\int \frac{z\,dz}{\sqrt{a^2 - z^2}} = -\sqrt{a^2 - z^2} \qquad \int z\sqrt{a^2 - z^2}\,dz = -\frac{1}{3}\sqrt{(a^2 - z^2)^3}$$

$$\int \sqrt{(a^2 - z^2)^3}\,dz = \frac{1}{4}\left[z\sqrt{(a^2 - x^2)^3} + \frac{3a^2 z}{2}\sqrt{a^2 - z^2} + \frac{3a^4}{2}\arcsin\frac{z}{|a|}\right]$$

$$\int \frac{dz}{\sqrt{(a^2 - z^2)^3}} = \frac{z}{a^2\sqrt{a^2 - z^2}} \qquad \int \frac{z\,dz}{\sqrt{(a^2 - z^2)^3}} = \frac{1}{\sqrt{a^2 - z^2}}$$

$$\int z\sqrt{(a^2 - z^2)^3}\,dz = -\frac{1}{5}\sqrt{(a^2 - z^2)^5}$$

$$\int z^2\sqrt{a^2 - z^2}\,dz = -\frac{z}{4}\sqrt{(a^2 - z^2)^3} + \frac{a^2}{8}\left(z\sqrt{a^2 - z^2} + a^2 \arcsin\frac{z}{|a|}\right)$$

$$\int z^3\sqrt{z^2 - z^2} = \left(-\frac{1}{5}z^2 - \frac{2}{15}a^2\right)\sqrt{(a^2 - z^2)^3}$$

$$\int z^2\sqrt{(a^2 - z^2)^3}\,dz = -\frac{z}{6}\sqrt{(a^2 - z^2)^5} + \frac{a^2 z}{24}\sqrt{(a^2 - z^2)^3} + \frac{a^4 z}{16}\sqrt{a^2 - z^2} + \frac{a^6}{16}\arcsin\frac{z}{|a|}$$

$$\int z^3\sqrt{(a^2 - z^2)^3}\, dz = \frac{1}{7}\sqrt{(a^2 - z^2)^7} - \frac{a^2}{5}\sqrt{(a^2 - x^2)^5}$$

$$\int \frac{z^2\, dz}{\sqrt{a^2 - z^2}} = -\frac{z}{2}\sqrt{a^2 - z^2} + \frac{a^2}{2}\arcsin\frac{z}{|a|}$$

$$\int \frac{dz}{z^2\sqrt{z^2 - z^2}} = -\frac{\sqrt{(a^2 - z^2)}}{a^2 z}$$

$$\int \frac{\sqrt{a^2 - z^2}}{z^2}\, dz = -\frac{\sqrt{a^2 - z^2}}{z} - \arcsin\frac{z}{|a|}$$

$$\int \frac{\sqrt{a^2 - z^2}}{z^3}\, dz = \frac{\sqrt{a^2 - z^2}}{2x^2} + \frac{1}{2a}\ln\frac{a + \sqrt{a^2 - z^2}}{z}$$

$$\int \frac{\sqrt{a^2 - z^2}}{z^4}\, dz = -\frac{\sqrt{(a^2 - x^2)^3}}{3a^2 z^3}$$

$$\int \frac{z^2 dz}{\sqrt{(a^2 - z^2)^3}} = \frac{z}{\sqrt{a^2 - z^2}} - \arcsin\frac{z}{|a|}$$

$$\int \frac{z^3\, dz}{\sqrt{a^2 - z^2}} = -\frac{2}{3}\sqrt{a^2 - z^2} - z^2\sqrt{a^2 - z^2} = -\frac{1}{3}\sqrt{a^2 - z^2}\,(z^2 + 2a^2)$$

$$\int \frac{z^3\,dz}{\sqrt{(a^2 - z^2)^3}} = 2\sqrt{a^2 - z^2} + \frac{z^2}{\sqrt{a^2 - z^2}} = -\frac{a^2}{\sqrt{a^2 - z^2}} + \sqrt{a^2 - z^2}$$

$$\int \frac{dz}{z^3\sqrt{a^2 - z^2}} = -\frac{\sqrt{a^2 - x^2}}{2a^2z^2} - \frac{1}{2a^3}\ln\frac{a + \sqrt{a^2 - z^2}}{z}$$

$$\int \frac{dz}{z\sqrt{(a^2 - z^2)^3}} = \frac{1}{a^2\sqrt{a^2 - z^2}} - \frac{1}{a^3}\ln\frac{a + \sqrt{a^2 - z^2}}{z}$$

$$\int \frac{dz}{z^2\sqrt{(a^2 - z^2)^3}} = \frac{1}{a^4}\left[-\frac{\sqrt{a^2 - z^2}}{z} + \frac{z}{\sqrt{a^2 - z^2}}\right]$$

$$\int \frac{dz}{z^3\sqrt{(a^2 - z^2)^3}} = \frac{1}{2a^2z^2\sqrt{a^2 - z^2}} + \frac{3}{2a^4\sqrt{a^2 - z^2}} - \frac{3}{2a^5}\ln\frac{a + \sqrt{a^2 - z^2}}{z}$$

$$\int \frac{z^m}{\sqrt{a^2 - z^2}}\,dz = -\frac{z^{m-1}\sqrt{a^2 - x^2}}{m} + \frac{(m-1)a^2}{m}\int \frac{z^{m-2}}{\sqrt{a^2 - z^2}}\,dz$$

$$\int \frac{z^{2m}}{\sqrt{a^2 - z^2}}\,dz = \frac{(2m)!}{(m!)^2}\left[-\sqrt{a^2 - z^2}\sum_{r=1}^{m}\frac{r!(r-1)!}{2^{2m-2r+1}(2r)!}a^{2m-2r}z^{2r-1} + \frac{a^{2m}}{2^{2m}}\arcsin\frac{x}{|a|}\right]$$

$$\int \frac{z^{2m+1}}{\sqrt{a^2 - z^2}}\,dz = -\sqrt{a^2 - z^2}\sum_{r=0}^{m}\frac{(2r)!(m!)^2}{(2m+1!)(r!)^2}(4a^2)^{m-r}z^{2r}$$

$$\int \frac{dz}{z^m\sqrt{a^2 - z^2}} = -\frac{\sqrt{a^2 - z^2}}{(m-1)a^2 z^{m-1}} + \frac{m-2}{(m-1)a^2}\int \frac{dz}{z^{m-2}\sqrt{a^2 - z^2}}$$

$$\int \frac{dz}{z^{2m}\sqrt{a^2 - z^2}} = -\sqrt{a^2 - x^2}\sum_{r=0}^{m-1} \frac{(m-1)!\,m!\,(2r)!\,2^{2m-2r-1}}{(r!)^2(2m)!\,a^{2m-2r}z^{2r+1}}$$

$$\int \frac{dz}{z^{2m+1}\sqrt{a^2 - z^2}} = \frac{(2m)!}{(m!)^2}\left[-\frac{\sqrt{a^2 - z^2}}{a^2}\sum_{r=1}^{m} \frac{r!\,(r-1)!}{2(2r)!\,(4a^2)^{m-r}z^{2r}} + \frac{1}{2^{2m}a^{2m+1}}\ln\frac{a - \sqrt{a^2 - x^2}}{z}\right]$$

$$\int \frac{dz}{(b^2 - z^2)\sqrt{a^2 - z^2}} = \frac{1}{2b\sqrt{a^2 - b^2}}\ln\frac{\left(b\sqrt{a^2 - z^2} + z\sqrt{a^2 - b^2}\right)^2}{b^2 - z^2} \qquad a^2 > b^2$$

$$\int \frac{dz}{(b^2 - z^2)\sqrt{a^2 - z^2}} = \frac{1}{b\sqrt{b^2 - a^2}}\arctan\frac{z\sqrt{b^2 - a^2}}{b\sqrt{a^2 - z^2}} \qquad b^2 > a^2$$

$$\int \frac{dz}{(b^2 + x^2)\sqrt{a^2 - z^2}} = \frac{1}{b\sqrt{a^2 + b^2}}\arctan\frac{z\sqrt{a^2 + b^2}}{b\sqrt{a^2 - z^2}}$$

$$\int \frac{\sqrt{a^2 - z^2}}{b^2 + z^2}\,dz = \frac{\sqrt{a^2 + b^2}}{|b|}\arcsin\frac{z\sqrt{a^2 + b^2}}{|a|\sqrt{z^2 + b^2}} - \arcsin\frac{z}{|a|}$$

$$\int f\left(z\,,\,\sqrt{(a^2 - z^2)}\right) dz = a \int f\,(a \sin u,\, a \cos u) \cos u\, du \qquad u = \arcsin \frac{z}{a} \qquad a > 0$$

FORMS CONTAINING $\sqrt{a + bz + cz^2}$

$$Z = a + bz + cz^2 \qquad q = 4ac - b^2 \qquad k = \frac{4c}{q}$$

$$\int \frac{dz}{\sqrt{Z}} = \frac{1}{\sqrt{c}} \arcsin \frac{2cz + b}{\sqrt{q}} \qquad c > 0$$

$$\int \frac{dz}{\sqrt{Z}} = -\frac{1}{\sqrt{-c}} \arcsin \frac{2cz + b}{\sqrt{-q}} \qquad c < 0$$

$$\int \frac{dz}{Z\sqrt{Z}} = \frac{2(2cz + b)}{q\sqrt{Z}}$$

$$\int \frac{dz}{Z^2\sqrt{Z}} = \frac{2(2cz + b)}{3q\sqrt{Z}}\left(\frac{1}{Z} + 2k\right)$$

$$\int \frac{dz}{Z^n \sqrt{Z}} = \frac{(2cz + b)(n!)(n - 1)!4^n k^{n-1}}{q[(2n)!]\sqrt{Z}} \sum_{r=0}^{n-1} \frac{(2r)!}{(4kZ)^r (r!)^2}$$

$$\int \sqrt{Z}\, dz = \frac{(2cz + b)\sqrt{Z}}{4c} + \frac{1}{2k} \int \frac{dz}{\sqrt{Z}}$$

$$\int Z\sqrt{Z}\, dz = \frac{(2cz + b)\sqrt{Z}}{8c}\left(Z + \frac{3}{2k}\right) + \frac{3}{8k^2} \int \frac{dz}{\sqrt{Z}}$$

$$\int Z^2 \sqrt{Z}\, dz = \frac{(2cz + b)\sqrt{Z}}{12c}\left(Z^2 + \frac{5Z}{4k} + \frac{15}{8k^2}\right) + \frac{5}{16k^3} \int \frac{dz}{\sqrt{Z}}$$

$$\int Z^n \sqrt{Z}\, dz = \frac{(2n+2)!}{[(n+1)!]^2 (4k)^{n+1}} \left[\frac{k(2cz + b)\sqrt{Z}}{c} \sum_{r=0}^{n} \frac{r!\,(r+1)!\,(4kZ)^r}{(2r+2)!} + \int \frac{dz}{\sqrt{Z}} \right]$$

$$\int \frac{z\, dz}{\sqrt{Z}} = \frac{\sqrt{Z}}{c} - \frac{b}{2c} \int \frac{dz}{\sqrt{Z}}$$

$$\int \frac{z\, dz}{Z\sqrt{Z}} = -\frac{2(bz + 2a)}{q\sqrt{Z}}$$

$$\int \frac{z\, dz}{Z^n \sqrt{Z}} = \frac{\sqrt{Z}}{(2n - 1)cZ^n} - \frac{b}{2c} \int \frac{dz}{Z^n \sqrt{Z}}$$

$$\int \frac{z^2\, dz}{\sqrt{Z}} = \left(\frac{z}{2c} - \frac{3b}{4c^2} \right) \sqrt{Z} + \frac{3b^2 - 4ac}{8c^2} \int \frac{dz}{\sqrt{Z}}$$

$$\int \frac{z^2\, dz}{Z\sqrt{Z}} = \frac{(2b^2 - 4ac)z + 2ab}{cq\sqrt{Z}} + \frac{1}{c} \int \frac{dz}{\sqrt{Z}}$$

$$\int \frac{z^2\, dz}{Z^n \sqrt{Z}} = \frac{(2b^2 - 4ac)z + 2ab}{(2n - 1)cqZ^{n-1}\sqrt{Z}} + \frac{4ac + (2n - 3)b^2}{(2n - 1)cq} \int \frac{dz}{Z^{n-1}\sqrt{Z}}$$

$$\int \frac{z^3\, dz}{\sqrt{Z}} = \left(\frac{x^2}{3c} - \frac{5bz}{12c^2} + \frac{5b^2}{8c^3} - \frac{2a}{3c^2} \right) \sqrt{Z} + \left(\frac{3ab}{4c^2} - \frac{5b^3}{16c^3} \right) \int \frac{dz}{\sqrt{Z}}$$

$$\int z\sqrt{Z}\,dz = \frac{Z\sqrt{Z}}{3c} - \frac{b(2cz + b)}{8c^2}\sqrt{Z} - \frac{b}{4ck}\int \frac{dz}{\sqrt{Z}}$$

$$\int z\,Z\sqrt{Z}\,dz = \frac{Z^2\sqrt{Z}}{5c} - \frac{b}{2c}\int Z\sqrt{Z}\,dz$$

$$\int z^2\sqrt{Z}\,dz = \left(z - \frac{5b}{6c}\right) + \frac{Z\sqrt{Z}}{4c} + \frac{5b^2 - 4ac}{16c^2}\int \sqrt{Z}\,dz$$

$$\int \frac{dz}{z\sqrt{Z}} = -\frac{1}{\sqrt{a}} \ln \frac{2\sqrt{aZ} + bz + 2a}{z} \qquad a > 0$$

$$\int \frac{dz}{z\sqrt{Z}} = \frac{1}{\sqrt{-a}} \arcsin\left(\frac{bz + 2a}{|z|\sqrt{-q}}\right) \qquad a < 0$$

$$\int \frac{dz}{z\sqrt{Z}} = -\frac{2\sqrt{Z}}{bz} \qquad a = 0$$

FORMS INVOLVING $\sqrt{2az - z^2}$

$$\int \sqrt{2az - z^2}\,dz = \frac{1}{2}\left[(z - a)\sqrt{2az - z^2} + a^2 \arcsin \frac{z - a}{|a|}\right]$$

$$\int \frac{dz}{\sqrt{2az - z^2}} = \arccos \frac{a - z}{|a|} = \arcsin \frac{z - a}{|a|}$$

$$\int z^n \sqrt{2az - z^2}\, dz = \sqrt{2az - z^2}\left[\frac{z^{n+1}}{n+2} - \sum_{r=0}^{n} \frac{(2n+1)!\,(r!)^2 a^{n-r+1}}{2^{n-r}(2r+1)!\,(n+2)!\,n!}\, z^r\right] + \frac{(2n+1)!\,a^{n+2}}{2^n n!\,(n+2)!} \arcsin\frac{z-a}{|a|}$$

$$\int \frac{\sqrt{2az - z^2}}{z^n}\, dz = \frac{(2az - z^2)^{\frac{3}{2}}}{(3-2n)az^n} + \frac{n-3}{(2n-3)a} \int \frac{\sqrt{2az - z^2}}{z^{n-1}} dz$$

$$\int \frac{z^n\, dz}{\sqrt{2az - z^2}} = -\sqrt{2az - z^2} \sum_{r=1}^{n} \frac{(2n)!\,r!\,(r-1)!\,a^{n-r}}{2^{n-r}(2r)!\,(n!)^2}\, z^{r-1} + \frac{(2n)!\,a^n}{2^n (n!)^2} \arcsin\frac{z-a}{|a|}$$

$$\int \frac{dz}{z^n\sqrt{2az - z^2}} = -\sqrt{2az - z^2} \sum_{r=0}^{n-1} \frac{2^{n-r}(n-1)!\,n!\,(2r)!}{(2n)!\,(r!)^2 a^{n-r} z^{r+1}}$$

$$\int \frac{dz}{(2az - z^2)^{\frac{3}{2}}} = \frac{z-a}{a^2\sqrt{2az - z^2}} \qquad \int \frac{z\, dz}{(2az - z^2)^{\frac{3}{2}}} = \frac{z}{a\sqrt{2az - z^2}}$$

MISCELLANEOUS ALGEBRAIC FORMS

$$\int \frac{dz}{\sqrt{2az + z^2}} = \ln\left(z + a + \sqrt{2az + z^2}\right)$$

$$\int \sqrt{az^2 + c}\, dz = \frac{z}{2}\sqrt{az^2 + c} + \frac{c}{2\sqrt{a}} \ln\left(z\sqrt{a} + \sqrt{az^2 + c}\right) \qquad a > 0$$

$$\int \sqrt{az^2 + c}\, dz = \frac{z}{2}\sqrt{az^2 + c} + \frac{c}{2\sqrt{-a}} \arcsin\left(z\sqrt{-\frac{a}{c}}\right) \qquad a < 0$$

$$\int \sqrt{\frac{1+z}{1-z}}\, dz = \arcsin z - \sqrt{1 - z^2}$$

$$\int \frac{dz}{z\sqrt{az^n + c}} = \frac{2}{n\sqrt{c}} \ln \frac{\sqrt{az^n + c} - \sqrt{c}}{\sqrt{z^n}} \qquad c > 0$$

$$\int \frac{dz}{z\sqrt{az^n + c}} = \frac{2}{n\sqrt{-c}} \operatorname{arcsec} \sqrt{-\frac{az^n}{c}} \qquad c < 0$$

$$\int \frac{dz}{\sqrt{az^2 + c}} = \frac{1}{\sqrt{a}} \ln\left(z\sqrt{a} + \sqrt{az^2 + c}\right) \qquad a > 0$$

$$\int \frac{dz}{\sqrt{az^2 + c}} = \frac{1}{\sqrt{-a}} \arcsin\left(z\sqrt{-\frac{a}{c}}\right) \qquad a < 0$$

$$\int \frac{dz}{(az^2 + c)^{m+\frac{1}{2}}} = \frac{z}{\sqrt{az^2 + c}} \sum_{r=0}^{m-1} \frac{2^{2m-2r-1}(m-1)!\, m!\, (2r)!}{(2m)!\, (r!)^2 c^{m-r} (az^2 + c)^r}$$

$$\int z(az^2 + c)^{m+\frac{1}{2}}\, dz = \frac{(az^2 + c)^{m+\frac{1}{2}}}{(2m+3)a}$$

$$\int \frac{(1 + z^2)\,dz}{(1 - z^2)\sqrt{1 + z^4}} = \frac{1}{\sqrt{2}} \ln \frac{z\sqrt{2} + \sqrt{1 + z^4}}{1 - z^2}$$

$$\int \frac{(1 - z^2)\,dz}{(1 - z^2)\sqrt{1 + z^4}} = \frac{1}{\sqrt{2}} \arctan \frac{z\sqrt{2}}{\sqrt{1 + z^4}}$$

$$\int \frac{dz}{z\sqrt{z^n + a^2}} = -\frac{2}{na} \ln \frac{a + \sqrt{z^n + a^2}}{\sqrt{z^n}}$$

$$\int \frac{dz}{z\sqrt{z^n - a^2}} = -\frac{2}{na} \arcsin \frac{a}{\sqrt{z^n}}$$

$$\int \sqrt{\frac{z}{a^3 - z^3}}\,dz = \frac{2}{3} \arcsin \left(\frac{z}{a}\right)^{\frac{3}{2}}$$

COMPLEX EXTENSION OF TABLES

In the world of real numbers, ln(x) and √(x) are not defined for negative x. In the world of complex numbers, ln(z) and √(z) are well defined everywhere except the negative real axis where the branch cut is. See the tutorial in Chapter 3. The conditions on certain integrals given above should be interpreted as to stay away from any branch cuts. Otherwise, all the above integrals apply to complex integration over analytic paths which do not encircle any poles of the integrand. Far more extensive integral tables are given in:

Gradshteyn, I., and Ryzhik, I., *Table of Integrals, Series, and Products*, New York, Academic Press, 1980.

APPENDIX E

MISCELLANEOUS SERIES

INTRODUCTION

This appendix gives a brief overview of approximation theory and also provides a short collection of finite and infinite series.

WEIERSTRASS APPROXIMATION THEOREM

The reason MATHLIB has so many commands which deal with polynomials is the Weierstrass approximation theorems, which state that if real function f(x) is continuous on the closed interval [a, b], then for every $\varepsilon > 0$, there exists

1. a real polynomial $P(x) = \sum_{k=0}^{N} a_k x^k$ such that $|f(x) - P(x)| < \varepsilon$, and

2. a real trigonometric polynomial $T(x) = \sum_{k=0}^{M} [\alpha_k \cos k\omega x + \beta_k \sin k\omega x]$ such that

$$|f(x) - T(x)| < \varepsilon$$

for all $x \in [a, b]$. The commands **PMAT** and **FMAT** allow polynomials to be expressed in terms of other polynomials. If f(x) is a rational function (the ratio of two polynomials), then **PLDVD** can be used to approximate it with a polynomial. If f(x) is infinitely differentiable and the series converges, then f(x) can be represented by its Taylor series in the region of convergence as

$$f(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(a)}{k!} (x - a)^k.$$

TALR1 and **TALR2** provide one- and two-dimensional Taylor series expansions. When a = 0, these expansions are called Maclaurin series expansions. **COEFL** provides one-dimensional Maclaurin series expansions. The residue integration commands **RESDP** and **RESDA** allow evaluation of Laurent expansions. Calculation of trigonometric polynomial (Fourier) approximations is discussed in Appendices F and G.

COMMON FINITE SERIES

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2} \qquad \sum_{k=1}^{n} k^2 = \frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6}$$

$$\sum_{k=1}^{n} k^3 = \left[\frac{n(n+1)}{2}\right]^2 \qquad \sum_{k=1}^{n} k^4 = \frac{n^5}{5} + \frac{n^4}{2} + \frac{n^3}{3} - \frac{n}{30}$$

$$\sum_{k=1}^{m} k^n = \frac{B_{n+1}(m+1) - B_{n+1}(0)}{n+1}$$

$$\sum_{k=1}^{m} (-1)^{m-k} k^n = \frac{E_n(m+1) + (-1)^m E_n(0)}{2} \qquad m, n \in \mathbf{N}$$

$$\sum_{k=0}^{n} \binom{n}{k} x^k = (1+x)^n \quad n = 0, 1, \ldots \qquad \sum_{k=n}^{m} \binom{k}{n} = \binom{m+1}{n+1} \quad m > n$$

$$\sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k} = (a+b)^n \quad n = 0, 1, \ldots \qquad \sum_{k=0}^{n} \binom{r}{k}\binom{s}{n-k} = \binom{r+s}{n} \quad r \geq n \quad s \geq n$$

$$\sum_{k=0}^{n} \binom{r}{k}\binom{s}{n+k} = \binom{r+s}{s-n} \quad r \geq n \quad s \geq n$$

$$\sum_{k=0}^{n} \binom{r}{k}\binom{n}{k} = \binom{r+n}{n} \quad r \geq n$$

$$\sum_{k=1}^{n} \binom{2n}{2k} = 2^{2n-1} - 1$$

$$\sum_{k=1}^{m} \binom{2n-1}{2k-1} = 2^{2n-2}$$

$$\sum_{k=1}^{n} k\binom{n}{k} = n\,2^{2n-1}$$

$$\sum_{k=1}^{n} k^2\binom{n}{k} = n\,(n+1)\,2^{2n+1}$$

$$\sum_{k=0}^{n} \binom{n}{k}^2 = \binom{2n}{n}$$

$$\sum_{k=1}^{n} k\binom{n}{k}^2 = \frac{n}{2}\binom{2n}{n}$$

$$\sum_{k=1}^{n} k^2\binom{n}{k}^2 = \frac{n}{2}(n+1)\binom{2n}{n}$$

$$\sum_{k=1}^{n} (-1)^{k+1}\binom{n}{k} = 1$$

$$\sum_{k=1}^{n} (-1)^{k+1} k\binom{n}{k} = 0$$

$$\sum_{k=1}^{n} (-1)^{k+1} k^n\binom{n}{k} = (-1)^{n+1}\,n!$$

$$\sum_{k=0}^{n} (-1)^k\binom{2n}{k}^2 = (-1)^n\binom{2n}{n}$$

$$\sum_{k=0}^{2n+1} (-1)^k\binom{2n+1}{k}^2 = 0$$

$$\sum_{k=0}^{n-1} z^k = \frac{z^n - 1}{z - 1} \quad z \neq 1$$

$$\sum_{k=0}^{n-1} k\,z^k = \frac{z(1 - z^n)}{(1 - z)^2} - \frac{nz^n}{1 - z} \quad z \neq 1$$

$$\sum_{k=1}^{n} \sin k\theta = \frac{\sin[(n+1)\theta/2] \sin n\theta/2}{\sin \theta/2} \qquad \sum_{k=1}^{n} \cos k\theta = \frac{\cos[(n+1)\theta/2] \sin n\theta/2}{\sin \theta/2}$$

$$\sum_{k=1}^{n} \sin[(2k-1)\theta] = \frac{\sin^2 n\theta}{\sin \theta} \qquad \sum_{k=1}^{n} \cos[(2k-1)\theta] = \frac{\sin 2n\theta}{2 \sin \theta}$$

$$\sum_{k=1}^{n-1} k \sin k\theta = \frac{\sin n\theta}{4 \sin^2 \theta/2} - \frac{n \cos[(2n-1)\theta/2]}{2 \sin \theta/2}$$

$$\sum_{k=1}^{n-1} k \cos k\theta = \frac{n \sin[(2n-1)\theta/2]}{2 \sin \theta/2} - \frac{1 - \cos n\theta}{4 \sin^2 \theta/2}$$

$$\sum_{k=1}^{n+1} (-1)^{k-1} \sin[(2k-1)\theta] = (-1)^n \frac{\sin[(2n+2)\theta]}{2 \cos \theta}$$

$$\sum_{k=1}^{n} (-1)^k \cos k\theta = -\frac{1}{2} + (-1)^n \frac{\cos[(2n+1)\theta/2]}{2 \cos \theta/2}$$

$$\sum_{k=1}^{n} \sin^2 k\theta = \frac{n}{2} - \frac{\cos(n+1)\theta \sin n\theta}{2 \sin \theta} \qquad \sum_{k=1}^{n} \cos^2 k\theta = \frac{n}{2} + \frac{\cos(n+1)\theta \sin n\theta}{2 \sin \theta}$$

$$\sum_{k=1}^{n-1} a^k \sin k\theta = \frac{a \sin \theta [1 - a^n \cos n\theta] - [1 - a \cos \theta] a^n \sin n\theta}{1 - 2a \cos \theta + a^2}$$

$$\sum_{k=1}^{n-1} a^k \cos k\theta = \frac{1 - a\cos\theta + a^{n+1}\cos(n-1)\theta - a^n \cos n\theta}{1 - 2a\cos\theta + a^2}$$

COMMON INFINITE SERIES

$$e^z = \exp z = \mathrm{EXP}\, z = 1 + \frac{z}{1!} + \frac{z^2}{2!} + \frac{z^3}{3!} + \ldots$$

$$\ln z = \left(\frac{z-1}{z}\right) + \frac{1}{2}\left(\frac{z-1}{z}\right)^2 + \frac{1}{3}\left(\frac{z-1}{z}\right)^3 + \ldots \qquad \mathrm{RE}\, z \geq \tfrac{1}{2}$$

$$\ln z = (z-1) - \tfrac{1}{2}(z-1)^2 + \tfrac{1}{3}(z-1)^3 - \ldots \qquad |z-1| \leq 1 \quad z \neq 0$$

$$\ln z = 2\left[\left(\frac{z-1}{z+1}\right) + \frac{1}{3}\left(\frac{z-1}{z+1}\right)^3 + \frac{1}{5}\left(\frac{z-1}{z+1}\right)^5 + \ldots\right] \qquad \mathrm{RE}\, z \geq 0 \quad z \neq 0$$

$$\ln(z+a) = \ln a + 2\left[\left(\frac{z}{z+2a}\right) + \frac{1}{3}\left(\frac{z}{z+2a}\right)^3 + \frac{1}{5}\left(\frac{z}{z+2a}\right)^5 + \ldots\right] \quad \begin{matrix}\mathrm{RE}\, z \geq -a \neq z \\ a > 0\end{matrix}$$

$$\sin z = z - \frac{z^3}{3!} + \frac{z^5}{5!} - \frac{z^7}{7!} + \ldots$$

$$\cos z = 1 - \frac{z^2}{2!} + \frac{z^4}{4!} - \frac{z^6}{6!} + \ldots$$

$$\tan z = z + \frac{z^3}{3} + \frac{2z^5}{15} + \frac{17z^7}{315} + \ldots + \frac{(-1)^{n-1}2^{2n}(2^{2n}-1)B_{2n}}{(2n)!}z^{2n-1} + \ldots \qquad |z| < \frac{\pi}{2}$$

$$\cot z = \frac{1}{z} - \frac{z}{3} - \frac{z^3}{45} - \frac{2z^5}{945} - \ldots - \frac{(-1)^{n-1}2^{2n}B_{2n}}{(2n)!}z^{2n-1} - \ldots \qquad |z| < \pi$$

$$\csc z = \frac{1}{z} + \frac{z}{6} + \frac{7z^3}{360} + \frac{31z^5}{15120} + \ldots + \frac{(-1)^{n-1}2(2^{2n-1}-1)B_{2n}}{(2n)!}z^{2n-1} + \ldots \qquad |z| < \pi$$

$$\sec z = 1 + \frac{z^2}{2} + \frac{5z^4}{24} + \frac{61z^6}{720} + \ldots + \frac{(-1)^n E_{2n}}{(2n)!}z^{2n} + \ldots \qquad |z| < \frac{\pi}{2}$$

$$\ln \frac{\sin z}{z} = \sum_{n=1}^{\infty} \frac{(-1)^n 2^{2n-1}B_{2n}}{n(2n)!} z^{2n} \qquad |z| < \pi$$

$$\ln \cos z = \sum_{n=1}^{\infty} \frac{(-1)^n 2^{2n-1}(2^{2n}-1)B_{2n}}{n(2n)!} z^{2n} \qquad |z| < \frac{\pi}{2}$$

$$\ln \frac{\tan z}{z} = \sum_{n=1}^{\infty} \frac{(-1)^{n-1} 2^{2n}(2^{2n-1}-1)B_{2n}}{n(2n)!} z^{2n} \qquad |z| < \frac{\pi}{2}$$

$$\sinh z = z + \frac{z^3}{3!} + \frac{z^5}{5!} + \frac{z^7}{7!} + \ldots$$

$$\cosh z = 1 + \frac{z^2}{2!} + \frac{z^4}{4!} + \frac{z^6}{6!} + \ldots$$

$$\tanh z = z - \frac{z^3}{3} + \frac{2z^5}{15} - \frac{17z^7}{315} + \ldots + \frac{2^{2n}(2^{2n}-1)B_{2n}}{(2n)!}z^{2n-1} + \ldots \qquad |z| < \frac{\pi}{2}$$

$$\coth z = \frac{1}{z} + \frac{z}{3} - \frac{z^3}{45} + \frac{2z^5}{945} - \ldots + \frac{2^{2n}B_{2n}}{(2n)!}z^{2n-1} - \ldots \qquad |z| < \pi$$

$$\operatorname{csch} z = \frac{1}{z} - \frac{z}{6} + \frac{7z^3}{360} - \frac{31z^5}{15120} + \ldots - \frac{2(2^{2n-1}-1)B_{2n}}{(2n)!}z^{2n-1} + \ldots \qquad |z| < \pi$$

$$\operatorname{sech} z = 1 - \frac{z^2}{2} + \frac{5z^4}{24} - \frac{61z^6}{720} + \ldots + \frac{E_{2n}}{(2n)!}z^{2n} + \ldots \qquad |z| < \frac{\pi}{2}$$

$$e^{\sin z} = 1 + z + \frac{z^2}{2!} - \frac{3z^4}{4!} - \frac{8z^5}{5!} - \frac{3z^6}{6!} + \frac{56z^7}{7!} + \ldots$$

$$e^{\cos z} = e\left(1 - \frac{z^2}{2!} + \frac{4z^4}{4!} - \frac{31z^6}{6!} + \ldots\right)$$

$$e^{\tan z} = 1 + z + \frac{z^2}{2!} + \frac{3z^3}{3!} + \frac{9z^4}{4!} + \frac{37z^5}{5!} + \ldots$$

$$\arcsin z = z + \frac{z^3}{2\cdot 3} + \frac{1\cdot 3z^5}{2\cdot 4\cdot 5} + \frac{1\cdot 3\cdot 5z^7}{2\cdot 4\cdot 6\cdot 7} + \ldots \qquad |z| < 1$$

$$\arctan z = z - \frac{z^3}{3} + \frac{z^5}{5} - \frac{z^7}{7} + \ldots \qquad |z| \leq 1 \text{ and } z^2 \neq -1$$

$$\arctan z = \frac{\pi}{2} - \frac{1}{z} + \frac{1}{3z^3} - \frac{1}{5z^5} + \cdots \qquad |z| > 1 \text{ and } z^2 \neq -1$$

$$E_1(z) = -\gamma - \ln z - \sum_{n=1}^{\infty} \frac{(-z)^n}{n\, n!} \qquad |\arg z| < \pi$$

$$\operatorname{erf} z = \frac{2}{\sqrt{\pi}} \sum_{n=0}^{\infty} \frac{(-1)^n z^{2n+1}}{n!\,(2n+1)} \qquad C(z) + i\,S(z) = z\,M\left(\tfrac{1}{2}, \tfrac{3}{2}, \tfrac{i\pi z^2}{2}\right)$$

$$J_\nu(z) = \left(\frac{z}{2}\right)^\nu \sum_{k=0}^{\infty} \frac{(-z^2/4)^k}{k!\,\Gamma(\nu+k+1)} \qquad I_\nu(z) = \left(\frac{z}{2}\right)^\nu \sum_{k=0}^{\infty} \frac{(z^2/4)^k}{k!\,\Gamma(\nu+k+1)}$$

$$K = \tfrac{\pi}{2} F\left(\tfrac{1}{2}, \tfrac{1}{2}, 1, m\right) \qquad E = \tfrac{\pi}{2} F\left(-\tfrac{1}{2}, \tfrac{1}{2}, 1, m\right)$$

$$Y_n(z) = -\frac{(z/2)^{-n}}{\pi} \sum_{k=0}^{n-1} \frac{(n-k-1)!}{k!} \left(\frac{z^2}{4}\right)^k + \frac{2}{\pi} \ln(z/2)\, J_n(z) - \frac{(z/2)^n}{\pi} \sum_{k=0}^{\infty} [\psi(k+1) + \psi(n+k+1)] \frac{(-z^2/4)^k}{k!\,(n+k)!}$$

$$K_n(z) = \frac{(z/2)^{-n}}{2} \sum_{k=0}^{n-1} \frac{(n-k-1)!}{k!} \left(-\frac{z^2}{4}\right)^k - (-1)^n \ln(z/2)\, I_n(z) + \frac{(-z/2)^n}{2} \sum_{k=0}^{\infty} [\psi(k+1) + \psi(n+k+1)] \frac{(z^2/4)^k}{k!\,(n+k)!}$$

$$(1+z)^\alpha = 1 + \alpha z + \frac{\alpha(\alpha-1)z^2}{2!} + \frac{\alpha(\alpha-1)(\alpha-2)z^3}{3!} + \cdots$$

APPENDIX F

CONTINUOUS TIME TRANSFORMS

INTRODUCTION

This appendix presents the basic concepts and formulas associated with Laplace, Fourier, and Hilbert transforms. We leave the rigorous details of convergence to the references. However, we do overview distribution theory since it is a key concept in the practical application of transforms to engineering problems. Many of the transform integrals only exist in a *Cauchy principal value* sense, which is defined on page 44 of Chapter 4. For an exhaustive list of transform pairs, see Erdelyi.

DISTRIBUTION FUNCTIONS

The central function in distribution theory is the *Dirac delta function*, $\delta(t)$. This function has no pointwise defined values other than to say that $\delta(t) = 0$ for $t \neq 0$, and $\delta(t) = \infty$ for $t = 0$. Its real definition is as a limit in the mean, which we will shortly define. The sequence of functions which converge in the mean to $\delta(t)$ is not unique. The *Riemann–Lebesgue lemma* states that if $\phi(t)$ is absolutely integrable in (a, b), then

$$\lim_{\omega \to \infty} \int_a^b e^{-i\omega t} \phi(t)\, dt = 0$$

where a and b are finite or infinite constants. In the sense of this integral, we say that the *limit in the mean* (LIM) of $e^{-i\omega t} = 0$ as $\omega \to \infty$. Since $e^{-i\omega t} = \cos \omega t - i \sin \omega t$, we also have the generalized limits or limits in the mean:

$$\underset{\omega \to \infty}{\text{LIM}} \sin \omega t = 0, \qquad \underset{\omega \to \infty}{\text{LIM}} \cos \omega t = 0.$$

Let $\phi(t)$ be continuous at the origin, and consider the following integral:

$$\int_{-\infty}^{\infty} \frac{\sin \omega t}{\pi t} \phi(t)\, dt = \int_{-\infty}^{-\varepsilon} + \int_{-\varepsilon}^{\varepsilon} + \int_{\varepsilon}^{\infty} \frac{\sin \omega t}{\pi t} \phi(t)\, dt.$$

Since $\phi(t)/t$ is integrable in the $(-\infty, -\varepsilon)$ and (ε, ∞) intervals, it follows from the Riemann–Lebesgue lemma that as $\omega \to \infty$, the first and last integrals go to zero. Also, for sufficiently small $\varepsilon > 0$:

$$\int_{-\varepsilon}^{\varepsilon} \frac{\sin \omega t}{\pi t} \phi(t)\, dt \sim \phi(0) \int_{-\varepsilon}^{\varepsilon} \frac{\sin \omega t}{\pi t}\, dt = \phi(0) \int_{-\omega\varepsilon}^{\omega\varepsilon} \frac{\sin x}{\pi x}\, dx,$$

and since

$$\int_{-\infty}^{\infty} \frac{\sin x}{\pi x}\, dx = 1$$

we see that

$$\lim_{\omega \to \infty} \int_{-\infty}^{\infty} \frac{\sin \omega t}{\pi t} \phi(t)\, dt = \phi(0).$$

We define any function $g(\omega, t)$ with the property

$$\lim_{\omega \to \infty} \int_{a}^{b} g(\omega, t)\, \phi(t)\, dt = \phi(0),$$

where (a, b) contains the origin, to be a Dirac delta function. For example,

$$\underset{\omega \to \infty}{\text{LIM}} \frac{\sin \omega t}{\pi t} = \delta(t).$$

The integral of $\delta(t)$ is the unit step function $u(t)$.

$$u(t) = \int_{-\infty}^{t} \delta(\tau)\, d\tau = \begin{cases} 1 & t > 0 \\ 0 & t < 0 \end{cases}$$

Clearly, $u(0)$ could be reasonably defined to be any number in $[0, 1]$. Defining it to equal ½ generally works provided integrals over $(0, \infty)$ are evaluated as if they were over $(-\varepsilon, \infty)$, where $\varepsilon > 0$ but goes to zero. This "detail," denoted by $(0^-, \infty)$, is required so that $\delta(t)$ fulfills its ultimate objective in Laplace and Fourier transform theory – that of providing the identity function for convolution, e.g., having a transform value of 1 so that $\delta(t)$ convolved with any function or distribution $\psi(t)$ equals $\psi(t)$. For example,

$$\int_{-\infty}^{\infty} \delta(t - \tau)\, u(\tau)\, d\tau = u(t) = \int_{0^-}^{\infty} \delta(t - \tau)\, d\tau.$$

For a good introduction to distribution theory, see Appendix I in Papoulis. For more detail see both Zemanian and Lighthill. Formulas include

$$t\,\delta(t) = 0, \qquad t\,\delta'(t) = -\delta(t), \qquad \frac{1}{t} = \frac{d}{dt} \ln|t|,$$

$$f(t)\,\delta^{(n)}(t) = \sum_{k=0}^{n} (-1)^k \binom{n}{k} f^{(k)}(0)\,\delta^{(n-k)}(t),$$

$$\int_{-\infty}^{\infty} \delta^{(n)}(t)\,\phi(t)\, dt = (-1)^n\,\phi^{(n)}(0).$$

FOURIER TRANSFORMS

The forward and inverse Fourier transforms are defined by:

$$F(\omega) = \mathcal{F}[\, f(t)\,](\omega) = \int_{-\infty}^{\infty} f(t)\, e^{-i\omega t}\, dt, \qquad f(t) = \mathcal{F}^{-1}[\, F(\omega)\,](t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)\, e^{i\omega t}\, d\omega.$$

The properties of the transform include

LINEARITY: $\mathcal{F}[\,\alpha_1 f_1(t) + \alpha_2 f_2(t)\,](\omega) = \alpha_1 \mathcal{F}[\,f_1(t)\,](\omega) + \alpha_2 \mathcal{F}[\,f_2(t)\,](\omega)$.

SYMMETRY: If $F(\omega) = \mathcal{F}[\,f(t)\,](\omega)$, then $\mathcal{F}[\,F(t)\,](\omega) = 2\pi f(-\omega)$.

TIME SCALING: If $F(\omega) = \mathcal{F}[\,f(t)\,](\omega)$ and $c \in \mathbb{R}$, then $\mathcal{F}[\,f(ct)\,](\omega) = F(\omega/c)/|c|$.

TIME SHIFTING: If $F(\omega) = \mathcal{F}[\,f(t)\,](\omega)$, then $\mathcal{F}[\,f(t-\tau)\,](\omega) = F(\omega)\, e^{-i\omega\tau}$.

FREQUENCY SHIFTING: If $F(\omega) = \mathcal{F}[\,f(t)\,](\omega)$, then $\mathcal{F}[\,e^{i\Omega t} f(t)\,](\omega) = F(\omega - \Omega)$.

TIME DIFFERENTIATION: If $F(\omega) = \mathcal{F}[\,f(t)\,](\omega)$, then $\mathcal{F}[\,f^{(n)}(t)\,](\omega) = (i\omega)^n F(\omega)$.

FREQUENCY DERIVATIVES: If $F(\omega) = \mathcal{F}[\,f(t)\,](\omega)$, then $\mathcal{F}[\,(-it)^n f(t)\,](\omega) = F^{(n)}(\omega)$.

CONJUGATION: If $F(\omega) = \mathcal{F}[\,f(t)\,](\omega)$, then $\mathcal{F}[\,f^*(t)\,](\omega) = F^*(-\omega)$.

TIME REVERSAL: If $F(\omega) = \mathcal{F}[\,f(t)\,](\omega)$, then $\mathcal{F}[\,f(-t)\,](\omega) = F(-\omega)$.

TIME CONVOLUTION: If $F_1(\omega) = \mathcal{F}[\,f_1(t)\,](\omega)$ and $F_2(\omega) = \mathcal{F}[\,f_2(t)\,](\omega)$, then

$$f_1(t) \otimes f_2(t) = \int_{-\infty}^{\infty} f_1(\tau)\, f_2(t - \tau)\, d\tau$$

is the convolution of $f_1(t)$ with $f_2(t)$, and $\mathcal{F}[\,f_1(t) \otimes f_2(t)\,](\omega) = F_1(\omega)\, F_2(\omega)$. We also have the equalities $f \otimes g = g \otimes f$ and $f \otimes (g \otimes h) = (f \otimes g) \otimes h$.

MULTIPLICATION: If $F_1(\omega) = \mathcal{F}[\,f_1(t)\,](\omega)$ and $F_2(\omega) = \mathcal{F}[\,f_2(t)\,](\omega)$, then

$$F_1(\omega) \otimes F_2(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F_1(\alpha)\, F_2(\omega - \alpha)\, d\alpha$$

is the convolution of $F_1(\omega)$ with $F_2(\omega)$, and $\mathcal{F}^{-1}[\,F_1(\omega) \otimes F_2(\omega)\,](t) = f_1(t)\, f_2(t)$.

COMPLEX TIME CORRELATION: If $F_1(\omega) = \mathcal{F}[\,f_1(t)\,](\omega)$ and $F_2(\omega) = \mathcal{F}[\,f_2(t)\,](\omega)$, then

$$f_c(t) = f_1(t) \otimes f_2^*(-t) = \int_{-\infty}^{\infty} f_1(\tau)\, f_2^*(\tau - t)\, d\tau$$

is the convolution of $f_1(t)$ with $f_2^*(-t)$, and $\mathcal{F}[\,f_c(t)\,](\omega) = F_1(\omega)\, F_2^*(\omega)$. Also we have that $\mathcal{F}[\,f_c(-t)\,](\omega) = F_1(-\omega)\, F_2^*(-\omega)$.

REAL TIME CORRELATION: If $F_1(\omega) = \mathcal{F}[\, f_1(t)\,](\omega)$ and $F_2(\omega) = \mathcal{F}[\, f_2(t)\,](\omega)$, then

$$f_r(t) = f_1(t) \otimes f_2(-t) = \int_{-\infty}^{\infty} f_1(\tau)\, f_2(\tau - t)\, d\tau$$

is the convolution of $f_1(t)$ with $f_2(-t)$ and $\mathcal{F}[\, f_r(t)\,](\omega) = F_1(\omega)\, F_2(-\omega)$. Also, we have that $\mathcal{F}[\, f_r(-t)\,](\omega) = F_1(-\omega)\, F_2(\omega)$.

MODULATION: If $F(\omega) = \mathcal{F}[\, f(t)\,](\omega)$, then $\mathcal{F}[\, f(t) \cos \beta t\,](\omega) = [F(\omega + \beta) + F(\omega - \beta)]/2$ and $\mathcal{F}[\, f(t) \sin \beta t\,](\omega) = i\,[F(\omega + \beta) - F(\omega - \beta)]/2$.

Some common functions used in Fourier transform analysis include:

$$\text{sinc}(t) = \frac{\sin t}{t}, \qquad p_T(t) = \begin{cases} 0 & |t| > T \\ 1 & |t| < T \end{cases}, \qquad q_T(t) = \begin{cases} 1 - \dfrac{|t|}{T} & |t| < T \\ 0 & |t| > T \end{cases},$$

$$\text{sign}(t) = 2u(t) - 1 = \begin{cases} 1 & t > 0 \\ -1 & t < 0 \end{cases}, \qquad s_T(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT).$$

Some of the more common transform pairs follow.

$\mathcal{F}[\, \delta(t)\,](\omega) = 1$ $\qquad$ $\mathcal{F}[\, \delta^{(n)}(t)\,](\omega) = (i\omega)^n \quad n = 0, 1, \ldots$

$\mathcal{F}[\, 1\,](\omega) = 2\pi\, \delta(\omega)$ $\qquad$ $\mathcal{F}[\, t^n\,](\omega) = 2\pi\, i^n\, \delta^{(n)}(\omega) \quad n = 0, 1, \ldots$

$\mathcal{F}[\, 1/(\pi t)\,](\omega) = -i\, \text{sign}(\omega)$ $\qquad$ $\mathcal{F}[\, t^{-n}\,](\omega) = -i\pi\, (-i\omega)^{n-1}\, \text{sign}(\omega)/(n-1)! \quad n = 1, 2, \ldots$

$\mathcal{F}[\, \text{sign}(t)\,](\omega) = 2/(i\omega)$ $\qquad$ $\mathcal{F}[\, t^n\, \text{sign}(t)\,](\omega) = 2\, n!/(i\omega)^{n+1} \quad n = 0, 1, \ldots$

$\mathcal{F}[\, u(t)\,](\omega) = \pi\, \delta(\omega) + (i\omega)^{-1}$ $\qquad$ $\mathcal{F}[\, t^n\, u(t)\,](\omega) = \pi\, i^n\, \delta^{(n)}(\omega) + n!/(i\omega)^{n+1} \quad n = 0, 1, \ldots$

$\mathcal{F}[\, e^{-\alpha t}\,](\omega) = 2\pi\, \delta(\omega + i\alpha)$ $\qquad$ $\mathcal{F}[\, t^n\, e^{-\alpha t}\,](\omega) = 2\pi\, i^n\, \delta^{(n)}(\omega + i\alpha) \quad n = 0, 1, \ldots$

$\mathcal{F}[\, e^{-\alpha t}\, u(t)\,](\omega) = (\alpha + i\omega)^{-1}$ $\qquad$ $\mathcal{F}[\, t^n\, e^{-\alpha t}\, u(t)](\omega) = n!/(\alpha + i\omega)^{n+1} \quad \alpha > 0, \; n = 0, 1, \ldots$

$\mathcal{F}[\, |t|\,](\omega) = 2\, \omega^{-2}$ $\qquad$ $\mathcal{F}[\, t^\alpha\, \text{sign}(t)\,](\omega) = 2\, \Gamma(\alpha + 1)\, (i\omega)^{-(\alpha+1)} \quad \alpha > -1$

$\mathcal{F}[\, \sin \alpha t\,](\omega) = i\pi[\delta(\omega + \alpha) - \delta(\omega - \alpha)]$ $\qquad$ $\mathcal{F}[\, \cos \alpha t\,](\omega) = \pi[\delta(\omega + \alpha) + \delta(\omega - \alpha)]$

$\mathcal{F}[\ \sinh \alpha t\ ](\omega) = i\pi[\delta(\omega + i\alpha) - \delta(\omega - i\alpha)]$ $\mathcal{F}[\ \cosh \alpha t\ ](\omega) = \pi[\delta(\omega + i\alpha) + \delta(\omega - i\alpha)]$

$\mathcal{F}[\ p_T(t)\ ](\omega) = 2\ T\ \mathrm{sinc}(\omega T)$ $\mathcal{F}[\ \mathrm{sinc}(at)\ ](\omega) = (\pi/a)\ p_a(\omega)$

$\mathcal{F}[\ q_T(t)\ ](\omega) = T\ \mathrm{sinc}^2(\omega T/2)$ $\mathcal{F}[\ \mathrm{sinc}^2(at)\ ](\omega) = (\pi/a)\ q_{2a}(\omega)$

$\mathcal{F}[\ s_T(t)\ ](\omega) = \Omega\ s_\Omega(\omega) \quad \Omega = (2\pi)\ /\ T$ $\mathcal{F}[\ e^{-\alpha|t|}\ ](\omega) = 2\alpha\ (\alpha^2 + \omega^2)^{-1}$

$\mathcal{F}[\ e^{-\alpha t}\cos \beta t\ u(t)\ ](\omega) = (\alpha+i\omega)\ [(\alpha+i\omega)^2+\beta^2]^{-1}$ $\mathcal{F}[\ e^{-\alpha t}\sin \beta t\ u(t)\ ](\omega) = \beta\ [(\alpha+i\omega)^2+\beta^2]^{-1}$

$$\mathcal{F}[\ u(t)\ \cos \beta t\ ](\omega) = \frac{\pi}{2}[\delta(\omega + \beta) + \delta(\omega - \beta)] + \frac{i\omega}{\beta^2 - \omega^2}$$

$$\mathcal{F}[\ u(t)\ \sin \beta t\ ](\omega) = \frac{i\pi}{2}[\delta(\omega + \beta) - \delta(\omega - \beta)] + \frac{\beta}{\beta^2 - \omega^2}$$

$$\mathcal{F}\left[\ \sum_{n=-\infty}^{\infty} \delta'(t - nT)\ \right](\omega) = i\Omega^2 \sum_{n=-\infty}^{\infty} n\ \delta(\omega - n\Omega) \qquad \Omega = \frac{2\pi}{T}$$

$$\mathcal{F}[\ e^{-\beta t^2}\ ](\omega) = \sqrt{\frac{\pi}{\beta}}\ e^{-\omega^2/(4\beta)} \qquad \beta \in \mathbb{C}$$

$$\mathcal{F}[\ \cos\ \alpha t^2\ ](\omega) = \sqrt{\frac{\pi}{2}}\ \cos\left(\frac{\omega^2}{4\alpha} - \frac{\pi}{4}\right)$$

$$\mathcal{F}[\ \sin\ \alpha t^2\ ](\omega) = -\sqrt{\frac{\pi}{2}}\ \sin\left(\frac{\omega^2}{4\alpha} - \frac{\pi}{4}\right)$$

Note that for $x \in \mathbb{R}$ and $\alpha > 0$, $\delta(x) \otimes \delta(x + i\alpha) = 0$. Distribution theory is a way of avoiding more formal Stieltjes integration.

POISSON'S SUM FORMULA: If $F(\omega) = \mathcal{F}[f(t)](\omega)$, then

$$\sum_{n=-\infty}^{\infty} f(t + nT) = \frac{1}{T} \sum_{n=-\infty}^{\infty} e^{i n\Omega t} F(n\Omega) \qquad \Omega = \frac{2\pi}{T}.$$

Consider the special case of this formula where $f(t) = 0$ for $t < 0$ and $t > T$, and define

$$g(t) = \sum_{n=-\infty}^{\infty} f(t + nT).$$

By Poisson's sum formula we see that

$$g(t) = \sum_{n=-\infty}^{\infty} \frac{F(n\Omega)}{T} e^{i n\Omega t}$$

is the Fourier series expansion of periodic $g(t)$ with coefficients

$$\frac{F(n\Omega)}{T} = \frac{1}{T} \int_0^T f(t) e^{-i n\Omega t} dt.$$

The Fourier transform of periodic $g(t)$ is

$$G(\omega) = \sum_{n=-\infty}^{\infty} \frac{F(n\Omega)}{T} \int_{-\infty}^{\infty} e^{-i(\omega - n\Omega)t} dt = \sum_{n=-\infty}^{\infty} \Omega F(n\Omega) \delta(\omega - n\Omega)$$

Hence, the Fourier transform of a periodic function is a train of Dirac delta functions with coefficients $\Omega F(n\Omega)$. Alternatively, $G(\omega) = F(\omega) \Omega s_\Omega(\omega)$, which is the Fourier transform of $f(t) \otimes s_T(t)$, which was our original definition of $g(t)$.

HILBERT TRANSFORMS

The Hilbert transform of $f(t)$ is simply the convolution of $f(t)$ with $1/(\pi t)$. If $y(t)$ is the Hilbert transform of $x(t)$, $y(t) = \mathcal{H}[f(t)](t)$, then $z(t) = x(t) + i\, y(t)$ is an *analytic signal* since $\mathcal{F}[\, 1/(\pi t)\,](\omega) = -i \operatorname{sign}(\omega)$ and $\mathcal{F}[z(t)](\omega) = 2\, U(\omega)\, X(\omega)$, where $X(\omega) = \mathcal{F}[x(t)](\omega)$. Thus, the Fourier transform of an analytic signal $z(t)$, is zero for $\omega < 0$. A *causal* function $f(t)$ has the property that $f(t) = 0$ for $t < 0$. Viewed as a function of real radian frequency ω, the Fourier transform of $z(t)$ is causal in frequency ω.

A few of these transform pairs are

$\mathcal{H}[\ 1/(\pi t)\ ](t) = -\delta(t)$ $\qquad$ $\mathcal{H}[\ \delta(t)\ ](t) = 1/(\pi t)$

$\mathcal{H}[\ \sin(\beta t + \alpha)\ ](t) = -\cos(\beta t + \alpha)$ $\qquad$ $\mathcal{H}[\ \cos(\beta t + \alpha)\ ](t) = \sin(\beta t + \alpha)$

$\mathcal{H}[\ \mathrm{sinc}(at)\ ](t) = (1 - \cos at)/(at)$ $\qquad$ $\mathcal{H}[\ (1 - \cos at)/(at)\ ](t) = -\mathrm{sinc}(at)$

In general, if $x(t) = a(t)\cos[\beta t + \phi(t)]$ is *bandpass*, then $\mathcal{H}[\ x(t)\ ](t) = a(t)\sin[\beta t + \phi(t)]$. Hilbert transformation thus performs a 90-degree phase shift on bandpass signals.

LAPLACE TRANSFORMS

The forward and inverse Laplace transforms are defined by:

$$F(s) = \mathcal{L}[\ f(t)\ ](s) = \int_0^\infty f(t)\, e^{-st}\, dt, \qquad f(t) = \mathcal{L}^{-1}[\ F(s)\ ](t) = \frac{1}{i2\pi}\int_{\alpha - i\infty}^{\alpha + i\infty} F(s)\, e^{st}\, ds$$

The properties of the transform include

LINEARITY: $\mathcal{L}[\ \alpha_1 f_1(t) + \alpha_2 f_2(t)\ ](s) = \alpha_1 \mathcal{L}[\ f_1(t)\ ](s) + \alpha_2 \mathcal{L}[\ f_2(t)\ ](s)$.

TIME SCALING: If $F(s) = \mathcal{L}[\ f(t)\ ](s)$ and $c>0$, then $\mathcal{L}[\ f(ct)\ ](s) = F(s/c)/c$.

TIME SHIFTING: If $F(s) = \mathcal{L}[\ f(t)\ ](s)$, then $\mathcal{L}[\ f(t - \tau)\ U(t - \tau)\ ](s) = F(s)\, e^{-s\tau}$ $\quad \tau > 0$.

FREQUENCY SHIFTING: If $F(s) = \mathcal{L}[\ f(t)\ ](s)$, then $\mathcal{L}[\ e^{\Omega t} f(t)\ ](s) = F(s - \Omega)$.

TIME DIFFERENTIATION: If $F(s) = \mathcal{L}[\ f(t)\ ](s)$, then

$$\mathcal{L}[\ f^{(n)}(t)\ ](s) = s^n F(s) - s^{n-1} f(0) - s^{n-2} f^{(1)}(0) - \ldots - f^{(n-1)}(0).$$

FREQUENCY DERIVATIVES: If $F(s) = \mathcal{L}[\ f(t)\ ](s)$, then $\mathcal{L}[\ (-t)^n f(t)\ ](s) = F^{(n)}(s)$.

TIME INTEGRATION: If $F(s) = \mathcal{L}[\ f(t)\ ](s)$, then $\mathcal{L}[\ \int_0^t f(\tau)\, d\tau\ ](s) = F(s) / s$.

FREQUENCY INTEGRATION: If $F(s) = \mathcal{L}[\ f(t)\ ](s)$, then $\mathcal{L}[\ f(t)/t\ ](s) = \int_s^\infty F(z)\, dz$.

TIME CONVOLUTION: If $F_1(s) = \mathcal{L}[\ f_1(t)\ ](s)$ and $F_2(s) = \mathcal{L}[\ f_2(t)\ ](s)$, then

$$f_1(t) \otimes f_2(t) = \int_0^t f_1(\tau)\, f_2(t - \tau)\, d\tau$$

is the convolution of $f_1(t)$ with $f_2(t)$, and $\mathfrak{L}[\, f_1(t) \otimes f_2(t)\,](s) = F_1(s)\, F_2(s)$.

MULTIPLICATION: If $F_1(s) = \mathfrak{L}[\, f_1(t)\,](s)$ and $F_2(s) = \mathfrak{L}[\, f_2(t)\,](s)$, then

$$F_1(s) \otimes F_2(s) = \frac{1}{i2\pi} \int_{\alpha-i\infty}^{\alpha+i\infty} F_1(\alpha)\, F_2(s - \alpha)\, d\alpha$$

is the convolution of $F_1(s)$ with $F_2(s)$ and $\mathfrak{L}^{-1}[\, F_1(s) \otimes F_2(s)\,](t) = f_1(t)\, f_2(t)$.

INITIAL VALUE THEOREM: If $F(s) = \mathfrak{L}[\, f(t)\,](s)$ and $f(t)$ is causal, then

$$f(0^+) = \lim_{s \to \infty} [\, s\, F(s)\,], \qquad f^{(n)}(0^+) = \lim_{s \to \infty} \left[s^{n+1} F(s) - \sum_{m=0}^{n-1} s^{n-m}\, f^{(m)}(0^+) \right].$$

FINAL VALUE THEOREM: If $F(s) = \mathfrak{L}[\, f(t)\,](s)$, then

$$\lim_{s \to 0} [\, s\, F(s)\,] = f(\infty) < \infty.$$

Some of the more common transform pairs follow. Let $\nu \in \mathbb{C}$, but $\nu \neq -1, -2, \ldots$.

$\mathfrak{L}[\, \delta(t)\,](s) = 1$

$\mathfrak{L}[\, \delta^{(n)}(t)\,](s) = s^n$

$\mathfrak{L}[\, 1\,](s) = s^{-1}$

$\mathfrak{L}[\, t^\nu\,](s) = \Gamma(\nu + 1)\, s^{-(\nu+1)}$

$\mathfrak{L}[\, e^{-at}\,](s) = [s + a]^{-1}$

$\mathfrak{L}[\, t^\nu\, e^{-at}\,](s) = \Gamma(\nu + 1)\, [s + a]^{-(\nu+1)}$

$\mathfrak{L}[\, \sin \beta t\,](s) = \beta\, [s^2 + \beta^2]^{-1}$

$\mathfrak{L}[\, t^{\nu-1} \sin \beta t\,](s) = i\, \Gamma(\nu)\, [(s + i\beta)^{-\nu} - (s - i\beta)^{-\nu}]/2$

$\mathfrak{L}[\, \cos \beta t\,](s) = s\, [s^2 + \beta^2]^{-1}$

$\mathfrak{L}[\, t^{\nu-1} \cos \beta t\,](s) = \Gamma(\nu)\, [(s + i\beta)^{-\nu} + (s - i\beta)^{-\nu}]/2$

$\mathfrak{L}[\, \sinh \beta t\,](s) = \beta\, [s^2 - \beta^2]^{-1}$

$\mathfrak{L}[\, t^{\nu-1} \sinh \beta t\,](s) = \Gamma(\nu)\, [(s - \beta)^{-\nu} - (s + \beta)^{-\nu}]/2$

$\mathfrak{L}[\, \cosh \beta t\,](s) = s\, [s^2 - \beta^2]^{-1}$

$\mathfrak{L}[\, t^{\nu-1} \cosh \beta t\,](s) = \Gamma(\nu)\, [(s - \beta)^{-\nu} + (s + \beta)^{-\nu}]/2$

$\mathfrak{L}[\, e^{-at} \sin \beta t\,](s) = \beta\, [(s + a)^2 + \beta^2]^{-1}$

$\mathfrak{L}[\, e^{-at} \cos \beta t\,](s) = (s + a)\, [(s + a)^2 + \beta^2]^{-1}$

where $\Gamma(z)$ is the **GAMMA** command whose symbolic form is defined in Chapter 19.

$$\mathcal{L}[\, t^{-1} \,](s) = -\gamma - \ln s \qquad \mathcal{L}[\, t^{-n} \,](s) = -(-s)^{n-1} [\ln s - \psi(n)]/\Gamma(n) \quad n = 2, 3, \ldots$$

$$\mathcal{L}[\, t^{-1} e^{-\alpha t} \,](s) = -\gamma - \ln(s + \alpha) \quad \text{RE } s > -\text{RE } \alpha$$

$$\mathcal{L}[\, t^{-n} e^{-\alpha t} \,](s) = (-1)^n (s + \alpha)^{n-1} [\, \ln(s + \alpha) - \psi(n) \,]/\Gamma(n) \quad \text{RE } s > -\text{RE } \alpha, \; n = 2, 3, \ldots$$

$$\mathcal{L}[\, t^{-1} \sin \beta t \,](s) = i [\ln(s - i\beta) - \ln(s + i\beta)]/2 \quad \text{RE } s > |\text{IM } \beta|$$

$$\mathcal{L}[\, t^{-1} \cos \beta t \,](s) = -\gamma - \ln \sqrt{(s^2 + \beta^2)} \quad \text{RE } s > |\text{IM } \beta|$$

$$\mathcal{L}[\, t^{-1} \sinh \beta t \,](s) = [\ln(s + \beta) - \ln(s - \beta)]/2 \quad \text{RE } s > |\text{RE } \beta|$$

$$\mathcal{L}[\, t^{-1} \cosh \beta t \,](s) = -\gamma - \ln \sqrt{(s^2 - \beta^2)} \quad \text{RE } s > |\text{RE } \beta|$$

where γ is the command γ, and $\psi(z)$ is the **PSI** command whose symbolic form is defined in Chapter 19. For $n = 2, 3, \ldots$ and RE $s >$ |IM β|, we have:

$$\mathcal{L}[\, t^{-n} \sin \beta t \,](s) = (-1)^{n-1} \{(s+i\beta)^{n-1} [\ln(s+i\beta) - \psi(n)] - (s-i\beta)^{n-1} [\ln(s-i\beta) - \psi(n)]\}/[i2\, \Gamma(n)]$$

$$\mathcal{L}[\, t^{-n} \cos \beta t \,](s) = (-1)^n \{(s+i\beta)^{n-1} [\ln(s+i\beta) - \psi(n)] + (s-i\beta)^{n-1} [\ln(s-i\beta) - \psi(n)]\}/[2\, \Gamma(n)]$$

Similarly, for $n = 2, 3, \ldots$ and RE $s >$ |RE β|, we have:

$$\mathcal{L}[\, t^{-n} \sinh \beta t \,](s) = (-1)^{n-1} \{(s+\beta)^{n-1} [\ln(s+\beta) - \psi(n)] - (s-\beta)^{n-1} [\ln(s-\beta) - \psi(n)]\}/[2\, \Gamma(n)]$$

$$\mathcal{L}[\, t^{-n} \cosh \beta t \,](s) = (-1)^n \{(s+\beta)^{n-1} [\ln(s+\beta) - \psi(n)] + (s-\beta)^{n-1} [\ln(s-\beta) - \psi(n)]\}/[2\, \Gamma(n)]$$

We also have the formulas:

$$\mathcal{L}[\, (t + \alpha)^{-n} \,](s) = \alpha^{1-n} e^{\alpha s} E_n(\alpha s) \quad \alpha > 0 \text{ and } n = 0, 1, 2, \ldots$$

$$\mathcal{L}[\, (t^2 + 1)^{-1} \,](s) = [\pi/2 - \text{Si}(s)] \cos s + \text{Ci}(s) \sin s$$

where $E_n(z)$ is the exponential integral, Si(z) is the sine integral, and Ci(z) is the cosine integral. Other trigonometric forms are:

$$\mathcal{L}[\, |\sin \beta t| \,](s) = \beta [s^2 + \beta^2]^{-1} \coth[\pi s/(2\beta)] \quad \beta > 0$$

$$\mathcal{L}[\, |\cos \beta t| \,](s) = [s^2 + \beta^2]^{-1} \{s + \beta \, \text{csch}[\pi s/(2\beta)]\} \quad \beta > 0$$

$\mathcal{L}[\ \sin \alpha t \sin \beta t\ ](s) = 2\alpha\beta s\,[s^2 + (\alpha + \beta)^2]^{-1}\,[s^2 + (\alpha - \beta)^2]^{-1} \quad \text{RE } s > |\text{IM}(\alpha \pm \beta)|$

$\mathcal{L}[\ \cos \alpha t \sin \beta t\ ](s) = \beta(s^2 - \alpha^2 + \beta^2)\,[s^2 + (\alpha + \beta)^2]^{-1}\,[s^2 + (\alpha - \beta)^2]^{-1} \quad \text{RE } s > |\text{IM}(\alpha \pm \beta)|$

$\mathcal{L}[\ \cos \alpha t \cos \beta t\ ](s) = s(s^2 + \alpha^2 + \beta^2)\,[s^2 + (\alpha + \beta)^2]^{-1}\,[s^2 + (\alpha - \beta)^2]^{-1} \quad \text{RE } s > |\text{IM}(\alpha \pm \beta)|$

$\mathcal{L}[\ t^{-1} \sin^2 \beta t\ ](s) = \ln[1 + 4(\beta/s)^2]/4 \quad \text{RE } s > 2\ |\text{IM } \beta|$

$\mathcal{L}[\ t^{-1} \cos^2 \beta t\ ](s) = -\gamma - \ln[s^2\,(s^2 + 4\beta^2)] \quad \text{RE } s > 2\ |\text{IM } \beta|$

Forms involving $\sqrt{t}$ are:

$\mathcal{L}[\ \sqrt{\alpha}\ t^{-1/2}\,(t + \alpha)^{-1}\ ](s) = \pi\ s^{\alpha s}\ \text{erfc}\ \sqrt{(\alpha s)} \quad |\arg \alpha| < \pi$

$\mathcal{L}[\ (\pi t)^{-1/2}\ u(t - \alpha)\ ](s) = s^{-1/2}\ \text{erfc}\ \sqrt{(\alpha s)} \quad \alpha \geq 0$

$\mathcal{L}[\ \{\pi(t + \alpha)\}^{-1/2}\ ](s) = s^{-1/2}\ e^{\alpha s}\ \text{erfc}\ \sqrt{(\alpha s)} \quad |\arg \alpha| < \pi$

$\mathcal{L}[\ (\pi t)^{-1/2}\ e^{-2\alpha\sqrt{t}}\ ](s) = s^{-1/2}\ \text{EXP}(\alpha^2/s)\ \text{erfc}\ \sqrt{(\alpha s)} \quad |\arg \alpha| < \pi$

$\mathcal{L}[\ t^{n-1/2}\ ](s) = \sqrt{\pi}\ 2^{-n}\ [1 \bullet 3 \bullet 5 \bullet \ldots (2n-1)]\ s^{-(n+1/2)} \quad n = 1, 2, \ldots$

$\mathcal{L}[\ 2\sqrt{(t/\pi)}\ ](s) = s^{-3/2}$ $\qquad$ $\mathcal{L}[\ (\pi t)^{-1/2}\ ](s) = s^{-1/2}$

$\mathcal{L}[\ (\pi\beta)^{-1/2} \sin 2\sqrt{(\beta t)}\ ](s) = s^{-3/2}\ e^{-\beta/s}$ $\qquad$ $\mathcal{L}[\ (\pi\beta)^{-1/2} \cos 2\sqrt{(\beta t)}\ ](s) = s^{-1/2}\ e^{-\beta/s}$

$\mathcal{L}[\ (\pi\beta)^{-1/2} \sinh 2\sqrt{(\beta t)}\ ](s) = s^{-3/2}\ e^{\beta/s}$ $\qquad$ $\mathcal{L}[\ (\pi\beta)^{-1/2} \cosh 2\sqrt{(\beta t)}\ ](s) = s^{-1/2}\ e^{\beta/s}$

$\mathcal{L}[\ t^{-1} \sin(2\beta\sqrt{t})\ ](s) = \pi\ \text{erf}(\beta/\sqrt{s}) \quad \text{RE } s > 0$

$\mathcal{L}[\ t^{-1/2} \sin 2\sqrt{(\beta t)}\ ](s) = -i\ \sqrt{(\pi/s)}\ e^{-\beta/s}\ \text{erf}\ \{i\sqrt{(\beta/s)}\} \quad \text{RE } s > 0$

$\mathcal{L}[\ t^{-1/2} \cos 2\sqrt{(\beta t)}\ ](s) = \sqrt{(\pi/s)}\ e^{-\beta/s} \quad \text{RE } s > 0$

$\mathcal{L}[\ t^{-1/2} \sinh 2\sqrt{(\beta t)}\ ](s) = \sqrt{(\pi/s)}\ e^{\beta/s}\ \text{erf}\ \{\sqrt{(\beta/s)}\} \quad \text{RE } s > 0$

$\mathcal{L}[\ t^{-1/2} \cosh 2\sqrt{(\beta t)}\ ](s) = \sqrt{(\pi/s)}\ e^{\beta/s} \quad \text{RE } s > 0$

$\mathcal{L}[\ t^{-1/2} \sinh^2 \sqrt{(\beta t)}\ ](s) = \sqrt{(\pi/s)}\ [e^{\beta/s} - 1]/2 \quad \text{RE } s > 0$

$\mathcal{L}[\ t^{-1/2} \cosh^2 \sqrt{(\beta t)}\ ](s) = \sqrt{(\pi/s)}\ [e^{\beta/s} + 1]/2 \quad \text{RE } s > 0$

Formulas involving ln t with RE $\beta \geq 0$ are:

$\mathfrak{L}[\ t^{\beta-1} \ln t\ ](s) = \Gamma(\beta)\ s^{-\beta}\ [\psi(\beta) - \ln s]$ $\qquad$ $\mathfrak{L}[\ \ln t\ ](s) = -s^{-1}\ [\gamma + \ln s]$

$\mathfrak{L}[\ \ln(1 + \alpha t)\ ](s) = s^{-1}\ e^{s/\alpha}\ E_1(s/\alpha)$ $\qquad$ $\mathfrak{L}[\ \ln(t + \alpha)\ ](s) = s^{-1}\ [\ln \alpha + e^{\alpha s}\ E_1(\alpha s)]$

where $|\arg \alpha| < \pi$, and as noted in Chapter 4, $E_1(z) = -Ei(-z)$ for $|\arg z| < \pi$.

Formulas for the error function erf with $\alpha > 0$ are:

$\mathfrak{L}[\ \text{erf } t/(2\alpha)\ ](s) = s^{-1}\ \text{EXP}(\alpha^2 s^2)\ \text{erfc } \alpha s$

$\mathfrak{L}[\ (\alpha\sqrt{\pi})^{-1}\ \text{EXP}(-t^2/(2\alpha)^2\ ](s) = \text{EXP}(\alpha^2 s^2)\ \text{erfc } \alpha s$

$\mathfrak{L}[\ a^{-1}\ \text{EXP}(a^2 t)\ \text{erf } a\sqrt{t}\ ](s) = s^{-1/2}\ (s - a^2)^{-1}$ $\qquad$ $\mathfrak{L}[\ \text{EXP}(a^2 t)\ \text{erfc } a\sqrt{t}\ ](s) = s^{-1/2}\ (\sqrt{s} + a)^{-1}$

$\mathfrak{L}[\ \text{erfc } \beta/(2\sqrt{t})\ ](s) = s^{-1}\ e^{-\beta\sqrt{s}}$ $\qquad$ $\mathfrak{L}[\ (4t)^{n/2}\ i^n\ \text{erfc } \beta/(2\sqrt{t})\ ](s) = s^{-(n/2+1)}\ e^{-\beta\sqrt{s}}$

where $\beta \geq 0$, n = 0, 1, . . ., and i^n erfc z denotes the nth integral of the complementary error function erfc z.

$\mathfrak{L}[\ (\pi t)^{-1/2} - a\ \text{EXP}(a^2 t)\ \text{erfc } a\sqrt{t}\ ](s) = (\sqrt{s} + a)^{-1}$

$\mathfrak{L}[\ (\pi t)^{-1/2} + a\ \text{EXP}(a^2 t)\ \text{erf } a\sqrt{t}\ ](s) = \sqrt{s}\ (s - a^2)^{-1}$

$\mathfrak{L}[\ (b - a)^{-1/2}\ e^{-at}\ \text{erf } [\sqrt{(b - a)}\sqrt{t}]\ ](s) = (s + a)^{-1}\ (s + b)^{-1/2}$

$\mathfrak{L}[\ e^{a\beta}\ \text{EXP}(a^2 t)\ \text{erfc}\{a\sqrt{t} + \beta/(2\sqrt{t})\}\ ](s) = s^{-1/2}\ (a + \sqrt{s})^{-1}\ e^{-\beta\sqrt{s}}$ $\qquad$ $\beta \geq 0$

Formulas related to the exponential integrals are:

$\mathfrak{L}[\ e^{at}\ \{\ln(a + b) + E_1([a + b]t)\}\ ](s) = (s - a)^{-1}\ \ln(s + b)$

$\mathfrak{L}[\ \cos t\ Si(t) - \sin t\ Ci(t)\ ](s) = (s^2 + 1)^{-1}\ \ln s$

$\mathfrak{L}[\ -\sin t\ Si(t) - \cos t\ Ci(t)\ ](s) = s\ (s^2 + 1)^{-1}\ \ln s$

$\mathfrak{L}[\ E_1(t/\alpha)\ ](s) = s^{-1}\ \ln(1 + \alpha s)$ $\qquad$ $\alpha > 0$

$\mathfrak{L}[\ t^{-1}\ (e^{-bt} - e^{-at})\ ](s) = \ln[\ (s + a)/(s + b)\ ]$

$\mathcal{L}[\ 2\ \mathrm{Ci}(t/\alpha)\ ](s) = -s^{-1}\ \ln(1 + \alpha^2 s^2) \quad \alpha > 0$

$\mathcal{L}[\ 2\ \ln a - 2\ \mathrm{Ci}(at)\ ](s) = s^{-1}\ \ln(s^2 + a^2) \quad \alpha > 0$

$\mathcal{L}[\ 2\ \{at\ \ln a + \sin at - at\ \mathrm{Ci}(at)\}\ ](s) = a\ s^{-2}\ \ln(s^2 + a^2) \quad a > 0$

$\mathcal{L}[\ \mathrm{Si}(\alpha t)\ ](s) = s^{-1}\ \arctan(\alpha/s)$

Formulas involving Bessel functions are:

$\mathcal{L}[\ J_0(at)\ ](s) = (s^2 + a^2)^{-1/2}$ $\qquad$ $\mathcal{L}[\ a^\nu\ J_\nu(at)\ ](s) = [\sqrt{(s^2 + a^2)} - s]^\nu\ (s^2 + a^2)^{-1/2} \quad \nu > -1$

$\mathcal{L}[\ t\ J_0(at)\ ](s) = s\ (s^2 + a^2)^{-3/2}$

$\mathcal{L}[\ t\ J_1(at)\ ](s) = a\ (s^2 + a^2)^{-3/2}$

$\mathcal{L}[\ \sqrt{\pi}\ \{t/(2a)\}^{n-1/2}\ J_{n-1/2}(at)\ ](s) = \Gamma(n)\ (s^2 + a^2)^{-n} \quad n > 0$

$\mathcal{L}[\ n\ a^n\ t^{-1}\ J_n(at)\ ](s) = [\sqrt{(s^2 + a^2)} - s]^n \quad n > 0$

$\mathcal{L}[\ I_0(at)\ ](s) = (s^2 - a^2)^{-1/2}$ $\qquad$ $\mathcal{L}[\ a^\nu\ I_\nu(at)\ ](s) = [s - \sqrt{(s^2 + a^2)}]^\nu\ (s^2 - a^2)^{-1/2} \quad \nu > -1$

$\mathcal{L}[\ t\ I_0(at)\ ](s) = s\ (s^2 - a^2)^{-3/2}$

$\mathcal{L}[\ t\ I_1(at)\ ](s) = a\ (s^2 - a^2)^{-3/2}$

$\mathcal{L}[\ \sqrt{\pi}\ \{t/(2a)\}^{n-1/2}\ I_{n-1/2}(at)\ ](s) = \Gamma(n)\ (s^2 - a^2)^{-n} \quad n > 0$

$\mathcal{L}[\ e^{-(a+b)t/2}\ I_0\{(a - b)t/2\}\ ](s) = (s + a)^{-1/2}\ (s + b)^{-1/2}$

$\mathcal{L}[\ n\ t^{-1}\ e^{-(a+b)t/2}\ I_n\{(a - b)t/2\}\ ](s) = (a - b)^n\ [\sqrt{(s + a)} + \sqrt{(s + b)}]^{-2n} \quad n > 0$

$\mathcal{L}[\ \sqrt{\pi}\ [t/(a - b)]^{n-1/2}\ e^{-(a+b)t/2}\ I_{n-1/2}\{(a - b)t/2\}\ ](s) = \Gamma(n)\ (s + a)^{-n}\ (s + b)^{-n} \quad n > 0$

$\mathcal{L}[\ (t/\alpha)^{(\mu-1)/2}\ J_{\mu-1}(2\ \sqrt{(\alpha t)}\ )\ ](s) = s^{-\mu}\ e^{-\alpha/s} \quad \mu > 0$

$\mathcal{L}[\ (t/\alpha)^{(\mu-1)/2}\ I_{\mu-1}(2\ \sqrt{(\alpha t)}\ )\ ](s) = s^{-\mu}\ e^{\alpha/s} \quad \mu > 0$

$\mathcal{L}[\ n!\ (\pi t)^{-1/2}\ H_{2n}(\sqrt{t})\ ](s) = (2n)!\ (1 - s)^n\ s^{-(n+1/2)}$ $\qquad$ Hermite polynomial

$\mathcal{L}[\ n!\ (\pi)^{-1/2}\ H_{2n+1}(\sqrt{t})\ ](s) = (2n + 1)!\ (1 - s)^n\ s^{-(n+3/2)}$ $\qquad$ Hermite polynomial

$$\mathcal{L}\left[\sum_{n=0}^{\infty} \delta(t - nT)\right](s) = \frac{1}{1 - e^{-Ts}} \qquad T > 0$$

$$\mathcal{L}\left[\sum_{n=0}^{\infty} (-1)^n \delta(t - nT)\right](s) = \frac{1}{1 + e^{-Ts}} \qquad T > 0$$

$$\mathcal{L}\left[2\sum_{n=0}^{\infty} \delta(t - [2n+1]T)\right](s) = \frac{1}{\sinh Ts} \qquad T > 0$$

$$\mathcal{L}\left[2\sum_{n=0}^{\infty} (-1)^n \delta(t - [2n + 1]T)\right](s) = \frac{1}{\cosh Ts} \qquad T > 0$$

$$\mathcal{L}\left[\delta(t) + 2\sum_{n=1}^{\infty} (-1)^n \delta(t - 2nT)\right](s) = \tanh Ts \qquad T > 0$$

$$\mathcal{L}\left[2\sum_{n=0}^{\infty} e^{-(2n + 1)\beta} \delta(t - [2n+1]T)\right](s) = \frac{1}{\sinh(Ts + \beta)} \qquad T > 0$$

$$\mathcal{L}\left[2\sum_{n=0}^{\infty} e^{-(2n + 1)\beta} \delta(t + \tau - [2n+1]T)\right](s) = \frac{e^{-\tau s}}{\sinh(Ts + \beta)} \qquad T > 0 \quad \tau > 0$$

Inverse Laplace transforms are easily computed with the **IXFRM** command. The symbolic higher transcendental functions with defined derivatives, which are required for some of the residue integrations, are available. See Appendix A.

Erdelyi, A., Magnus, W., Oberhettinger, F., and Tricomi, F., *Tables of Integral Transforms*, 2 Volumes, New York, McGraw–Hill, 1954.

Lighthill, M., *Fourier Analysis and Generalized Functions*, New York, Cambridge Univ. Press, 1970.

Papoulis, A., *The Fourier Integral and its Applications*, New York, McGraw–Hill, 1962.

Zemanian, A., *Distribution Theory and Transform Analysis*, New York, McGraw–Hill, 1965.

APPENDIX G

DISCRETE TIME TRANSFORMS

INTRODUCTION

Again we will leave the rigorous details to the Chapter 27 references and focus on the basic concepts. Discrete time waveforms are created by a process called sampling, which can be modeled mathematically as multiplying the input function x(t) by the sampling distribution $s_T(t)$[†]:

$$x_s(t) = x(t)\, s_T(t) = x(t) \sum_{n=-\infty}^{\infty} \delta(t - nT) = \sum_{n=-\infty}^{\infty} x(nT)\, \delta(t - nT),$$

where T is the sampling period. Now let $\Omega = 2\pi/T$, $\mathcal{L}[\ ]$ denote Laplace transform, and $\mathcal{F}[\ ]$ denote Fourier transform. Since $\mathcal{F}[\ s_T(t)\ ](\omega) = \Omega\, s_\Omega(\omega)$ and multiplication in the time domain corresponds to convolution in the frequency domain, we have the result

$$\mathcal{F}[\ x_s(t)\ ](\omega) = \int_{-\infty}^{\infty} \Omega \sum_{n=-\infty}^{\infty} \delta(\alpha - n\Omega)\, \mathcal{F}[\ x(t)\ ](\omega - \alpha)\, d\alpha = \Omega \sum_{n=-\infty}^{\infty} \mathcal{F}[\ x(t)\ ](\omega - n\Omega).$$

The process of sampling has repeated periodically the transform of x(t), $\mathcal{F}[\ x(t)\ ](\omega)$, with period Ω. The Nyquist sampling theorem defines the minimal conditions such that the periodic repetitions of $\mathcal{F}[\ x(t)\ ](\omega)$ do not overlap (alias).

[†] This can be extended to multidimensional sampling lattices in the presence of timing jitter, level errors, and track–hold amplifier hardware errors. See Holland.

Now suppose that x(t) is *causal*, that is, x(t) = 0 for t < 0. Then the Laplace transform of $x_s(t)$ is

$$\mathcal{L}[\ x_s(t)\ ](s) = \sum_{n=0}^{\infty} x(nT)\ e^{-nTs}.$$

Define X(z) to be the z transform of x(t). Then we have the relations

$$X(z) = \mathcal{L}[\ x_s(t)\ ](\ln(z)/T) = \sum_{n=0}^{\infty} x(nT)z^{-n}.$$

If z is evaluated on the unit circle as $z = e^{i\,2\pi k/N}$, we have the discrete Fourier series

$$X[k] = \sum_{n=0}^{\infty} x(nT)\ e^{-i\,2\pi kn/N}.$$

Computing the inverse discrete Fourier series gives the result

$$y(nT) = \frac{1}{N}\sum_{k=0}^{N} X[k]\ e^{i\,2\pi kn/N} = \sum_{r=-\infty}^{\infty} x(nT + rNT) \neq x(nT).$$

Hence, in general y(nT) is an aliased version of x(nT). If in addition to causality we require that x(nT) = 0 for $n \geq N$, then y(nT) does equal x(nT), and the discrete Fourier series formula can be used as the discrete Fourier transform. This is how the discrete transforms relate to the continuous ones and is covered in far more detail in the references for Chapter 27.

z TRANSFORM

The forward and inverse z transforms are defined by:

$$X(z) = \zeta[\ x[n]\ ](z) = \sum_{n=-\infty}^{\infty} x[n]\ z^{-n}, \qquad x[n] = \zeta^{-1}[\ X(z)\ ](n) = \frac{1}{2\pi i}\oint_C X(z)\ z^{n-1}\ dz,$$

where x[n] = x(nT), and C is a closed contour which encircles the origin. The above z transform is called the *two-sided* or *bilateral* z transform. The *one-sided* or *unilateral* z transform is

$$X^{+}(z) = \zeta^{+}[\, x[n] \,](z) = \sum_{n=0}^{\infty} x[n]\, z^{-n}.$$

DEFINITIONS: Let u[n] = u(nT) denote the *unit step function* and $\delta[n]$ the *unit sample sequence*. u[n] = 1 for $n \geq 0$, and zero otherwise. $\delta[n] = 1$ for $n = 0$, and equals zero otherwise.

The properties of the z transform are

LINEARITY: $\zeta[\, \alpha_1 x_1[n] + \alpha_2 x_2[n] \,](z) = \alpha_1 \zeta[\, x_1[n] \,](z) + \alpha_2 \zeta[\, x_2[n] \,](z)$.

TIME SHIFTING: If $X(z) = \zeta[\, x[n] \,](z)$, then $\zeta[\, x[n-k] \,](z) = z^{-k} X(z)$.

SCALING IN Z DOMAIN: If $X(z) = \zeta[\, x[n] \,](z)$, then $\zeta[\, a^n x[n] \,](z) = X(a^{-1}z)$.

TIME REVERSAL: If $X(z) = \zeta[\, x[n] \,](z)$, then $\zeta[\, x[-n] \,](z) = X(z^{-1})$.

Z DOMAIN DIFFERENTIATION: If $X(z) = \zeta[\, x[n] \,](z)$, then

$$\zeta[\, n\, x[n] \,](z) = -z \frac{dX(z)}{dz}.$$

TIME CONVOLUTION: If $X_1(z) = \zeta[\, x_1[n] \,](z)$ and $X_2(z) = \zeta[\, x_2[n] \,](z)$, then

$$x[n] = x_1[n] \otimes x_2[n] = \sum_{m=-\infty}^{\infty} x_1[m]\, x_2[n - m]$$

is the convolution of $x_1[n]$ with $x_2[n]$, and $\zeta[\, x[n] \,](z) = X_1(z)\, X_2(z)$.

MULTIPLICATION: If $X_1(z) = \zeta[\, x_1[n] \,](z)$ and $X_2(z) = \zeta[\, x_2[n] \,](z)$, then

$$X(z) = X_1(z) \otimes X_2(z) = \frac{1}{2\pi i} \oint_C X_1(w)\, X_2\left(\frac{z}{w}\right) w^{-1}\, dw$$

is the convolution of $X_1(z)$ with $X_2(z)$, $\zeta^{-1}[\, X(z) \,](n) = x_1[n]\, x_2[n]$, and C is a contour which encloses the origin and lies within the common region of convergence.

TIME CORRELATION: If $X_1(z) = \zeta[\, x_1[n] \,](z)$ and $X_2(z) = \zeta[\, x_2[n] \,](z)$, then

$$\rho[n] = x_1[n] \otimes x_2[-n] = \sum_{m=-\infty}^{\infty} x_1[m]\, x_2[m - n]$$

is the correlation of $x_1[n]$ with $x_2[n]$, and $\zeta[\, \rho[n] \,](z) = X_1(z)\, X_2(z^{-1})$.

INITIAL VALUE THEOREM: If $X(z) = \zeta[\, x[n] \,](z)$ and $x[n]$ is causal, then

$$\lim_{z \to \infty} X(z) = x[0].$$

Some of the common z transform pairs follow.

TIME: $\tau = t/T$	DISCRETE	Z TRANSFORM
$\delta(t)$	$\delta[n]$	1
$\delta(t - kT)$	$\delta[n - k]$	z^{-k}
$u(t)$	$u[n]$	$z\,[z - 1]^{-1}$
$\tau\, u(t)$	$n\, u[n]$	$z\,[z - 1]^{-2}$
$a^{\tau}\, u(t)$	$a^n\, u[n]$	$z\,[z - a]^{-1}$
$\tau\, a^{\tau}\, u(t)$	$n\, a^n\, u[n]$	$a\, z\,[z - a]^{-2}$
$\tau^2\, u(t)$	$n^2\, u[n]$	$z\,(z+1)\,[z - 1]^{-3}$
$\tau^3\, u(t)$	$n^3\, u[n]$	$z\,(z^2 + 4z + 1)\,[z - 1]^{-4}$
$a^{\tau}\, u(-t - T)$	$a^n\, u[-n - 1]$	$-z\,[z - a]^{-1}$
$\tau\, a^{\tau}\, u(-t - T)$	$n\, a^n\, u[-n - 1]$	$a\, z\,[z - a]^{-2}$
$\sin \Omega\tau\, u(t)$	$\sin \Omega n\, u[n]$	$z \sin \Omega\,[z^2 - 2z \cos \Omega + 1]^{-1}$
$\cos \Omega\tau\, u(t)$	$\cos \Omega n\, u[n]$	$z\,(z - \cos \Omega)\,[z^2 - 2z \cos \Omega + 1]^{-1}$

TIME: $\tau = t/T$	DISCRETE	Z TRANSFORM
$\sinh \Omega\tau\ u(t)$	$\sinh \Omega n\ u[n]$	$z \sinh \Omega\ [z^2 - 2z \cosh \Omega + 1]^{-1}$
$\cosh \Omega\tau\ u(t)$	$\cosh \Omega n\ u[n]$	$z\ (z - \cosh \Omega)\ [z^2 - 2z \cosh \Omega + 1]^{-1}$
$e^{-\alpha\tau}\ u(t)$	$e^{-\alpha n}\ u[n]$	$z\ [z - e^{-\alpha}]$
$a^{\tau} \sin \Omega\tau\ u(t)$	$a^{n} \sin \Omega n\ u[n]$	$az \sin \Omega\ [z^2 - 2za \cos \Omega + a^2]^{-1}$
$a^{\tau} \cos \Omega\tau\ u(t)$	$a^{n} \cos \Omega n\ u[n]$	$az\ (z - a \cos \Omega)\ [z^2 - 2za \cos \Omega + a^2]^{-1}$
$e^{-\alpha\tau} \sin \Omega\tau\ u(t)$	$e^{-\alpha n} \sin \Omega n\ u[n]$	$z\ e^{-\alpha} \sin \Omega\ [z^2 - 2ze^{-\alpha} \cos \Omega + e^{-2\alpha}]^{-1}$
$e^{-\alpha\tau} \cos \Omega\tau\ u(t)$	$e^{-\alpha n} \cos \Omega n\ u[n]$	$z\ e^{-\alpha}(z - e^{-\alpha} \cos \Omega)[z^2 - 2ze^{-\alpha} \cos \Omega + e^{-2\alpha}]^{-1}$
$\tau\ e^{-\alpha\tau}\ u(t)$	$n\ e^{-\alpha n}\ u[n]$	$z\ e^{-\alpha}\ [z - e^{-\alpha}]^{-2}$
$\tau^2\ e^{-\alpha\tau}\ u(t)$	$n^2\ e^{-\alpha n}\ u[n]$	$z\ e^{-\alpha}\ (z + e^{-\alpha})\ [z - e^{-\alpha}]^{-3}$

Inverse z transforms are easily computed with the **IXFRM** command.

DISCRETE FOURIER TRANSFORM

The forward and inverse discrete Fourier transforms (DFT) are defined by:

$$X[k] = DFT[\ x[n]\ ](k) = \sum_{n=0}^{N-1} x[n]\ e^{-i\ 2\pi kn/N},$$

$$x[n] = IDFT[\ X[k]\ ](n) = \frac{1}{N} \sum_{k=0}^{N-1} X[k]\ e^{i\ 2\pi kn/N},$$

where X[k] is the DFT of x[n]. The properties of the DFT are

PERIODICITY: $x[n + N] = x[n]$ and $X[k + N] = X[k]$.

LINEARITY: If $x[n] = \alpha_1\ x_1[n] + \alpha_2\ x_2[n]$, then $X[k] = \alpha_1\ X_1[k] + \alpha_2\ X_2[k]$.

TIME REVERSAL: If $y[n] = x[-n \text{ MOD } N] = x[N - n]$, then

$$Y[k] = X[-k \text{ MOD } N] = X[N - k].$$

CIRCULAR TIME SHIFT: If $y[n] = x[n - m \text{ MOD } N]$, then $Y[k] = X[k]\, e^{-i\, 2\pi km/N}$.

CIRCULAR FREQUENCY SHIFT: If $y[n] = x[n]\, e^{i\, 2\pi mn/N}$, then

$$Y[k] = X[k - m \text{ MOD } N].$$

CONJUGATE: If $y[n] = x^*[n]$, then $Y[k] = X^*[-k \text{ MOD } N] = X^*[N - k]$.

CONVOLUTION: If $X_1[k] = \text{DFT}[\, x_1[n]\,](k)$ and $X_2[k] = \text{DFT}[\, x_2[n]\,](k)$, then

$$y[n] = x_1[n] \otimes x_2[n] = \sum_{m=0}^{N-1} x_1[m]\, x_2[n - m \text{ MOD } N]$$
$$= \sum_{m=0}^{N-1} x_2[m]\, x_1[n - m \text{ MOD } N]$$

is the circular convolution of $x_1[n]$ with $x_2[n]$, and $\text{DFT}[\, y[n]\,](k) = X_1[k]\, X_2[k]$.

MULTIPLICATION: If $X_1[k] = \text{DFT}[\, x_1[n]\,](k)$ and $X_2[k] = \text{DFT}[\, x_2[n]\,](k)$, then

$$Y[k] = X_1[k] \otimes X_2[k] = \frac{1}{N} \sum_{m=0}^{N-1} X_1[m]\, X_2[k - m \text{ MOD } N]$$
$$= \frac{1}{N} \sum_{m=0}^{N-1} X_2[m]\, X_1[k - m \text{ MOD } N]$$

is the circular convolution of $X_1[k]$ with $X_2[k]$, and $\text{IDFT}[\, Y[k]\,](n) = x_1[n]\, x_2[n]$.

CORRELATION: If $X_1[k] = \text{DFT}[\, x_1[n]\,](k)$ and $X_2[k] = \text{DFT}[\, x_2[n]\,](k)$, then

$$\rho[n] = x_1[n] \otimes x_2^*[-n] = \sum_{m=0}^{N-1} x_1[m]\, x_2^*[m - n \text{ MOD } N]$$

is the circular correlation of $x_1[n]$ with $x_2[n]$, and $\text{DFT}[\, \rho[n]\,](k) = X_1[k]\, X_2^*[k]$.

The commands **FFT** and **IFFT** perform forward and inverse DFTs for N equal to a power of two. Programs DFT and IDFT are also available for arbitrary N.

Commands **FFT** and **IFFT** use the classic decimation-in-time algorithm. The first step is the bit reversal of the data. This is performed by the command **BITRV**. There are self-sorting in-place algorithms available. See Temperton. By using tensor product notation, one can fine tune an FFT algorithm to a particular computer architecture of interest. Commands **RORDR** and **CORDR** can be used to build the permutation stride matrices and the below program KPRD will perform the tensor (Kronecker) products. See Granata. The matrix approach to representing discrete Fourier transforms and other discrete transforms is introduced in the next section. Two dimensional FFTs are available. See Appendix A.

PERFECT RECONSTRUCTION FILTER BANKS

The discrete Fourier transform is easily represented as a matrix multiply. Define $W = e^{-i2\pi/N}$ where N is the size of the DFT. Then if X[k] = DFT[x[n]], we have

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ \vdots \\ X[N-1] \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & \cdots & W^0 \\ W^0 & W^1 & W^2 & \cdots & W^{N-1} \\ W^0 & W^2 & W^4 & \cdots & W^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ W^0 & W^{N-1} & W^{2(N-1)} & \cdots & W^{(N-1)^2} \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ \vdots \\ x[N-1] \end{bmatrix} = Mx \ .$$

Consider each row of the above matrix as a finite impulse response (FIR) filter. Then each output value X[k] represents one value of the convolution of the kth row with the input sequence x[n], in particular, the convolution output with input x[n] for n = 0, 1, . . ., N−1. In addition, for a sufficiently long sequence x[n], the convolution of each of the matrix rows with x[m], x[m + 1], . . ., x[m + (N−1)] for m = 0, 1, . . ., provides outputs $X_m[k]$. Now the discrete Fourier transform theorem says for each non-overlapping block of N input values, the outputs $X_m[k]$ for m = 0 are sufficient to perfectly reconstruct the input values x[0], x[1], . . ., x[N−1]. Thus, the convolution outputs $X_m[k]$ may be decimated by a factor of N, and only the values X[0], X[1], . . ., X[N−1] retained. This is called *maximal or critical decimation*. The commands **VDEC** and **VINTP** are available for decimating and interpolating sequences.

The above transform may be regarded as an example of sub-band decomposition. The forward transform is called the *analysis* operation and the inverse transform (IDFT) is called the *synthesis* operation. Observe that the inverse transform matrix is the Hermitian transpose of the above matrix M divided by N. The program on the next page constructs the above DFT matrix.

```
« → N « 'EXP(-i×2×π/N)' →NUM → W « 0 N 1 - FOR K W K ^
  N EINDX →ROW NEXT 2 N START RCMB NEXT » » »
```

Again regarding each row of the DFT matrix as a FIR filter, the z transform of that filter can be written as

$$H_r(z) = \sum_{m=0}^{N-1} W^{rm} z^{-m} \qquad r = 0, 1, \ldots, N-1$$

where r denotes the row. The collection of these values forms the column vector

$$H(z) = \begin{bmatrix} H_0(z) \\ H_1(z) \\ \vdots \\ H_{N-1}(z) \end{bmatrix} = MV_N(z) \qquad V_N(z) = \begin{bmatrix} 1 \\ z^{-1} \\ \vdots \\ z^{-(N-1)} \end{bmatrix}$$

which mathematically defines the analysis operation. The corresponding synthesis operation is $H^H(z^{-1})$. The value of $H^H(z^{-1})H(z)$ after reordering the sums is

$$H^H(z^{-1})H(z) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} \sum_{r=0}^{N-1} W^{-rm} z^m W^{rn} z^{-n}$$

which is the overall system *transfer function*. Observing that

$$z^{m-n} \sum_{r=0}^{N-1} W^{r(n-m)} = \begin{cases} N & m = n \\ 0 & m \neq n \end{cases}$$

we have the result that $H^H(z^{-1})H(z) = N^2$. This is the condition for *perfect reconstruction*, that is, having decomposed and decimated the input sequence x[n] into N sub-bands with analysis filter H(z), we can perfectly reconstruct x[n] by using the synthesis filter $H^H(z^{-1})$ within a scale factor. Please note, however, perfect reconstruction only implies that *aliasing* can be canceled, not that it does not occur.

Suppose we wish to reduce the aliasing error between sub-bands. This can be done by employing a polyphase filter of length gN, for *genus* g = 1, 2, . . ., ahead of the DFT. After appropriate decimation, this effectively increases the impulse response of each of the N sub-band filters to gN samples, which improves the isolation between the sub-bands. By using this technique, the unwindowed DFT 13 dB sub-band isolation can be made over 100 dB using a genus 8 design.

In the polyphase filtering case, the above equations generalize to $H(z) = P(z)V_N(z)$ where $P(z)$ is an $N \times N$ matrix. The condition for perfect reconstruction is that $V_N^T(z^{-1})P^H(z^{-1})P(z)V_N(z)$ equals a constant as it did in the DFT case with $P(z) = M$. Thus, we are motivated to examine the properties of M, the DFT matrix, which give rise to perfect reconstruction. M has orthogonal rows and columns so that $M^HM = MM^H = NI$. A matrix is said to be *paraunitary* if it satisfies the quadratic condition $P^H(z^{-1})P(z) = P(z)P^H(z^{-1}) = cI$, for $c \in \mathbb{R}$. This is the key perfect reconstruction property.

Clearly, if $P(z)$ is paraunitary, so is $MP(z)$. Consequently, if $P(z)$ is a diagonal matrix with elements $P_m(z)$, the rows of $H(z) = MP(z)V_N(z)$ are

$$H_r(z) = \sum_{m=0}^{N-1} W^{rm} z^{-m} P_m(z) \qquad r = 0, 1, \ldots, N - 1$$

which is the standard polyphase filter expression in the signal processing literature. Observe that $P_m^*(z^{-1})P_m(z) = c$ for $m = 0, 1, \ldots, N - 1$ in this special case.

For $g \geq 1$ and $P(z)$ a FIR filter, the elements of $P(z)$ take the form of the sums

$$p_{sr}(z) = \sum_{k=0}^{g-1} a_{kN+r}^{s} z^{-kN} \qquad s, r = 0, 1, \ldots, N - 1$$

where a_r^s are coefficients of an $N \times gN$ matrix A with $0 \leq s < N$. In the IIR case, each $p_{sr}(z)$ could be an infinite Laurent expansion. Hence, $P(z)$ is called a *Laurent matrix*.

The existence of a matrix $P^H(z^{-1})$ which, apart from a constant, is the inverse of $P(z)$ has an interesting implication. Since the inverse of $P(z)$ is the adjoint of $P(z)$ divided by the determinant of $P(z)$, DET $P(z) \neq 0$. Thus, $P(z)$ must be full rank for all $|z| = 1$. If $P(z)$ is FIR, and invertible, then $P^H(z^{-1})$ can be made realizable by inserting sufficient delay. The minimum delay is the *McMillan degree* of a rational matrix.

Let $N = 2$. Then M reduces to the Haar matrix [[1 1][1 −1]] and $MP(z)$ is the column vector $[[P_0(z) + P_1(z)][P_0(z) - P_1(z)]]$ which we define to equal $H(z) = [[H_0(z)][H_1(z)]]$. $H(z)$ is the analysis filter and is $H^H(z^{-1})$ the synthesis filter. The transfer function of the system is $H^H(z^{-1})H(z)$ and equals

$$H_0^*(z^{-1})H_0(z) + H_1^*(z^{-1})H_1(z) = 2[P_0^*(z^{-N})P_0(z^N) + P_1^*(z^{-N})P_1(z^N)] = 2c \ .$$

Now if $H_0(z)$ and $H_1(z)$ are chosen to be a *quadrature mirror filter* (QMF) pair so that $H_1(z) = H_0(-z)$, then the transfer function becomes $H_0^*(z^{-1})H_0(z) + H_0^*(-z^{-1})H_0(-z)$. These relations can be computed symbolically, using the commands in Chapter 25.

One method of implementing a $N = 2^n$ sub-band decomposition is by building a tree of lowpass–highpass sub-band analysis and synthesis filters. In this case, discrete wavelet transforms and polyphase DFTs are essentially the same. The differences are associated with how one descends in resolution between stages of the tree. For more detail, see Gopinath. After a short digression, we will generalize these results.

INTERFERENCE CANCELLATION APPLICATIONS

Observe that while perfect reconstruction is an important requirement in coding and data compression applications, it is less important in adaptive interference cancellation (AIC) applications where sub-band isolation is far more important. This is due to the fact that the spectrum of the input signal is modified by the AIC, making the perfect cancellation of aliasing components impossible. In AIC applications, the transform simply provides a computationally efficient linear convolution engine and the polyphase filter improves the sub-band isolation. The transmux interference canceler software on pages 388 through 390 provides linear convolution, but the polyphase filter design does not satisfy the perfect reconstruction orthogonality requirements. However, it does provide linear convolution with over 30 dB sub-band isolation. The classic methods of providing linear convolution with DFTs:

1. Factor of 2 zero pad
2. Overlap and save
3. Overlap and add

are quite intuitive and easy to understand with a few simple pictures. However, achieving linear convolution with *lapped transforms* (DFT with polyphase filter) is not as intuitive for higher genus wavelet and Fourier transforms. What is true is that, for equal filter and data block sizes, the AIC transform convolver should operate at least a factor of 2 above critical decimation as do the linear convolution commands and programs in Chapter 26. This factor of 2 provides the anti-aliasing provision required to extract linearly convolved outputs. This factor of 2 is built into the programs on pages 388 through 390. They show how it may be implemented.

With the availability of the multishot optimum (minimum probability of error) Radon–Nikodym derivative sequential demodulation technology, reducing the interference to an innovations process is jointly optimum. One can design the best AIC, since there are no signal of interest (SOI) AIC distortion or intersymbol interference issues to consider, even with multi-million tap filters. Since the AIC never "sees" the SOI, the interference canceler is easy to adapt.

GENERALIZED WAVELET TRANSFORMS

The above transformation matrices for sub-band decomposition play two roles relative to wavelets. The first is the notion of a generalized wavelet transform unifying a collection of linear transformations and approximations. The second is that of providing the scaling and wavelet vectors for the calculation of wavelets. Observe that the perfect reconstruction properties follow from the N × gN coefficient matrix A, defined above. Gopinath states that *all orthogonal and biorthogonal wavelet bases with compactly supported wavelets are special cases of FIR filter bank theory.* Using the terminology of Heller, we define a complex N × N *Haar wavelet matrix* A to be one which satisfies the orthogonal row condition $AA^H = NI$ and the linear condition

$$\sum_{n=0}^{N-1} A_{mn} = N\delta_{m0} .$$

The above defined DFT matrix M satisfies these conditions, for example. Now Heller shows that *an N × N complex matrix A is a Haar wavelet matrix if and only if*

$$A = \begin{bmatrix} 1 & 0 \\ 0 & U \end{bmatrix} h$$

where U is an (N – 1) × (N – 1) unitary matrix and h is the canonical Haar matrix of rank N defined by

$$h = \begin{bmatrix} 1 & 1 & \cdots & \cdots & \cdots & \cdots & 1 \\ -(N-1)\sqrt{\frac{1}{N-1}} & \sqrt{\frac{1}{N-1}} & \cdots & \cdots & \cdots & \cdots & \sqrt{\frac{1}{N-1}} \\ \vdots & \ddots & \ddots & \cdots & \cdots & \cdots & \vdots \\ 0 & \cdots & 0 & -s\sqrt{\frac{N}{s^2+s}} & \sqrt{\frac{N}{s^2+s}} & \cdots & \sqrt{\frac{N}{s^2+s}} \\ \vdots & \cdots & \cdots & \cdots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 & -\sqrt{\frac{N}{2}} & \sqrt{\frac{N}{2}} \end{bmatrix}$$

where s = (N – k) and k = 1, 2, . . ., N – 1 are the row numbers of the matrix. In general, the top row of a *wavelet matrix* A is called the *scaling vector* or *lowpass vector* and the other rows are the *wavelet vectors* or *bandpass (highpass for N = 2) vectors.* The program CHAAR on the next page computes the canonical Haar matrix h.

```
« → N « N ONE V→L IF N 1 > THEN DUP → V « N 1 - 1 FOR
  S N S DUP SQ + √ DUP S ONE × SWAP S × NEG S 1 + IVAL
  V LZFIL DROP -1 STEP » END N →LIST M←SO REV→ » »
```

This is a particularly normalized upper Hessenberg form. The programs below compute several example Haar wavelet matrices. Use the command **LSOVL** to compute U. COSM computes the discrete cosine transform (DCT) matrix.

```
« → N « N ONE V→L 1 N 1 - FOR R 0 N 1 - FOR C '√2×
COS(R×(2×C+1)×π/2/N)' →NUM NEXT N →LIST NEXT N →LIST M←SO » »
```

Another example is the normalized discrete Chebyshev polynomials. CBYM uses command **CHEBY** to compute this example Haar wavelet matrix.

```
« → N « 0 N 1 - FOR n N 1 - n CHEBY → L « 0 N 1 - FOR m 0
    n FOR k m k PERMF NEXT n 1 + →LIST L SDOT NEXT » N
    →LIST '√((2×n+1)×(N-1)!/(N+n)!×PERMF(N-1,n))' →NUM SMPY NEXT
    N →LIST M←SO » »
```

Walsh matrices are Hadamard matrices which have been ordered in a specific way. These are *flat wavelet matrices*, since all of the elements have the same absolute value. Using the tensor (Kronecker) product program KPRD, HDM computes some example Hadamard matrices. Use the Chapter 20 and 21 commands for reordering.

```
HDM: « → N « IF N 1 == THEN [[ 1 ]] ELSE [[ 1  1 ] [1 -1]] IF N 2 >
    THEN → M « M 2 N START M KPRD NEXT » END END » »
```

```
KPRD: « MN M1 « MN SIZE EVAL → R C « 1 R FOR J 1 C FOR K
    MN J K 2 →LIST GET M1 × IF K 1 > THEN CCMB END NEXT IF
    J 1 > THEN RCMB END NEXT » » »
```

As stated above, the wavelet matrix A is in general N × gN, when P(z) is FIR.

$$A = \begin{bmatrix} a_0^0 & a_1^0 & a_2^0 & \cdots & a_{gN-1}^0 \\ a_0^1 & a_1^1 & a_2^1 & \cdots & a_{gN-1}^1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_0^{N-1} & a_1^{N-1} & a_2^{N-1} & \cdots & a_{gN-1}^{N-1} \end{bmatrix}$$

In order that P(z) be a paraunitary Laurent matrix, A must satisfy

$$\sum_{k=-\infty}^{\infty} \bar{a}_{k+Nr}^{s} \, a_{k+Nr'}^{s'} = c\delta_{rr'}\delta_{ss'} \qquad \bar{a} = a^{*}, \quad a_k^{s} = 0 \text{ for } k < 0 \text{ and } k \geq gN \; .$$

Heller defines a *rank N, genus g, N × gN wavelet matrix* as one satisfying the normalized paraunitary Laurent matrix conditions with c = N, and the linear condition

$$\sum_{r=0}^{N-1} p_{sr}(1) = \sum_{k=0}^{gN-1} a_k^{s} = N\delta_{s0} \; .$$

An example is Daubechies' wavelet matrix of rank 2 and genus 2.

$$A = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \frac{1}{4} \begin{bmatrix} \sqrt{3} & 3 & -\sqrt{3} & 1 \\ -1 & -\sqrt{3} & -3 & \sqrt{3} \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1+\sqrt{3} & 3+\sqrt{3} & 3-\sqrt{3} & 1-\sqrt{3} \\ -1+\sqrt{3} & 3-\sqrt{3} & -3-\sqrt{3} & 1+\sqrt{3} \end{bmatrix}$$

where we note that the Haar (paraunitary) transform of her coefficients results in a matrix which also satisfies the paraunitary Laurent conditions with c = 1, is a QMF pair, but fails to satisfy the linear condition. Thus, finding "good" coefficients is not trivial. Row sums are easily performed with the **RSUM** command, and symbolic Laurent matrices may be constructed and tested using the commands in Chapter 25.

The significance of the above conditions on the elements of the A matrix extends beyond perfect reconstruction filter banks. These conditions give rise to *tight frames*, that is, a set of scaling and wavelet functions that behave like orthonormal bases relative to expansions and expansion coefficients. See the wavelet tight frame theorem in Gopinath and the discussion in Alpert. Research in this area is still evolving.

An example of an orthogonal transformation used in digital processing which is not a Haar wavelet matrix is the Karhunen–Loeve transform (KLT). The KLT is essentially the Schur decomposition. In practice, this decomposition is normally applied to a covariance matrix which is Hermitian Toeplitz. The programs on page 387 provide this capability. Principal factor (component) analysis is the process of representing (approximating) a signal or data using the principal eigenvalues and associated eigenvectors. The large eigenvalues are the principal ones and the small ones are thrown away to deflate the space. Nearest–neighbor classification algorithms are cursed by high dimensionality, so reducing the size of the space is important.

Binomial filters are used in coding and are related to discrete Hermite sequences and transforms (DHT). The theory is in Haddad. Programs are available. See Appendix A. Neither the KLT nor the DHT satisfy the linear wavelet matrix condition.

If s(x) is the scaling function and $w_m(x)$ for m = 1, 2, . . ., N – 1, are the wavelet functions, the coefficients of the scaling vector (top row) can be used to solve for s(x). The coefficients of the other rows with s(x) can then be used to solve for the wavelet functions. Let N = 2 and the wavelet matrix A equal

$$A = \begin{bmatrix} a_0 & a_1 & \cdots & a_{2g-1} \\ b_0 & b_1 & \cdots & b_{2g-1} \end{bmatrix}.$$

Then we solve

$$s(x) = \sum_{k=0}^{2g-1} a_k s(2x - k) \qquad w(x) = \sum_{k=0}^{2g-1} b_k s(2x - k) .$$

See Alpert and Gopinath. For a comparison of wavelet, short-time Fourier transforms (STFT), and various time-frequency representations (TFR), see Hlawatsch. Compactly supported wavelets and transforms are easily computed with MATHLIB. Wavelet exploration programs and a tutorial on computation are available. See Appendix A.

Alpert, B., "Wavelets and Other Bases for Fast Numerical Linear Algebra," *Wavelets: Theory and Applications*, Chui, C., ed, Cambridge, Academic Press, 1991.

Daubechies, I., "Orthonormal Bases of Compactly Supported Wavelets," *Comm. Pure Appl. Math.*, 41, 1988.

Gopinath, R., and Burrus, C., "Wavelet Transforms and Filter Banks," *Wavelets: Theory and Applications*, Chui, C., ed, Cambridge, Academic Press, 1991.

Granata, J., Conner, M., and Tolimieri, R., "The Tensor Product: A Mathematical Programming Language for FFTs," *IEEE ASSP Magazine*, Vol 9, No 1, January, 1992.

Haddad, R., and Parsons, T., *Digital Signal Processing*, New York, Computer Science Press, 1991.

Heller, P., Resnikoff, H., and Wells, R., "Wavelet Matrices and the Representation of Discrete Functions," *Wavelets: Theory and Applications*, Chui, C., ed, Cambridge, Academic Press, 1991.

Hlawatsch, F., and Boudreaux-Bartels, G., "Linear and Quadratic Time–Frequency Signal Representations," *IEEE Signal Processing Magazine*, New York, Vol. 9, No. 2, April, 1992.

Holland, J., *Statistical Analysis of Analog–Digital–Analog Systems*, Ph.D. Dissertation, Stanford University, March, 1974.

Temperton, C., "Self-Sorting In-Place Fast Fourier Transform," *SIAM J. SCI. STAT. COMPUT*, Vol 12, No 4, July, 1991.

IEEE Transactions on Information Theory, New York, Vol. 38, No. 2, March, 1992, special issue on wavelet transforms and multiresolution signal analysis.

APPENDIX H

EVALUATING COMMANDS, HP 48 MENUS AND KEYBOARDS, AND PROGRAMMING TUTORIAL

INTRODUCTION

Learning to evaluate with simple programs on the HP 48 is very important, because if you get a strange answer, you will have a written record of your keystrokes to review.

This appendix gives a quick overview of how to evaluate commands, build temporary menus, make user keyboard assignments, and do programming on the HP 48.

The HP 48 has 12 keyboards. The six system ones are

1. UNSHIFTED
2. ALPHA SHIFTED
3. LEFT-SHIFTED (YELLOW)
4. ALPHA LEFT-SHIFTED
5. RIGHT-SHIFTED (BLUE)
6. ALPHA RIGHT-SHIFTED

The six user ones are the same, but the HP 48 must first be put into USR mode by pressing LEFT-SHIFTED ALPHA once to press a single user key, or twice to stay in the user keyboard mode until LEFT-SHIFTED ALPHA is pressed a third time to return to the system keyboards. Upper- and lower-case English characters are on keyboards 2 and 4, while Greek characters are on keyboard 6. The keyboard overlay kit is a useful thing to have because the keyboard layout takes time to learn. Page 52 of the HP 48 owner's manual shows most of the unlabeled keys on the HP 48 keyboard. In particular menus, the key definitions do change, as explained in the owner's manual.

MENUS

By MENU, we mean the displayed definitions of the unshifted top six white HP 48 keys which appear at the bottom of the HP48 graphics display. The HP 48 has over 60 system menus, and MATHLIB has 34 application menus. The key strokes LIBRARY MATH MENUE, where LIBRARY is the left-shifted ▲ key, take you into the MATHLIB system of menus. User menus can also be built as shown below. Each menu may have one or more pages which are accessed by pushing the NXT key.

OBJECTS AND COMMANDS

Everything from a number, an equation, a program, an array, to a list is an object on the HP 48. User objects are stored in directories and displayed in the VAR menu by pushing the VAR key. The key sequence 2 ' T W O STO stores the number 2 in the variable TWO, which is then displayed in the VAR menu.

A command is a programmable operation. There are 1,110 HP 48 and MATHLIB commands. User programs are user commands. To compute on the HP 48, you evaluate. Evaluating a program runs the program. Pushing the TWO key in the VAR menu evaluates TWO and puts the contents, the number 2, on Level 1 of the stack.

EVALUATING COMMANDS

Here are six ways to evaluate sin(2) = .909297426826 on the HP 48. The first four use Reverse Polish Notation (RPN), where the argument is placed on the stack and then replaced by the result. The last two use the ' key to define algebraic notation.

- Push the 2 key, then the ENTER key, then the SIN key.
- Push the 2 key, then the SIN key.
- Push the 2 key, the ENTER key, then the S key, then the I key, then the N key, then the ENTER key.
- Push the 2 key, the ENTER key, then the S key, then the I key, then the N key, then the EVAL key.
- Push the ' key, then the SIN key, then the 2 key, then the ENTER key, then the EVAL key.
- Push the ' key, then the SIN key, then the 2 key, then the EVAL key.

If you are getting a different answer, you probably need to push the left-shifted 1 key to put the HP 48 into radians mode, with RAD displayed in the display status area.

As if these were not enough, how about these algebraic approaches:

- Push the ' key, then the S key, then the I key, then the N key, then the () key, then the 2 key, then the ENTER key, then the EVAL key,

- Push the ' key, then the S key, then the I key, then the N key, then the () key, then the 2 key, then the EVAL key,

where the S I N is alpha SIN, alpha CST, alpha STO, and () the is left-shifted ÷ key.

Now in case you do not like any of these eight ways, there is nearly a countable infinite number of alternatives. For example, type in the program « → z 'SIN(z)' » and store it in a variable called MYSIN. The key strokes are:

« » → z SPC ' SIN z ENTER ' M Y S I N STO

where « » is the left-shifted – key and → is the alpha right-shifted 0 key. Then by substituting MYSIN for SIN in the above eight ways, we have created eight more.

Now suppose the SIN key is broken, you can not remember that MYSIN evaluates the SIN function, and you do not wish to keep typing S I N. Then type in the program « { SIN } TMENU » and store it in a variable MYMENU, for example.

« » {} SIN ▶ T M E N U ENTER ' M Y M E N U STO

where the ▶ key moves the pointer right one character so that TMENU is after the closing brace } and thus not inside the list. Then each time you push the MYMENU key or:

- Push the ' key, then the MYMENU key, then the ENTER key, then the EVAL key,
- Push the ' key, then the MYMENU key, then the EVAL key,
- ETC . . . ETC . . . ETC . . .,

you evaluate the command MYMENU, which redefines the top six keys of the keyboard and places SIN above the first key. Now you can evaluate the SIN function by pressing the 2 key, followed by pressing this new SIN key or ETC . . . ETC . . . ETC

Now you may not like this approach because you must go to the VAR directory to push the MYMENU key. The program « { SIN } 'CST' STO » will also create a menu with SIN in it, but you need only press the CST key on the keyboard to access the menu.

If you are still having trouble evaluating the SIN function, why not assign a user key to it? There are six (planes of) user keyboards, and you can assign SIN to any key of any of the six user keyboards. The program « { SIN 43.1 } STOKEYS » makes the TAN key on the unshifted user keyboard also the SIN function. Now you may push the 2 key, then the USR key (left-shifted alpha) to activate the user keyboard mode, and finally push the TAN key to evaluate sin(2), or . . ..

Still having trouble? Try: ENTRY 2 SIN ENTER

ENTRY is the right-shifted α key. The above evaluation techniques apply to all the MATHLIB commands in addition to most of the HP 48 algebraic commands. Algebraic commands are those for which the notation 'SIN(2)' works and are called functions on page 42 of the HP 48 owner's manual. All of the MATHLIB commands are algebraic, and since each provides one or more operational capabilities, each command provides one or more functions in the common English language definition of function.

EVALUATION AND THE STACK

The HP 48 does not have a real stack such as in the old days of calculators. The HP 48 stack is an ordered display of data pointers showing objects available to commands as arguments for immediate evaluation. In the above examples, placing 2 on the stack made 2 the next available argument to any single argument command that the user might choose to evaluate by pushing the SIN key, for example. Alternatively, while 2 is on the stack, we can add below it SIN, followed by either ENTER or EVAL, which evaluates the SIN command, which in turn looks above in the stack for its argument. The algebraic notation 'SIN(2)' put on the stack, followed by pushing the EVAL key, is interpreted as "evaluate the SIN command whose argument is 2."

More complicated algebraics are interpreted similarly. The HP 48 interpretation of the algebraic 'SIN(2)×PERMF(4,3)' is to evaluate the SIN command with argument 2 and place the result in temporary memory. Then evaluate MATHLIB command PERMF with first argument of 4 and second argument of 3, and finally, multiply the results. Thus, 'SIN(2)×PERMF(4,3)' EVAL is equivalent to the keystrokes:

2 SIN 4 ENTER 3 ENTER P E R M F ENTER ×

Hence, the HP 48 is a computer with a CPU and memory. It is also a calculator since it has an immediate execution capability. The stack is an ordered display of the arguments available for immediate execution which is called command evaluation. In RPN, commands take their arguments (which are called objects) in order from the stack.

PROGRAMMING THE HP 48

In its most basic sense, programming on the HP 48 is simply wrapping French quotes, «», around a series of keystrokes. The program « 2 SIN 4 3 PERMF × » evaluates the above equation. Using algebraic notation, the program « 2 4 3 → A B C 'SIN(A)×PERMF(B,C)' » also evaluates it where the notation → A B C defines the local variables A, B, C. This program could also be written with explicit evaluation as « 2 4 3 → A B C « 'SIN(A)×PERMF(B,C)' EVAL » ». To generalize the program for any inputs, use « → A B C 'SIN(A)×PERMF(B,C)' », where now the user must place on the stack the required arguments 2, 4, and 3 before executing (evaluating) this program. Alternatively, « → A B C « A SIN B C PERMF × » » computes the same result. Over 700 programming examples are available for MATHLIB, as discussed in Appendix A. See also Part 4 of the HP 48 owner's manual.

CUSTOM MENUS AND KEYBOARDS

The program « { PADD PSUB PMPY PDVD PLDVD " " PDERV RDERV PINTG } TMENU » stored in a variable PMENU will bring up a two page polynomial menu each time PMENU is evaluated. The first page contains five algebra commands, and the second page (push NXT) contains three calculus commands. The same list stored in a variable 'CST' will provide the same menu each time you push the CST key, until the variable 'CST' is purged or you move to another directory.

The program « { PADD 21.3 PSUB 22.3 PMPY 23.3 PDVD 24.3 PLDVD 25.3 PDERV 21.6 RDERV 22.6 PINTG 23.6 } STOKEYS » assigns the five algebra operations to the MTH, PRG, CST, VAR, and ▲ keys on the right-shifted user keyboard and the three calculus commands to the MTH, PRG, and CST keys on the user alpha right-shifted keyboard. The program « 0 DELKEYS {S} STOKEYS » will restore all the user keyboard definitions to the standard ones. Similarly, the program « { PMENU 21.1 } STOKEYS » assigns the above polynomial temporary menu example to the unshifted user MTH key so that USR MTH brings up the polynomial menu.

The program « { MENUE 21.1 } STOKEYS » assigns the unshifted user keyboard MTH key to the MATHLIB command MENUE. Provided you are not in EDIT mode, pushing this key will evaluate MENUE, which takes you into the MATHLIB system of 34 applications menus. To get to the MATHLIB menus during editing, see page 16.

For more detail on customizing the HP 48, see Chapter 15 of the owner's manual. I recommend that you never use command ASN, but always use STOKEYS, even for one key as in the above MENUE example. ASN does not work properly my HP 48s.

OPERATIONAL USE OF MENUS

Menus serve two basic purposes on the HP 48. Since pushing the menu key evaluates the object stored there, it is a convenient way of computing. You simply place the arguments in order on the stack and push the key. It is also useful in programming as a typing aid. The following keystrokes factor the number 123456.

LIBRARY MATH MENUE NUMB ENTRY 1 2 3 4 5 6 FACTR ENTER

SUBROUTINES

The HP 48 is a Lisp-like computer. Any program can call any other program. Any program can also call itself.

CONDITIONAL STRUCTURES

- IF *test-clause* THEN *true-clause* END
- IF *test-clause* THEN *true-clause* ELSE *false-clause* END
- CASE *test-clause-1* THEN *true-clause* END
  test-clause-2 THEN *true-clause* END

  test-clause-n THEN *true-clause* END
  optional-default-clause END

LOOP STRUCTURES

- *start finish* START *loop-clause* NEXT
- *start finish* FOR *counter loop-clause* NEXT
- *start finish* START *loop-clause increment* STEP
- *start finish* FOR counter *loop-clause increment* STEP
- DO *loop-clause* UNTIL *test-clause* END
- WHILE *test-clause* REPEAT *loop-clause* END

HP 48 MATRIX EDITOR TIP

When using the matrix editor, right-shifted ENTER, you may enter elements such as $2 - \sqrt{3}$ by typing 2 SPC 3 √ NEG + ENTER, for example.

HP48 SYSTEM MENUS

MENU	SUB MENU	KEY 1	KEY 2	KEY 3	KEY 4	KEY 5	KEY 6
MTH		*PARTS*	*PROB*	*HYP*	*MATR*	*VECTR*	*BASE*
MTH	PARTS	ABS MIN MANT RND	SIGN MAX XPON TRNC	CONJ MOD IP MAXR	ARG % FP MINR	RE %CH FLOOR	IM %T CEIL
MTH	PROB	COMB UTPC	PERM UTPF	! UTPN	RAND UTPT	RDZ	
MTH	HYP	SINH EXPM	ASINH LNP1	COSH	ACOSH	TANH	ATANH
MTH	MATR	CON ABS	IDN RNRM	TRN CNRM	RDM	DET	RSD
MTH	VECTR	XYZ V→	R∠Z →V2	R∠∠ →V3	CROSS D→R	DOT R→D	ABS
MTH	BASE	HEX RL SL AND	DEC RR SR OR	OCT RLB SLB XOR	BIN RRB SRB NOT	STWS R→B ASR	RCWS B→R
PRG		*STK*	*OBJ*	*DSPL*	*CTRL*	*BRCH*	*TEST*
PRG	STK	OVER DUP	ROT DUP2	ROLL DUPN	ROLLD DROP2	PICK DRPN	DEPTH
PRG	OBJ	OBJ→ R→C SIZE PUT	EQ→ C→R POS GET	→ARRY DTAG REPL PUTI	→LIST →UNIT SUB GETI	→STR TYPE NUM	→TAG VTYPE CHR
PRG	DSPL	PICT PIXON →GROB →LCD	PVIEW PIXOFF BLANK LCD→	LINE PIX? GOR CLLCD	TLINE PX→C GXOR DISP	BOX C→PX REPL FREEZE	ARC SIZE SUB TEXT
PRG	CTRL	DBUG INPUT DOERR	SST PROMPT ERRN	SST↓ DISP ERRM	NEXT MENU ERR0	HALT WAIT BEEP	KILL KEY OFF
PRG	BRCH	IF THEN ELSE	CASE END IFERR	START NEXT IFT	FOR STEP IFTE	DO UNTIL	WHILE REPEAT
PRG	TEST	AND == SF	OR ≠ CF	XOR < FS?	NOT > FC?	SAME ≤ FS?C	TYPE ≥ FC?C
PRINT		PR1 DELAY	PRST OLDPRT	PRSTC	PRLCD	PRVAR	CR

HP48 SYSTEM MENUS

MENU	SUB MENU	KEY 1	KEY 2	KEY 3	KEY 4	KEY 5	KEY 6
I/O		SEND RECN XMIT	RECV PKT SRECV	SERVER KERRM STIME	KGET SBRK	FINISH OPENIO BUFLEN	*SETUP* CLOSEIO
I/O	SETUP	IR/W	ASCII	BAUD	PARIT	CKSM	TRANSIO
LEFT-SHIFTED MODES		STD STK DEG HEX	FIX ARG RAD DEC	SCI CMD GRAD OCT	ENG CNCT XYZ BIN	SYM ML R∠Z FM,	BEEP CLK R∠∠
RIGHT-SHIFTED MODES		ASN TMENU FS?	STOKEYS RCLMENU FC?	RCLKEYS STOF FS?C	DELKEYS RCLF FC?C	MENU SF	CST CF
LEFT-SHIFTED MEMORY		MEM TVARS MERGE	BYTES PVARS FREE	VARS NEWOB ARCHIVE	ORDER LIBS RESTORE	PATH ATTACH PGDIR	CRDIR DETACH
RIGHT-SHIFTED MEMORY		STO+ SINV	STO− SNEG	STO× SCONJ	STO/	INCR	DECR
EDIT		←SKIP	→SKIP	←DEL	→DEL	INS	↑STK
SOLVE		*SOLVR*	ROOT	NEW	EDEQ	STEQ	CAT
PLOT		*PLOTR*	*PTYPE*	NEW	EDEQ	STEQ	CAT
PLOT	PTYPE	FUNCTION HISTOGRAM	CONIC SCATTER	POLAR	PARAMETRIC	TRUTH	BAR
PLOT	PLOTR	ERASE DEPN AXES	DRAW *PTYPE* DRAX	AUTO RES LABEL	XRNG CENT ×H	YRNG SCALE ×W	INDEP RESET PDIM
ALGEBRA		COLCT ↑MATCH	EXPAN ↓MATCH	ISOL |	QUAD APPLY	SHOW QUOTE	TAYLR →Qπ
LEFT-SHIFTED TIME		*SET* DATE+ →HMS	*ADJST* DDAYS HMS→	*ALRM* DATE HMS+	ACK TIME HMS−	ACKA TSTR	CAT TICKS
L TIME	ADJST	HR+ CLKADJ	HR−	MIN+	MIN−	SEC+	SEC−
L TIME	ALRM	>DATE STOALARM	>TIME RCLALARM	A/PM DELALARM	EXEC FINDALARM	*RPT*	SET
L TIME	ALRM RPT	WEEK	DAY	HOUR	MIN	SEC	NONE
L TIME	SET	→DATE	→TIME	A/PM	12/24	M/D	

HP48 SYSTEM MENUS

MENU	SUB MENU	KEY 1	KEY 2	KEY 3	KEY 4	KEY 5	KEY 6
STAT		Σ+ TOT XCOL LR ΣX	CLΣ MEAN YCOL PREDX ΣY	NEW SDEV BARPL PREDY ΣX^2	EDITΣ MAXΣ HISTP CORR ΣY^2	STOΣ MINΣ SCATR COV ΣX×Y	CAT BINS ΣLINE *MODL* NΣ
STAT	MODL	LIN	LOG	EXP	PWR	BEST	
LEFT-SHIFTED UNITS		*LENG* *FORCE* *ANGL*	*AREA* *ENRG* *LIGHT*	*VOL* *POWR* *RAD*	*TIME* *PRESS* *VISC*	*SPEED* *TEMP*	*MASS* *ELEC*
L UNITS	LENG	m Mpc nmi mil	cm pc miUS µ	mm lyr chain Å	yd au rd fermi	ft km fath	in mi ftUS
L UNITS	AREA	m^2 km^2	cm^2 ha	b a	yd^2 mi^2	ft^2 miUS^2	in^2 acre
L UNITS	VOL	m^3 l ml bbl	st galUK cu bu	cm^3 galC ozfl pk	yd^3 gal ozUK fbm	ft^3 qt tbsp	in^3 pt tsp
L UNITS	TIME	yr	d	h	min	s	Hz
L UNITS	SPEED	m/s c	cm/s ga	ft/s	kph	mph	knot
L UNITS	MASS	kg ton u	g tonUK mol	lb t	oz ozt	slug ct	lbt grain
L UNITS	FORCE	N	dyn	gf	kip	lbf	pdl
L UNITS	ENRG	J therm	erg Mev	Kcal eV	cal	Btu	ft×lbf
L UNITS	POWR	W	hp				
L UNITS	PRESS	Pa inHg	atm inH2O	bar	psi	torr	mmHg
L UNITS	TEMP	°C	°F	K	°R		
L UNITS	ELEC	V Fdy	A H	C mho	Ω S	F T	W Wb
L UNITS	ANGL	°	r	grad	arcmin	arcs	sr
L UNITS	LIGHT	fc cd	flam lam	lx	ph	sb	lm

HP48 SYSTEM MENUS

MENU	SUB MENU	KEY 1	KEY 2	KEY 3	KEY 4	KEY 5	KEY 6
L UNITS	RAD	Gy R	rad	rem	Sv	Bq	Ci
L UNITS	VISC	P	St				
RIGHT-SHIFTED UNITS	CONVERT	UBASE	UVAL	UFACT	→UNIT		
MATRIX		EDIT +ROW	VEC −ROW	←WID +COL	WID→ −COL	GO→ →STK	GO↓ ↑STK
▲		ECHO DUPN	VIEW DRPN	PICK KEEP	ROLL LEVEL	ROLLD	→LIST
GRAPH		*ZOOM*	Z-BOX	CENT	COORD	LABEL	*FCN*
GRAPH	ZOOM	XAUTO	X	Y	XY		EXIT
GRAPH	FCN	ROOT F(X)	ISECT F'	SLOPE NXEQ	AREA	EXTR	EXIT
RIGHT-SHIFTED TIME		PURG		EXECS	EDIT	→STK	VIEW
	SAME AS LEFT-SHIFTED TIME AND THEN CAT						
VAR	UNSHIFTED = EVALUATE LEFT-SHIFTED = STORE CONTENTS RIGHT-SHIFTED = RECALL CONTENTS						

USER APPLICATION MENUS

NAME	KEY 1	KEY 2	KEY 3	KEY 4	KEY 5	KEY 6

MATHLIB APPLICATION MENUS

MENU	KEY 1	KEY 2	KEY 3	KEY 4	KEY 5	KEY 6
MENUE OR UPDIR	*PLOT* *STAT* *PROC*	*FTNS* *PROB* *FILTR*	*NUMB* *IPROB* *WIND*	*ALGB* *BIVN* *VECTR*	*LINAG* *QUE* *VSAG*	*MATR* *SYMB* *MSAG*
PLOT	PLT1 FPLOT	PLT2 FPLT1	PLTC FPLT2	PLT1L PLT3	PLT2L ERASE	PLTCL UPDIR
FTNS	TO MATHLIB FUNCTION MENUS					
NUMB	GCD BR RZETA EXCO	LCM BPOLY CZETA ¡	FACTR BCOEF STRL1 p	FMPY ER STRL2 UPDIR	PRIME EPOLY COLCT	FIBON ECOEF EXPAN
ALGB	TALR1 FEVAL ↑MATCH H1F1 EVALC DEFLT PLDVD	TALR2 COEFL ↓MATCH DH1F1 ISOL PDERV CLIST	PADD LREV | H2F1 QUAD RDERV SRND	PSUB COLCT COEFE DH2F1 PROOT PINTG LZDEL	PMPY EXPAN APPLY EVLΣ AROOT PEQN TIFRE	PDVD EXCO QUOTE SUMΣ LROOT XEQN UPDIR
LINAG	LSOVL LSVDL TRNP CON MINV GLUD HBDD EIGNS HOUSE RROT HBDDR	LSOVR LSVDR TRNH IDN HILBT LDLTD HBDV EIGEN RHOUS CROT SVDMI	USOVL LSERL SYM? RDM RNRM AGAUS HTRDD WSQR CHOUS GROT SVDDI	USOVR LSERR HRM? RORDR CNRM GAUSS UHESD GSQR VHOUS D→M CPOLY	OSOVL SVD ORTH? CORDR RABS HRQR SCRSD TRACE PHOUS D←M AROOT	OSOVR B←M UNIT? IORDR CABS HBDU SCHRD MINOR GIVEN CHOLD UPDIR
MATR	→ROW ERWS IROW MROW EROW M→RL RDLV REV→ RDM λ?	→COL ECOLS ICOL MCOL ECOL M←RL CDLV REV↑ IDN λV?	RSORT RRWS DROW CROW RROW M→CL RSPLT RGET CON EVS?	RSRTI RCOLS DCOL CCOL RCOL M←CL CSPLT CGET DIAG UPDIR	CSORT EPSM SROW TRNP ESBM RNLV RCMB B→M COMP	CSRTI RPSM SCOL →VTR RSBM CNLV CCMB B←M HTOEP
STAT	μ MEDIN FTDV HIST CUMΣ AVAR1 ROWμ RNLV	σ MODE CDT1 SRT↑ MAVE AVAR2 COLμ RDLV	ADEV COVAR CDT2 SRT↓ VADD ACOVR ROWσ RSPLT	SKEW TTSV KST1 SRTI VSUB RNUM COLσ RCMB	KURT TTDV KST2 RNDU CTA2D CNUM VSPLT SIZE	VARM TTPS LCNT RNDN SRCTT SRND VCMB UPDIR

MATHLIB APPLICATION MENUS

MENU	KEY 1	KEY 2	KEY 3	KEY 4	KEY 5	KEY 6
PROB	UτPC WEIB UTPβ LTPH MγN	UτPF EXTV LTPβ NEGB MPN	UτPN UNIF UτηC POSN COMB	UτPT CAUCH UτηF UTKS PERM	LTPN LAPL UτηT RAYL !	UTPE UTPG BINM RICE UPDIR
IPROB	IUTPC IEXTV ILTPβ IMγN	IUTPF IUNIF IBINM UPDIR	IUTPN ICAUCH INEGB	IUTPT ILAPL IPOSN	IUTPE IUTPG IUTKS	IWEIB IUTPβ IRAYL
BIVN	PUXUY UPDIR	PUXLY	PLXUY	PLXLY	UτPN	LTPN
QUE	LM PNSM	LQM PDQW	LPM UTPE	WM UTPG	WQM POSN	PQB UPDIR
SYMB	SADD SOB→ SDERV M→SO SEVAL SCROS SABS HVSDE SRND SCNJ	SSUB →SOB SINTG M←SO SEXCO SDET STRAC RESDP SMI L1F1	SMPY SERW SADDS L→SO SVBAR SMNR S1F1 RESDA SRE UPDIR	SDOT SECOL SSUBS L←SO MAT↑ SGRD DESOL IXFRM SIM	SCHS SRRW SCOLC SGET MAT↓ SDIV PROOT COEFL TIFRE	STRN SRCOL SEXPD SPUT SNUM SCURL AROOT CLIST SSIZE
PROC	DER1 FINDP FFT CONV ZFILL VROT PHASU ICONV	DER2 FINDV IFFT CCORR HALF REFLT VTRUD UNITI	LINT FINDX TWIDL RNDU ZFIL1 VDEC MAVE RNDC	RINT FINDN BITRV RNDN HALF1 VINTP VMPY DGLIT	INDEX VΣ WL1 SPECT ZFILN SQWV VDVD PLT1	PINDX CUMΣ WL2 PHASE REDN ΔWAV VCORR UPDIR
FILTR	FTRV1 FTRVL BPOLE ΩMIN PSCAL WARP DHBRT S→ZP VLOG	FTRV2 POLEP CPOLE ΩMAX COEFL L→LP GAIN1 TF→C SRND	τVT1 FORDR θFCFS ρ→AX PROOT L→HP IXFRM C↑O PLT1	τVT2 RZINV ESOLV ρ←AX AROOT L→BP RATAP COMT PLTC	HOFω R→CL EZERO AX→ε δη→A L→BS SRE SPECT PLT3	τOFω PZINV EPOLE AX←ε BESLF BILNT SIM PHASE UPDIR
WIND	HAMM PARZ ONE TOFXL V→DB ρ→AX	GENH KAISR FIRID CHEBY V←DB ρ←AX	HANN WELC IXFRM DOCAP N→DB AX→ε	BARTL CLIPP PLDVD STRLT N←DB AX←ε	BLAC CLIPB FMAT WL1 δη→A VECTX	GAUS CLIPN MINV WL2 δη←A UPDIR

MATHLIB APPLICATION MENUS						
MENU	KEY 1	KEY 2	KEY 3	KEY 4	KEY 5	KEY 6
VECTR	V→L VREV CVAL VCMB	V←L LREV VINST LZDEL	→VTR IVAL VDEL LZFIL	SRT↑ DVAL VSUBS ONE	SRT↓ SVAL VREPL UNITI	SRTI MVAL VSPLT UPDIR
VSAG	VMAX VLN VABS VRND EINDX	VMIN VEXP VARG VFLOR VCDEL	VADD VSQ ARGU VCEIL V1F1	VSUB VSRT VTRUR RNDU V2F1	VLOG VECTX VSINV RNDN ZERO	VALOG VECTD VMOD RNDC UPDIR
MSAG	MMAX MLN RSUM MMOD MRDU M2F1	MMIN MEXP CSUM MRND MRDN UPDIR	MADD MSQ RΣ0D MFLR MRDC	MSUB MSRT CΣ0D MCEIL MRDS	MLOG MATX MABS RABS2 MRDH	MALOG MATD MARG CABS2 M1F1

MATHLIB FUNCTION MENUS						
MENU	KEY 1	KEY 2	KEY 3	KEY 4	KEY 5	KEY 6
FTNS OR UPDIR	*TRIG* *SBESL* *GHYP*	*HYP* *ELLIPI* *LGDR*	*EXPIN* *JACOB* *STRUV*	*GAMA* *THETA* *POLY*	*ERROR* *CHYPR* *MISC*	*BESEL* *PCLDR* *UPDIR*
TRIG	CSC ARG2	ACSC ATN2	SEC R→D	ASEC D→R	COT GAMMA	ACOT UPDIR
HYP	SINH CSCH EXPM	ASINH ASCSCH LNP1	COSH SECH GAMMA	ACOSH ASECH PSI	TANH COTH UPDIR	ATANH ACOTH
EXPIN	EIOX SCINT αNOZ	LIOX SINC βNOZ	E1OZ SINC2 GAMMA	ENOZ SHIZ PSI	SINT CHIZ CINTG	CINT γ UPDIR
GAMA	GAMMA INCB	PSI DNPSI	INCG INCγ	INCγ INCβ	PINCG ¡¡	BETA UPDIR
ERROR	SOFZ ERFZ CINTG	COFZ ERFCZ UPDIR	S1OZ ZOFZ	C1OZ INERFC	S2OZ GAMMA	C2OZ PSI

MATHLIB FUNCTION MENUS

MENU	KEY 1	KEY 2	KEY 3	KEY 4	KEY 5	KEY 6
BESEL	JNOZ INOZ BEN& GAMMA	J0OZ I0OX KEN& UPDIR	J10Z I1OX H1NZ	YNOX KNOX H2NZ	Y0OX K0OX YNOZ	Y10X K10X KNOZ
SBESL	SJNZ SKNZ	SYNZ AIOZ	SH1Z BIOZ	SH2Z GAMMA	SI1NZ PSI	SI2NZ UPDIR
ELLIPI	FXKP KOKP $\phi\leftarrow$X ΔZKP UPDIR	EXKP K$_i$OK$_i$ $\alpha\leftarrow$K$_i$ K$_i$OK	BXKP EOKP M$\rightarrow$K IK$_i$OK	DXKP ΠNXKP M$\leftarrow$K KP$\rightarrow$M	$\phi\rightarrow$X ZUKP K$\uparrow$KP UZKP	$\alpha\rightarrow$K$_i$ λXKP M$\uparrow$M$_i$ SNUK$_i$
JACOB	SNUK$_i$ DCUK$_i$ N$\rightarrow$D NUKP $\alpha\rightarrow$K$_i$ KOKP	CNUK$_i$ NCUK$_i$ N$\leftarrow$D DUKP $\alpha\leftarrow$K$_i$ K$_i$OK$_i$	DNUK$_i$ SCUK$_i$ N$\rightarrow$C CUKP M$\rightarrow$K KP$\rightarrow$M	CDUK$_i$ NSUK$_i$ N$\leftarrow$C SUKP M$\leftarrow$K KP$\leftarrow$M	SDUK$_i$ DSUK$_i$ N$\rightarrow$S UZKP K$\uparrow$KP AGMN	NDUK$_i$ CSUK$_i$ N$\leftarrow$S AMUKP M$\uparrow$M$_i$ UPDIR
THETA	$\phi\rightarrow$X θSUK$_i$ θUKP θ1ZQ TSUK$_i$ T1UQ LNθ1	$\phi\leftarrow$X θCUK$_i$ θ1UK$_i$ θ2ZQ TCUK$_i$ T2UQ LNθ2	UZKP θDUK$_i$ HUKP θ3ZQ TDUK$_i$ T3UQ LNθ3	SNUK$_i$ θNUK$_i$ H1UK$_i$ θ4ZQ TNUK$_i$ T4UQ LNθ4	$\alpha\rightarrow$K$_i$ $\varepsilon\alpha\rightarrow$U KOKP K$\uparrow$KP KP$\rightarrow$Q R$\rightarrow$D KP$\rightarrow$M	$\alpha\leftarrow$K$_i$ $\varepsilon\alpha\leftarrow$U K$_iOK_i$ M$\uparrow$M$_i$ Dθ1K$_i$ D$\rightarrow$R UPDIR
CHYPR	MABZ JVOZ BEV&	DNM YVOZ KEV&	UABZ IVOZ MKμZ	DNU KVOZ WKμZ	INCGC H1VZ GAMMA	INCγC H2VZ UPDIR
PCLDR	UOAX GAMMA	VOAX PSI	WOAX TNP	DVOZ TI	EV0Z TNI	EV1Z UPDIR
GHYP	F2F1	D2F1	FOGC	INCBH	GAMMA	UPDIR
LGDR	PμVX	QμVX	PQUP	PμVZ	QμVZ	UPDIR
STRUV	HVOZ	LVOZ	YVOZ	IVOZ	GAMMA	UPDIR
POLY	P$\alpha\beta$X SOFX HOX	GPQX TSFX HEOX	CαOX USFX UPDIR	TOFX POFX	UOFX LαOX	COFX LOFX
MISC	CSERS COEFL RATAP SDOT TIFRE	TALR1 COEFV REVAL LREV UNIQE	TALR2 PMAT LREG PERMF SSORT	FEVAL SRND FSTAT POCH ZSORT	EXCO PEQN TSTAT MQ SIGNP	TIMIT XEQN LSSOV MULTI UPDIR
UPDIR	TO MATHLIB APPLICATION MENUS					

ALPHABETIC COMMAND SUMMARY AND MENU INDEX

MATHLIB COMMAND	DESCRIPTION	DEFINITION MENU (ALL ARE IN MAIN)
ACOT	ARC COTANGENT	TRIG
ACOTH	HYPERBOLIC ARC COTANGENT	HYP
ACOVR	ONE-WAY ANALYSIS OF COVARIANCE	STAT
ACSC	ARC COSECANT	TRIG
ACSCH	HYPERBOLIC ARC COSECANT	HYP
ADEV	STATISTICAL ABSOLUTE DEVIATION	STAT
AGAUS	APPLY GAUSSIAN TRANSFORMATION	LINAG
AGMN	ARITHMETIC-GEOMETRIC MEAN	JACOB
AIOZ	AIRY FUNCTION Ai(z)	SBESL
AMUKP	COMPUTE ELLIPTIC FUNCTION am(u, k)	JACOB
ARG2	ARGUMENT IN [0, 2π] RANGE	TRIG
ARGU	PHASE UNWRAP ARGUMENT VECTOR IN RADIANS	VSAG
AROOT	LAGUERRE COMPLEX POLYNOMIAL ROOT SOLVER	ALGB
ASEC	ARC SECANT	TRIG
ASECH	HYPERBOLIC ARC SECANT	HYP
ATN2	FOUR-QUADRANT ARC TANGENT (ATAN2)	TRIG
AVAR1	ONE-WAY ANALYSIS OF VARIANCE	STAT
AVAR2	TWO-WAY ANALYSIS OF VARIANCE	STAT
AX→ε	a_{MAX} TO RIPPLE FACTOR	FILTR
AX←ε	RIPPLE FACTOR TO a_{MAX}	FILTR
BARTL	BARTLETT WINDOW	WIND
BCOEF	COMPUTE BERNOULLI COEFFICIENTS	NUMB
BEN&	KELVIN FUNCTION $ber_n(x) + i\ bei_n(x)$	BESEL
BESLF	BESSEL FILTER COEFFICIENT LIST	FILTR
BETA	BETA FUNCTION B(z, w)	GAMA
BEV&	GENERAL KELVIN $ber_v(z) + i\ bei_v(z)$	CHYPR

ALPHABETIC COMMAND SUMMARY AND MENU INDEX

MATHLIB COMMAND	DESCRIPTION	DEFINITION MENU (ALL ARE IN MAIN)
BILNT	BILINEAR TRANSFORM	FILTR
BINM	UPPER TAIL BINOMIAL DISTRIBUTION	PROB
BIOZ	AIRY FUNCTION Bi(z)	SBESL
BITRV	PERFORM FFT BIT REVERSAL	PROC
BLAC	BLACKMAN WINDOW	WIND
BPOLE	POLES OF BUTTERWORTH FILTER	FILTR
BPOLY	COMPUTE BERNOULLI POLYNOMIALS	NUMB
BR	COMPUTE BERNOULLI NUMBERS	NUMB
BXKP	INCOMPLETE ELLIPTIC INTEGRAL B(x, k)	ELLIPI
B←M	EXTRACT DIAGONAL FROM MATRIX	MATR
B→M	INSERT DIAGONAL VECTOR IN MATRIX	MATR
C↑O	CONTROLLABLE FORM ↑ OBSERVABLE FORM	FILTR
CαOX	GEGENBAUER POLYNOMIAL $C_n^{(\alpha)}(x)$	POLY
CΣ0D	COLUMN SUM WITH COL DELETE IF ZERO	MSAG
C1OZ	FRESNEL INTEGRAL $C_1(z)$	ERROR
C2OZ	FRESNEL INTEGRAL $C_2(z)$	ERROR
CABS	EUCLIDEAN NORM OF EACH COLUMN	LINAG
CABS2	EUCLIDEAN COLUMN NORM SQUARED	MSAG
CAUCH	UPPER TAIL CAUCHY DISTRIBUTION	PROB
CCMB	COMBINE TWO MATRICES BY COLUMNS	MATR
CCOL	COPY MATRIX COLUMN	MATR
CCORR	FFT CROSS-CORRELATION	PROC
CDLV	DE-INTERLEAVE MATRIX COLUMNS	MATR
CDT1	χ^2 TEST FOR DISTRIBUTIONS (1 DATA SET)	STAT
CDT2	χ^2 TEST FOR DISTRIBUTIONS (2 DATA SETS)	STAT
CDUKj	JACOBI ELLIPTIC FUNCTION cd(u, k)	JACOB

ALPHABETIC COMMAND SUMMARY AND MENU INDEX

MATHLIB COMMAND	DESCRIPTION	DEFINITION MENU (ALL ARE IN MAIN)
CGET	EXTRACT MATRIX COLUMN SUBSET	MATR
CHEBY	DISCRETE ORTHOGONAL CHEBYSHEV POLYNOMIAL	WIND
CHIZ	EXPONENTIAL INTEGRAL Chi(z)	EXPIN
CHOLD	CHOLESKY MATRIX DECOMPOSITION	LINAG
CHOUS	APPLY HOUSEHOLDER VECTOR TO MATRIX	LINAG
CINT	COSINE INTEGRAL Ci(z)	EXPIN
CINTG	COMPLEX NUMERICAL INTEGRATION	EXPIN ERROR
CLIPB	CLIP ALL PEAKS	WIND
CLIPN	CLIP NEGATIVE PEAKS ONLY	WIND
CLIPP	CLIP POSITIVE PEAKS ONLY	WIND
CLIST	COMPUTE POLYNOMIAL FROM ROOTS	ALGB
CNLV	INTERLEAVE MATRIX COLUMNS	MATR
CNUK$_j$	JACOBI ELLIPTIC FUNCTION cn(u, k)	JACOB
CNUM	COUNT NUMBER OF ENTRIES IN EACH COLUMN	STAT
COEFE	COEFFICIENT EVALUATION	ALGB
COEFL	ONE-DIMENSIONAL MACLAURIN SERIES (LIST OUT)	MISC
COEFV	ONE-DIMENSIONAL MACLAURIN SERIES (VECT OUT)	MISC
COFX	CHEBYSHEV POLYNOMIAL $C_n(x)$	POLY
COFZ	FRESNEL INTEGRAL C(z)	ERROR
COLμ	MEAN VALUE OF EACH MATRIX COLUMN	STAT
COLσ	STANDARD DEVIATION OF EACH MATRIX COLUMN	STAT
COMP	CREATE COMPANION MATRIX	MATR
COMT	CONTROLLABILITY-OBSERVABILITY MATRIX	FILTR
CONV	FFT CONVOLUTION	PROC
CORDR	REORDER MATRIX COLUMNS USING PIVOT VECTOR	LINAG
COT	COTANGENT	TRIG

ALPHABETIC COMMAND SUMMARY AND MENU INDEX

MATHLIB COMMAND	DESCRIPTION	DEFINITION MENU (ALL ARE IN MAIN)
COTH	HYPERBOLIC COTANGENT	HYP
COVAR	COVARIANCE OF TWO VECTORS	STAT
CPOLE	POLES OF CHEBYSHEV FILTER	FILTR
CPOLY	MATRIX CHARACTERISTIC POLYNOMIAL	LINAG
CROT	COLUMN GIVENS ROTATE	LINAG
CROW	COPY MATRIX ROW	MATR
CSC	COSECANT	TRIG
CSCH	HYPERBOLIC COSECANT	HYP
CSERS	CREATE COMPLEX SERIES FROM FUNCTION	MISC
CSORT	MATRIX COLUMN SORT ↑	MATR
CSPLT	SPLIT MATRIX BY COLUMNS	MATR
CSRTI	MATRIX COLUMN SORT ↓	MATR
CSUKj	JACOBI ELLIPTIC FUNCTION cs(u, k)	JACOB
CSUM	COMPUTE VECTOR OF COLUMN SUMS	MSAG
CTA2D	CONTINGENCY TABLE ANALYSIS	STAT
CUKP	COMPUTE C SET OF JACOBI ELLIPTIC FTNS	JACOB
CUMΣ	CUMULATIVE SUM OF VALUES	PROC
CVAL	COPY VECTOR VALUE	VECTR
CZETA	COMPLEMENTARY RIEMANN ZETA FTN	NUMB
Dθ1Kj	COMPUTE DERIVATIVE $\vartheta_1'(0, k)$	THETA
D2F1	NTH DERIVATIVE OF F(a, b, c, z)	GYHP
DCOL	DELETE MATRIX COLUMN	MATR
DCUKj	JACOBI ELLIPTIC FUNCTION dc(u, k)	JACOB
DEFLT	DEFLATE POLYNOMIAL BY DIVIDING OUT ROOT	ALGB
DER1	NUMERICAL FIRST DERIVATIVE	PROC
DER2	NUMERICAL SECOND DERIVATIVE	PROC

ALPHABETIC COMMAND SUMMARY AND MENU INDEX

MATHLIB COMMAND	DESCRIPTION	DEFINITION MENU (ALL ARE IN MAIN)
DESOL	RUNGE–KUTTA DIFFERENTIAL EQUATION SOLUTIONS	SYMB
DGLIT	DEGLITCH DATA POINTS OF VECTOR	PROC
DH1F1	SYMBOLIC NTH DERIVATIVE M(a, b, z)	ALGB
DH2F1	SYMBOLIC NTH DERIVATIVE F(a, b, c, z)	ALGB
DHBRT	DESIGN DIGITAL HILBERT TRANSFORMER	FILTR
DIAG	CREATE DIAGONAL MATRIX	MATR
DNM	NTH DERIVATIVE OF M(a, b, z)	CHYPR
DNPSI	POLYGAMMA FUNCTION $\psi^{(n)}(z)$	GAMA
DNU	NTH DERIVATIVE OF U(a, b, z)	CHYPR
DNUK$_j$	JACOBI ELLIPTIC FUNCTION dn(u, k)	JACOB
DOCAP	LEAST SQUARES APPROXIMATION WITH CHEBY	WIND
DROW	DELETE MATRIX ROW	MATR
DSUK$_j$	JACOBI ELLIPTIC FUNCTION ds(u, k)	JACOB
DUKP	COMPUTE D SET OF JACOBI ELLIPTIC FTNS	JACOB
DVAL	DELETE VECTOR VALUE	VECTR
DVOZ	PARABOLIC CYLINDER $D_v(x)$	PCLDR
DXKP	INCOMPLETE ELLIPTIC INTEGRAL D(x, k)	ELLIPI
D→M	VECTOR TO DIAGONAL MATRIX	LINAG
D←M	DIAGONAL MATRIX TO VECTOR	LINAG
E1OZ	EXPONENTIAL INTEGRAL $E_1(Z)$	EXPIN
ECOEF	COMPUTE EULER COEFFICIENTS	NUMB
ECOL	EXTRACT ENTIRE MATRIX COLUMN	MATR
ECOLS	MATRIX EXTRACT COLUMN SUBSET	MATR
EIGEN	GENERAL EIGEN DECOMPOSITION	LINAG
EIGNS	SYMMETRIC & HERMITIAN EIGEN DECOMPOSITION	LINAG
EINDX	CREATE EXPONENTIAL INDEX VECTOR	VSAG

ALPHABETIC COMMAND SUMMARY AND MENU INDEX

MATHLIB COMMAND	DESCRIPTION	DEFINITION MENU (ALL ARE IN MAIN)
EIOX	EXPONENTIAL INTEGRAL Ei(x)	EXPIN
ENOZ	EXPONENTIAL INTEGRAL $E_n(z)$	EXPIN
EOKP	COMPLETE ELLIPTIC INTEGRAL E	ELLIPI
EPOLE	POLES OF ELLIPTIC FILTER	FILTR
EPOLY	COMPUTE EULER POLYNOMIALS	NUMB
EPSM	EXTRACT PARTIAL SUBMATRIX	MATR
ER	COMPUTE EULER NUMBERS	NUMB
ERFCZ	COMPLEMENTARY ERROR FUNCTION erfc(z)	ERROR
ERFZ	ERROR FUNCTION erf(z)	ERROR
EROW	EXTRACT ENTIRE MATRIX ROW	MATR
ERWS	MATRIX EXTRACT ROW SUBSET	MATR
ESBM	EXTRACT SUBMATRIX	MATR
ESOLV	SOLVE FOR ELLIPTIC FILTER PARAMETERS	FILTR
EV0Z	PARABOLIC CYLINDER $E_v^{(0)}(z)$	PCLDR
EV1Z	PARABOLIC CYLINDER $E_v^{(1)}(z)$	PCLDR
EVALC	EVALUATE COMPLETELY	ALGB
EVLΣ	MULTIPLE UNCONDITIONAL ↑MATCH COMMAND	ALGB
EVS?	TEST IF EIGENVECTORS ARE THE SAME	MATR
EXCO	EXPAND AND COLLECT COMPLETELY	MISC
EXKP	INCOMPLETE ELLIPTIC INTEGRAL E(x, k)	ELLIPI
EXTV	UPPER TAIL EXTREME VALUE DISTRIBUTION	PROB
EZERO	ZEROS OF ELLIPTIC FILTER	FILTR
F2F1	GAUSSIAN HYPERGEOMETRIC F(a, b, c, z)	GHYP
FACTR	FACTOR INTEGER INTO PRIMES	NUMB
FEVAL	EVALUATE POLYNOMIAL APPROXIMATION	MISC
FFT	DISCRETE FOURIER TRANSFORM	PROC

ALPHABETIC COMMAND SUMMARY AND MENU INDEX

MATHLIB COMMAND	DESCRIPTION	DEFINITION MENU (ALL ARE IN MAIN)
FIBON	COMPUTE FIBONACCI NUMBERS	NUMB
FINDN	FIND LOCATION OF NEXT MINIMUM	PROC
FINDP	FIND NEXT DATA PEAK LOCATION	PROC
FINDV	FIND NEXT DATA VALLEY LOCATION	PROC
FINDX	FIND LOCATION OF NEXT MAXIMUM	PROC
FIRID	FIR DESIGN BASED ON IDEAL PROTOTYPE	WIND
FMAT	FAST POLYNOMIAL APPROXIMATION MATRIX	WIND
FMPY	MULTIPLY PRIME FACTORS	NUMB
FOGC	RATIO F(a, b, c, z) / Γ(c)	GHYP
FORDR	COMPUTE REQUIRED FILTER ORDER	FILTR
FPLOT	PLOT ONE FUNCTION WITHOUT LABELS	MAIN PLOT
FPLT1	PLOT ONE FUNCTION WITH LABELS	MAIN PLOT
FPLT2	PLOT TWO FUNCTIONS WITH LABELS	MAIN PLOT
FSTAT	LINEAR REGRESSION F STATISTICS	MISC
FTDV	F TEST FOR DIFFERENT VARIANCES	STAT
FTRV1	ANALOG AND DIGITAL FILTER RESPONSE	FILTR
FTRV2	ANALOG AND DIGITAL FILTER RESPONSE	FILTR
FTRVL	LOGARITHMIC FREQUENCY SCALE RESPONSE	FILTR
FXKP	INCOMPLETE ELLIPTIC INTEGRAL F(x, k)	ELLIPI
GAIN1	SET GAIN OF FILTER TO 1 AT A FREQUENCY	FILTR
GAMMA	GAMMA FUNCTION Γ(z)	GAMA
GAUS	GAUSSIAN WINDOW	WIND
GAUSS	COMPUTE GAUSSIAN TRANSFORMATION	LINAG
GCD	GREATEST COMMON DIVISOR	NUMB
GENH	GENERALIZED HAMMING WINDOW	WIND
GIVEN	COMPUTE GIVENS ROTATION	LINAG

ALPHABETIC COMMAND SUMMARY AND MENU INDEX

MATHLIB COMMAND	DESCRIPTION	DEFINITION MENU (ALL ARE IN MAIN)
GLUD	GAUSSIAN LU MATRIX DECOMPOSITION	LINAG
GPQX	JACOBI POLYNOMIAL $G_v(p, q, x)$	POLY
GROT	COMPUTE GIVENS ROTATION	LINAG
GSQR	GENERAL EIGEN DECOMPOSITION STEP	LINAG
H1F1	SYMBOLIC HYPERGEOMETRIC M(a, b, z)	ALGB
H1NZ	HANKEL FUNCTION $H_n^{(1)}(z)$	BESEL
H1UKj	JACOBI ETA FUNCTION $H_1(u, k)$	THETA
H1VZ	GENERAL HANKEL $H_v^{(1)}(z)$	CHYPR
H2F1	SYMBOLIC HYPERGEOMETRIC F(a, b, c, z)	ALGB
H2NZ	HANKEL FUNCTION $H_n^{(2)}(z)$	BESEL
H2VZ	GENERAL HANKEL $H_v^{(2)}(z)$	CHYPR
HALF	TRUNCATES TO HALF THE LENGTH	PROC
HALF1	TRUNCATES TO HALF THE LENGTH + 1	PROC
HAMM	HAMMING WINDOW	WIND
HANN	HANNING WINDOW	WIND
HBDD	HOUSEHOLDER BIDIAGONAL DECOMPOSITION	LINAG
HBDDR	HOUSEHOLDER BIDIAG DECOMPOSITION	LINAG
HBDU	COMPUTE HOUSEHOLDER U UNITARY MATRIX	LINAG
HBDV	COMPUTE HOUSEHOLDER V UNITARY MATRIX	LINAG
HEOX	HERMITE POLYNOMIAL $He_n(x)$	POLY
HILBT	COMPUTE HILBERT MATRIX	LINAG
HIST	COMPUTE DATA HISTOGRAM	STAT
HOFω	EVALUATE RESPONSE AT A FREQUENCY	FILTR
HOUSE	COMPUTE HOUSEHOLDER VECTOR	LINAG
HOX	HERMITE POLYNOMIAL $H_n(x)$	POLY
HRM?	TEST IF MATRIX IS HERMITIAN	LINAG

ALPHABETIC COMMAND SUMMARY AND MENU INDEX

MATHLIB COMMAND	DESCRIPTION	DEFINITION MENU (ALL ARE IN MAIN)
HRQR	HOUSEHOLDER RANK AND QR DECOMPOSITION	LINAG
HTOEP	CREATE HERMITIAN TOEPLITZ MATRIX	MATR
HTRDD	HOUSEHOLDER TRIDIAGONAL DECOMPOSITION	LINAG
HUKP	JACOBI ETA FUNCTION H(u, k)	THETA
HVOZ	COMPLEX STRUVE FUNCTION $H_v(z)$	STRUV
HVSDE	HEAVISIDE EXPANSION FORMULA	SYMB
I0OX	BESSEL FUNCTION $I_0(x)$	BESEL
I1OX	BESSEL FUNCTION $I_1(x)$	BESEL
IBINM	INVERSE BINOMIAL DISTRIBUTION	IPROB
ICAUCH	INVERSE CAUCHY DISTRIBUTION	IPROB
ICOL	INSERT COLUMN IN MATRIX	MATR
ICONV	RECURSIVE IIR CONVOLVE WITHOUT FFT	PROC
IEXTV	INVERSE EXTREME VALUE DISTRIBUTION	IPROB
IFFT	INVERSE DISCRETE FOURIER TRANSFORM	PROC
IK$_1$OK	INVERSE OF QUARTER PERIOD RATIO **K$_1$OK**	ELLIPI
ILAPL	INVERSE LAPLACE DISTRIBUTION	IPROB
ILTPβ	INVERSE BETA DISTRIBUTION	IPROB
IMγN	INVERSE MARCUM Γ_N DISTRIBUTION	IPROB
INCβ	INCOMPLETE BETA FUNCTION $B_z(a, b)$	GAMA
INCγ	INCOMPLETE GAMMA γ(a, z)	GAMA
INCγC	GENERAL INCOMPLETE GAMMA γ(a, z)	CHYPR
INCγ$_1$	INCOMPLETE GAMMA γ*(a, z)	GAMA
INCB	INCOMPLETE BETA FUNCTION $I_z(a, b)$	GAMA
INCBH	GENERAL INCOMPLETE BETA $I_z(a, b)$	GHYP
INCG	INCOMPLETE GAMMA Γ(a, z)	GAMA
INCGC	GENERAL INCOMPLETE GAMMA Γ(a, z)	CHYPR

ALPHABETIC COMMAND SUMMARY AND MENU INDEX

MATHLIB COMMAND	DESCRIPTION	DEFINITION MENU (ALL ARE IN MAIN)
INDEX	CREATE AMPLITUDE INDEX VECTOR	PROC
INEGB	INVERSE NEG BINOMIAL DISTRIBUTION	IPROB
INERFC	NTH INTEGRAL OF erfc(z) = i^n erfc(z)	ERROR
INOZ	BESSEL FUNCTION $I_n(z)$	BESEL
IORDR	COMPUTE INVERSE PIVOT VECTOR	LINAG
IPOSN	INVERSE POISSON DISTRIBUTION	IPROB
IRAYL	INVERSE RAYLEIGH DISTRIBUTION	IPROB
IROW	INSERT ROW IN MATRIX	MATR
IUNIF	INVERSE UNIFORM DISTRIBUTION	IPROB
IUTKS	INVERSE KOLMOGOROV–SMIRNOV DISTRIBUTION	IPROB
IUTPβ	INVERSE BETA DISTRIBUTION	IPROB
IUTPC	INVERSE χ^2 DISTRIBUTION	IPROB
IUTPE	INVERSE EXPONENTIAL DISTRIBUTION	IPROB
IUTPF	INVERSE F DISTRIBUTION	IPROB
IUTPG	INVERSE GAMMA DISTRIBUTION	IPROB
IUTPN	INVERSE NORMAL DISTRIBUTION	IPROB
IUTPT	INVERSE T DISTRIBUTION	IPROB
IVAL	INSERT VECTOR VALUE	VECTR
IVOZ	GENERAL BESSEL $I_v(z)$	CHYPR
IWEIB	INVERSE WEIBULL DISTRIBUTION	IPROB
IXFRM	SYMBOLIC INVERSE LAPLACE AND z TRANSFORM	SYMB FILTR WIND
J0OX	BESSEL FUNCTION $J_0(x)$	BESEL
J1OX	BESSEL FUNCTION $J_1(x)$	BESEL
JNOZ	BESSEL FUNCTION $J_n(z)$	BESEL
JVOZ	GENERAL BESSEL $J_v(z)$	CHYPR
K0OX	BESSEL FUNCTION $K_0(x)$	BESEL

ALPHABETIC COMMAND SUMMARY AND MENU INDEX

MATHLIB COMMAND	DESCRIPTION	DEFINITION MENU (ALL ARE IN MAIN)
K1OX	BESSEL FUNCTION $K_1(x)$	BESEL
KAISR	KAISER WINDOW	WIND
KEEP	KEEPS FIRST N OBJECTS ON STACK	MAIN
KEN&	KELVIN FUNCTION $ker_n(x) + i\ kei_n(x)$	BESEL
KEV&	GENERAL KELVIN $ker_v(z) + i\ kei_v(z)$	CHYPR
KNOX	BESSEL FUNCTION $K_n(x)$	BESEL
KNOZ	BESSEL FUNCTION $K_n(z)$	BESEL
KOKP	COMPLETE ELLIPTIC INTEGRAL K	ELLIPI
KP←M	CONVERT PARAMETER m TO k'	JACOB
KP→M	CONVERT k' TO PARAMETER m	THETA
KP→Q	COMPUTE NOME q(k)	THETA
KST1	KOLMOGOROV-SMIRNOV DISTRIBUTION TEST	STAT
KST2	KOLMOGOROV-SMIRNOV DISTRIBUTION TEST	STAT
KURT	STATISTICAL KURTOSIS	STAT
KVOZ	GENERAL BESSEL $K_v(z)$	CHYPR
K↑KP	CONVERT BETWEEN MODULUS k AND k'	THETA
K¡OK	QUARTER PERIOD RATIO K '/K	ELLIPI
K¡OK¡	COMPLETE ELLIPTIC INTEGRAL K '	ELLIPI
L1F1	LIST ELEMENT OPERATIONS	SYMB
LαOX	LAGUERRE POLYNOMIAL $L_n^{(\alpha)}(x)$	POLY
L→BP	LOWPASS TO BANDPASS AND SCALING	FILTR
L→BS	LOWPASS TO BANDSTOP AND SCALING	FILTR
L→HP	LOWPASS TO HIGHPASS AND SCALING	FILTR
L→LP	LOWPASS FILTER SCALING	FILTR
LAPL	UPPER TAIL LAPLACE DISTRIBUTION	PROB
LCM	LEAST COMMON MULTIPLE	NUMB

ALPHABETIC COMMAND SUMMARY AND MENU INDEX

MATHLIB COMMAND	DESCRIPTION	DEFINITION MENU (ALL ARE IN MAIN)
LCNT	CORRELATION COEFFICIENT TESTS	STAT
LDLTD	LOWER DIAGONAL UPPER DECOMPOSITION	LINAG
LINT	NUMERICAL INTEGRATION FROM LEFT	PROC
LIOX	LOGARITHMIC INTEGRAL Li(x)	EXPIN
LM	COMPUTE TOTAL LOAD	QUE
LNθ1	COMPUTE $Ln[\vartheta_1(\alpha + \beta, k)/\vartheta_1(\alpha - \beta, k)]$	THETA
LNθ2	COMPUTE $Ln[\vartheta_2(\alpha + \beta, k)/\vartheta_2(\alpha - \beta, k)]$	THETA
LNθ3	COMPUTE $Ln[\vartheta_3(\alpha + \beta, k)/\vartheta_3(\alpha - \beta, k)]$	THETA
LNθ4	COMPUTE $Ln[\vartheta_4(\alpha + \beta, k)/\vartheta_4(\alpha - \beta, k)]$	THETA
LOFX	LAGUERRE POLYNOMIAL $L_n(x)$	POLY
LPM	COMPUTE PROCESSING LOAD	QUE
LQM	COMPUTE QUEUEING LOAD	QUE
LREG	LINEAR REGRESSION WITH ANOVA AND SVD	MISC
LREV	LIST ELEMENT REVERSE	VECTR
LROOT	SINGLE ROOT COMPLEX POLYNOMIAL ROOT SOLVER	ALGB
LSERL	LEAST SQUARES ERROR – LEFT SOLVE	LINAG
LSERR	LEAST SQUARES ERROR – RIGHT SOLVE	LINAG
LSOVL	FULLY DETERMINED SOLVE LEFT	LINAG
LSOVR	FULLY DETERMINED SOLVE RIGHT	LINAG
LSSOV	LEAST SQUARES APPROXIMATIONS WITH ANOVA	MISC
LSVDL	SVD LEAST SQUARES SOLVE LEFT	LINAG
LSVDR	SVD LEAST SQUARES SOLVE RIGHT	LINAG
LTPβ	LOWER TAIL BETA DISTRIBUTION	PROB
LTPH	LOWER TAIL HYPERGEOMETRIC DISTRIBUTION	PROB
LTPN	LOWER TAIL NORMAL DISTRIBUTION	PROB
LVOZ	COMPLEX STRUVE FUNCTION $L_v(z)$	STRUV

ALPHABETIC COMMAND SUMMARY AND MENU INDEX

MATHLIB COMMAND	DESCRIPTION	DEFINITION MENU (ALL ARE IN MAIN)
LZDEL	DELETE TRAILING ZEROS IN LIST	ALGB
LZFIL	ZERO-FILL LISTS TO SAME SIZE	VECTR
L←SO	SYMBOLIC MATRIX TO SYMBOLIC VECTOR	SYMB
L→SO	SYMBOLIC VECTOR TO SYMBOLIC MATRIX	SYMB
MγN	UPPER TAIL MARCUM Γ_N DISTRIBUTION	PROB
M1F1	ONE-MATRIX-ELEMENT OPERATIONS	MSAG
M2F1	TWO-MATRIX-ELEMENT OPERATIONS	MSAG
MABS	MATRIX ELEMENT ABS	MSAG
MABZ	CONFLUENT HYPERGEOMETRIC M(a, b, z)	CHYPR
MADD	MATRIX SCALAR ADDITION	MSAG
MALOG	MATRIX ELEMENT ALOG	MSAG
MARG	MATRIX ELEMENT ARG	MSAG
MATD	MATRIX ELEMENT ÷	MSAG
MATX	MATRIX ELEMENT ×	MSAG
MAT↑	SYMBOLIC MATRIX ↑MATCH COMMAND	SYMB
MAT↓	SYMBOLIC MATRIX ↓MATCH COMMAND	SYMB
MAVE	VECTOR MOVING AVERAGE	PROC
MCEIL	MATRIX ELEMENT CEILING	MSAG
MCOL	MOVE MATRIX COLUMN	MATR
MEDIN	MEDIAN OF A DISTRIBUTION	STAT
MENUE	EVALUATE MENU SYSTEM	MAIN
MEXP	MATRIX ELEMENT EXP	MSAG
MFLR	MATRIX ELEMENT FLOOR	MSAG
MINOR	COMPUTE MINOR OF MATRIX	LINAG
MINV	MATRIX INVERSE USING LSOVR	LINAG
MKμZ	WHITTAKER'S FUNCTION $M_{\mu,K}(z)$	CHYPR

ALPHABETIC COMMAND SUMMARY AND MENU INDEX

MATHLIB COMMAND	DESCRIPTION	DEFINITION MENU (ALL ARE IN MAIN)
MLN	MATRIX ELEMENT LN	MSAG
MLOG	MATRIX ELEMENT LOG	MSAG
MMAX	MATRIX MAXIMUM VALUE	MSAG
MMIN	MATRIX MINIMUM VALUE	MSAG
MMOD	MATRIX ELEMENT MOD	MSAG
MODE	MODE OF A DISTRIBUTION	STAT
MPN	UPPER TAIL MARCUM P_N DISTRIBUTION	PROB
MQ	MARCUM Q FUNCTION	MISC
MRDC	RANDOM COMPLEX BIVARIATE NORMAL MATRIX	MSAG
MRDH	RANDOM NORMAL HERMITIAN MATRIX	MSAG
MRDN	RANDOM NORMALLY DISTRIBUTED MATRIX	MSAG
MRDS	RANDOM NORMAL SYMMETRIC MATRIX	MSAG
MRDU	RANDOM UNIFORMLY DISTRIBUTED MATRIX	MSAG
MRND	MATRIX ELEMENT RND	MSAG
MROW	MOVE MATRIX ROW	MATR
MSQ	MATRIX ELEMENT SQ	MSAG
MSRT	MATRIX ELEMENT √	MSAG
MSUB	MATRIX SCALAR SUBTRACTION	MSAG
MULTI	DO UNTIL NO CHANGE	MISC
MVAL	MOVE VECTOR VALUE	VECTR
M←CL	COLUMN LIST TO MATRIX	MATR
M→CL	MATRIX TO COLUMN LIST	MATR
M←K	CONVERT MODULUS k TO m	THETA
M→K	CONVERT m TO MODULUS k	THETA
M↑M_i	CONVERT BETWEEN PARAMETER m AND m_1	THETA
M→RL	MATRIX TO ROW LIST	MATR

ALPHABETIC COMMAND SUMMARY AND MENU INDEX

MATHLIB COMMAND	DESCRIPTION	DEFINITION MENU (ALL ARE IN MAIN)
M←RL	ROW LIST TO MATRIX	MATR
M→SO	MATRIX TO SYMBOLIC MATRIX	SYMB
M←SO	SYMBOLIC MATRIX TO MATRIX	SYMB
NCUK¡	JACOBI ELLIPTIC FUNCTION nc(u, k)	JACOB
NDUK¡	JACOBI ELLIPTIC FUNCTION nd(u, k)	JACOB
NEGB	UPPER TAIL NEGATIVE BINOMIAL DISTRIBUTION	PROB
NSUK¡	JACOBI ELLIPTIC FUNCTION ns(u, k)	JACOB
NUKP	COMPUTE N SET OF JACOBI ELLIPTIC FTNS	JACOB
N→C	SET TRANSFORM FROM N→C SET	JACOB
N←C	SET TRANSFORM FROM N←C SET	JACOB
N←D	SET TRANSFORM FROM N←D SET	JACOB
N→D	SET TRANSFORM FROM N→D SET	JACOB
N→DB	NEPERS TO dB	WIND
N←DB	dB TO NEPERS	WIND
N→S	SET TRANSFORM FROM N→S SET	JACOB
N←S	SET TRANSFORM FROM N←S SET	JACOB
ONE	CREATE A VECTOR WITH ELEMENTS EQUAL TO 1	VECTR
ORTH?	TEST IF MATRIX IS ORTHOGONAL	LINAG
OSOVL	OVERDETERMINED SOLVE LEFT	LINAG
OSOVR	OVERDETERMINED SOLVE RIGHT	LINAG
p	LOWERCASE P GIVES THE VALUE OF π	NUMB
PαβX	JACOBI POLYNOMIAL $P_n^{(\alpha,\beta)}(x)$	POLY
PμVX	LEGENDRE FUNCTION $P_\nu^\mu(x)$ ON CUT	LGDR
PμVZ	LEGENDRE FUNCTION $P_\nu^\mu(z)$	LGDR
PADD	POLYNOMIAL ADDITION	ALGB
PARZ	PARZEN WINDOW	WIND

ALPHABETIC COMMAND SUMMARY AND MENU INDEX		
MATHLIB COMMAND	DESCRIPTION	DEFINITION MENU (ALL ARE IN MAIN)
PDERV	POLYNOMIAL DERIVATIVE	ALGB
PDQW	PROBABILITY OF WAITING MORE THAN T	QUE
PDVD	POLYNOMIAL DIVISION	ALGB
PEQN	PRETTY POLYNOMIAL EQUATION IN Tn	MISC
PERMF	EVALUATE COMPLEX PERMUTATIONS	MISC
PHASE	COMPUTE PHASE IN DEGREES	PROC
PHASU	UNWRAP PHASE VECTOR IN DEGREES	PROC
PHOUS	COMPUTE HOUSEHOLDER MATRIX	LINAG
PINCG	PEARSON INCOMPLETE GAMMA I(u, p)	GAMA
PINDX	CREATE PHASE INDEX VECTOR	PROC
PINTG	POLYNOMIAL INTEGRATION	ALGB
PLDVD	POLYNOMIAL LONG DIVIDE	ALGB
PLT1	PLOT ONE VECTOR WITHOUT LABELS	PLOT
PLT1L	PLOT ONE VECTOR WITH LABELS	PLOT
PLT2	PLOT TWO VECTORS WITHOUT LABELS	PLOT
PLT2L	PLOT TWO VECTOR WITH LABELS	PLOT
PLT3	OVERLAYS ADDITIONAL PLOT ON PICT	PLOT
PLTC	COMPLEX PLOT VECTOR WITHOUT LABELS	PLOT
PLTCL	COMPLEX PLOT VECTOR WITH LABELS	PLOT
PLXLY	BIVARIATE NORMAL DISTRIBUTION	BIVN
PLXUY	BIVARIATE NORMAL DISTRIBUTION	BIVN
PMAT	POLYNOMIAL APPROXIMATION MATRIX	MISC
PMPY	POLYNOMIAL MULTIPLICATION	ALGB
PNSM	PROBABILITY OF N IN SYSTEM	QUE
POCH	EVALUATE POCHHAMMER'S SYMBOL	MISC
POFX	LEGENDRE POLYNOMIAL $P_n(x)$	POLY

ALPHABETIC COMMAND SUMMARY AND MENU INDEX

MATHLIB COMMAND	DESCRIPTION	DEFINITION MENU (ALL ARE IN MAIN)
POLEP	PLOT ZEROS AND POLES OF RESPONSE	FILTR
POSN	UPPER TAIL POISSON DISTRIBUTION	PROB
PQB	PROBABILITY OF BUSY	QUE
PQUP	UPWARD RECURRENCE FOR P AND Q	LGDR
PRIME	COMPUTE PRIME NUMBERS IN RANGE	NUMB
PROOT	MATRIX COMPLEX POLYNOMIAL ROOT SOLVER	ALGB
PSCAL	POLYNOMIAL LIST SCALE a_0 TO 1	FILTR
PSI	DIGAMMA FUNCTION $\psi(z)$	GAMA
PSUB	POLYNOMIAL SUBTRACTION	ALGB
PUXLY	BIVARIATE NORMAL DISTRIBUTION	BIVN
PUXUY	BIVARIATE NORMAL DISTRIBUTION	BIVN
PZINV	z TO z^{-1} SUBSTITUTION – POLYNOMIAL LISTS	FILTR
QμVX	LEGENDRE FUNCTION $Q_v^{\mu}(x)$ ON CUT	LGDR
QμVZ	LEGENDRE FUNCTION $P_v^{\mu}(z)$	LGDR
R→CL	ROOT VECTORS TO POLYNOMIAL LISTS	FILTR
RΣ0D	ROW SUM WITH ROW DELETE IF ZERO	MSAG
RABS	EUCLIDEAN NORM OF EACH ROW	LINAG
RABS2	EUCLIDEAN ROW NORM SQUARED	MSAG
RATAP	RATIONAL APPROXIMATION OF FUNCTIONS	MISC
RAYL	UPPER TAIL RAYLEIGH DISTRIBUTION	PROB
RCMB	COMBINE TWO MATRICES BY ROWS	MATR
RCOL	REPLACE ENTIRE MATRIX COLUMN	MATR
RCOLS	MATRIX REPLACE COLUMN SUBSET	MATR
RDERV	POLYNOMIAL DERIVATIVE OF QUOTIENT	ALGB
RDLV	DE–INTERLEAVE MATRIX ROWS	MATR
REDN	TRUNCATE VECTOR TO LENGTH N	PROC

ALPHABETIC COMMAND SUMMARY AND MENU INDEX

MATHLIB COMMAND	DESCRIPTION	DEFINITION MENU (ALL ARE IN MAIN)
REFLT	VECTOR COMPLEX REFLECT	PROC
RESDA	ARBITRARY RESIDUE EVALUATION	SYMB
RESDP	POLYNOMIAL RESIDUE EVALUATION	SYMB
REVAL	EVALUATE RATIONAL APPROXIMATION	MISC
REV→	LEFT–RIGHT MATRIX REVERSE	MATR
REV↑	UP–DOWN MATRIX REVERSE	MATR
RGET	EXTRACT MATRIX ROW SUBSET	MATR
RHOUS	APPLY HOUSEHOLDER VECTOR TO MATRIX	LINAG
RICE	UPPER TAIL RICIAN DISTRIBUTION	PROB
RINT	NUMERICAL INTEGRATION FROM RIGHT	PROC
RNDC	RANDOM COMPLEX BIVARIATE NORMAL VECTOR	PROC
RNDN	RANDOM NORMALLY DISTRIBUTED VECTOR	PROC
RNDU	RANDOM UNIFORMLY DISTRIBUTED VECTOR	PROC
RNLV	INTERLEAVE MATRIX ROWS	MATR
RNUM	COUNT NUMBER OF ENTRIES IN EACH ROW	STAT
RORDR	REORDER MATRIX ROWS USING PIVOT VECTOR	LINAG
ROWμ	MEAN VALUE OF EACH MATRIX ROW	STAT
ROWσ	STANDARD DEVIATION OF EACH MATRIX ROW	STAT
RPSM	REPLACE PARTIAL SUBMATRIX	MATR
RROT	ROW GIVENS ROTATE	LINAG
RROW	REPLACE ENTIRE MATRIX ROW	MATR
RRWS	MATRIX REPLACE ROW SUBSET	MATR
RSBM	REPLACE SUBMATRIX	MATR
RSORT	MATRIX ROW SORT ↑	MATR
RSPLT	SPLIT MATRIX BY ROWS	MATR
RSRTI	MATRIX ROW SORT ↓	MATR

ALPHABETIC COMMAND SUMMARY AND MENU INDEX

MATHLIB COMMAND	DESCRIPTION	DEFINITION MENU (ALL ARE IN MAIN)
RSUM	COMPUTE VECTOR OF ROW SUMS	MSAG
RZETA	RIEMANN ZETA FUNCTION	NUMB
RZINV	z TO z^{-1} SUBSTITUTION – ROOT VECTORS	FILTR
S→ZP	STATE SPACE TO ZEROS AND POLES	FILTR
S1F1	SYMBOLIC MATRIX ELEMENT OPERATIONS	SYMB
S1OZ	FRESNEL INTEGRAL $S_1(z)$	ERROR
S2OZ	FRESNEL INTEGRAL $S_2(z)$	ERROR
SABS	SYMBOLIC MATRIX ABS COMMAND	SYMB
SADD	SYMBOLIC MATRIX ADDITION	SYMB
SADDS	SYMBOLIC MATRIX SCALAR ADD	SYMB
SCHRD	GENERAL SCHUR DECOMPOSITION	LINAG
SCHS	SYMBOLIC MATRIX CHANGE SIGN	SYMB
SCINT	SINC SQUARED INTEGRAL	EXPIN
SCNJ	SYMBOLIC VECTOR AND MATRIX CONJ	SYMB
SCOL	SWAP MATRIX COLUMNS	MATR
SCOLC	SYMBOLIC MATRIX COLLECT	SYMB
SCROS	SYMBOLIC VECTOR CROSS PRODUCT	SYMB
SCRSD	SYMMETRIC & HERMITIAN SCHUR DECOMPOSITION	LINAG
$SCUK_j$	JACOBI ELLIPTIC FUNCTION sc(u, k)	JACOB
SCURL	SYMBOLIC CURL	SYMB
SDERV	SYMBOLIC MATRIX DERIVATIVE	SYMB
SDET	SYMBOLIC MATRIX DETERMINANT	SYMB
SDIV	SYMBOLIC DIVERGENCE	SYMB
SDOT	SYMBOLIC MATRIX DOT PRODUCT	SYMB
$SDUK_j$	JACOBI ELLIPTIC FUNCTION sd(u, k)	JACOB
SEC	SECANT	TRIG

ALPHABETIC COMMAND SUMMARY AND MENU INDEX

MATHLIB COMMAND	DESCRIPTION	DEFINITION MENU (ALL ARE IN MAIN)
SECH	HYPERBOLIC SECANT	HYP
SECOL	SYMBOLIC MATRIX EXTRACT COLUMN	SYMB
SERW	SYMBOLIC MATRIX EXTRACT ROW	SYMB
SEVAL	SYMBOLIC MATRIX EVAL COMMAND	SYMB
SEXCO	SYMBOLIC MATRIX EXCO COMMAND	SYMB
SEXPD	SYMBOLIC MATRIX EXPAND	SYMB
SGET	SYMBOLIC MATRIX GET COMMAND	SYMB
SGRD	SYMBOLIC GRADIENT	SYMB
SH1Z	SPHERICAL HANKEL FUNCTION $h_n^{(1)}(z)$	SBESL
SH2Z	SPHERICAL HANKEL FUNCTION $h_n^{(2)}(z)$	SBESL
SHIZ	EXPONENTIAL INTEGRAL Shi(z)	EXPIN
SI1NZ	SPHERICAL BESSEL FUNCTION $i_n^{(1)}(z)$	SBESL
SI2NZ	SPHERICAL BESSEL FUNCTION $i_n^{(2)}(z)$	SBESL
SIGNP	SIGN FUNCTION WITH SIGNP(0) = 1	MISC
SIM	SYMBOLIC MATRIX IMAGINARY PART	SYMB
SINC	SINC FUNCTION = sin z / z	EXPIN
SINC2	SQUARE OF SINC FUNCTION	EXPIN
SINT	SINE INTEGRAL Si(z)	EXPIN
SINTG	SYMBOLIC MATRIX INTEGRATION	SYMB
SJNZ	SPHERICAL BESSEL FUNCTION $j_n(z)$	SBESL
SKEW	STATISTICAL SKEWNESS	STAT
SKNZ	SPHERICAL BESSEL FUNCTION $k_n(z)$	SBESL
SMI	SYMBOLIC MATRIX INVERSION UTILITY	SYMB
SMNR	SYMBOLIC MATRIX MINOR	SYMB
SMPY	SYMBOLIC MATRIX MULTIPLICATION	SYMB
SNUK¡	JACOBI ELLIPTIC FUNCTION sn(u, k)	JACOB

ALPHABETIC COMMAND SUMMARY AND MENU INDEX

MATHLIB COMMAND	DESCRIPTION	DEFINITION MENU (ALL ARE IN MAIN)
SNUM	SYMBOLIC MATRIX →NUM COMMAND	SYMB
SOB→	SYMBOLIC MATRIX TO STACK	SYMB
SOFX	CHEBYSHEV POLYNOMIAL $S_n(x)$	POLY
SOFZ	FRESNEL INTEGRAL S(z)	ERROR
SPECT	COMPUTE POWER SPECTRUM	PROC
SPUT	SYMBOLIC MATRIX PUT COMMAND	SYMB
SQWV	GENERATE SQUARE WAVE	PROC
SRCOL	SYMBOLIC MATRIX REPLACE COLUMN	SYMB
SRCTT	SPEARMAN RANK CORRELATION TEST	STAT
SRE	SYMBOLIC MATRIX REAL PART	SYMB
SRND	SYMBOLIC MATRIX AND VECTOR ROUND	SYMB
SROW	SWAP MATRIX ROWS	MATR
SRRW	SYMBOLIC MATRIX REPLACE ROW	SYMB
SRTI	SORT ↑ WITH SORTED INDEX	VECTR
SRT↑	SORT VECTOR ↑	VECTR
SRT↓	SORT VECTOR ↓	VECTR
SSIZE	SYMBOLIC VECTOR AND MATRIX SIZE	SYMB
SSORT	SORT ACCORDING TO RE(s) > = < 0	MISC
SSUB	SYMBOLIC MATRIX SUBTRACTION	SYMB
SSUBS	SYMBOLIC MATRIX SCALAR SUBTRACT	SYMB
STRAC	SYMBOLIC MATRIX TRACE	SYMB
STRL1	STIRLING NUMBER FIRST KIND	NUMB
STRL2	STIRLING NUMBER SECOND KIND	NUMB
STRLT	STIRLING COEFFICIENT TRANSFORM	WIND
STRN	SYMBOLIC MATRIX TRANSPOSE	SYMB
SUKP	COMPUTE S SET OF JACOBI ELLIPTIC FTNS	JACOB

ALPHABETIC COMMAND SUMMARY AND MENU INDEX

MATHLIB COMMAND	DESCRIPTION	DEFINITION MENU (ALL ARE IN MAIN)
SUMΣ	SUM SYMBOLIC SERIES	ALGB
SVAL	SWAP VECTOR VALUES	VECTR
SVBAR	SYMBOLIC MATRIX | (WHERE) COMMAND	SYMB
SVD	SINGULAR VALUE MATRIX DECOMPOSITION	LINAG
SVDDI	INVERT SVD DIAGONAL VECTOR	LINAG
SVDMI	INVERT SVD MATRIX	LINAG
SYM?	TEST IF MATRIX IS SYMMETRIC	LINAG
SYNZ	SPHERICAL BESSEL FUNCTION $y_n(z)$	SBESL
T1UQ	LOGARITHMIC DERIVATIVE $\vartheta_1(u\|q)$	THETA
T2UQ	LOGARITHMIC DERIVATIVE $\vartheta_2(u\|q)$	THETA
T3UQ	LOGARITHMIC DERIVATIVE $\vartheta_3(u\|q)$	THETA
T4UQ	LOGARITHMIC DERIVATIVE $\vartheta_4(u\|q)$	THETA
TALR1	ONE DIMENSIONAL TAYLOR SERIES	MISC
TALR2	TWO DIMENSIONAL TAYLOR SERIES	MISC
TCUKj	LOGARITHMIC DERIVATIVE $\vartheta_c(u, k)$	THETA
TDUKj	LOGARITHMIC DERIVATIVE $\vartheta_d(u, k)$	THETA
TF→C	TRANSFER FUNCTION TO CONTROLLABLE FORM	FILTR
TI	TEST IF INTEGER	PCLDR
TIFRE	TEST IF REAL AND MAKE REAL	SYMB
TIMIT	COMPUTE EXECUTION TIME	MISC
TNI	TEST IF NEGATIVE INTEGER OR ZERO	PCLDR
TNP	TEST IF NOT POSITIVE REAL NUMBER	PCLDR
TNUKj	LOGARITHMIC DERIVATIVE $\vartheta_n(u, k)$	THETA
TOFX	CHEBYSHEV POLYNOMIAL $T_n(x)$	POLY
TOFXL	CHEBYSHEV POLYNOMIAL $T_n(x)$ LIST	WIND
TRACE	COMPUTE TRACE OF MATRIX	LINAG

ALPHABETIC COMMAND SUMMARY AND MENU INDEX

MATHLIB COMMAND	DESCRIPTION	DEFINITION MENU (ALL ARE IN MAIN)
TRNH	HERMITIAN MATRIX TRANSPOSE	LINAG
TRNP	NON-HERMITIAN MATRIX TRANSPOSE	LINAG
TSFX	CHEBYSHEV SHIFTED POLYNOMIAL $T_n(x)$	POLY
TSTAT	LINEAR REGRESSION T STATISTICS	MISC
TSUK$_j$	LOGARITHMIC DERIVATIVE $\vartheta_4(u, k)$	THETA
TTDV	T TEST FOR MEANS DIFFERENT VARIANCE	STAT
TTPS	T TEST FOR MEANS PAIRED SAMPLES	STAT
TTSV	T TEST FOR MEANS SAME VARIANCE	STAT
TWIDL	COMPUTE FFT TWIDDLE VECTOR	PROC
UτηC	UPPER TAIL NON-CENTRAL χ^2 DISTRIBUTION	PROB
UτηF	UPPER TAIL NON-CENTRAL F DISTRIBUTION	PROB
UτηT	UPPER TAIL NON-CENTRAL T DISTRIBUTION	PROB
UτPC	UPPER TAIL χ^2 DISTRIBUTION	PROB
UτPF	UPPER TAIL F DISTRIBUTION	PROB
UτPN	UPPER TAIL NORMAL DISTRIBUTION	PROB
UτPT	UPPER TAIL F DISTRIBUTION	PROB
UABZ	CONFLUENT HYPERGEOMETRIC U(a, b, z)	CHYPR
UHESD	UPPER HESSENBERG DECOMPOSITION	LINAG
UNIF	UPPER TAIL UNIFORM DISTRIBUTION	PROB
UNIQE	IDENTIFY AND COUNT UNIQUE ELEMENTS	MISC
UNIT?	TEST IF MATRIX IS UNITARY	LINAG
UNITI	CREATE UNIT IMPULSE VECTOR: [1 0 ... 0]	VECTR
UOAX	PARABOLIC CYLINDER U(a, x)	PCLDR
UOFX	CHEBYSHEV POLYNOMIAL $U_n(x)$	POLY
USFX	CHEBYSHEV SHIFTED POLYNOMIAL $T_n(x)$	POLY
USOVL	UNDERDETERMINED SOLVE LEFT	LINAG

ALPHABETIC COMMAND SUMMARY AND MENU INDEX

MATHLIB COMMAND	DESCRIPTION	DEFINITION MENU (ALL ARE IN MAIN)
USOVR	UNDERDETERMINED SOLVE RIGHT	LINAG
UTKS	UPPER TAIL KOLMOGOROV–SMIRNOV DISTRIBUTION	PROB
UTPβ	UPPER TAIL BETA DISTRIBUTION	PROB
UTPE	UPPER TAIL EXPONENTIAL DISTRIBUTION	PROB
UTPG	UPPER TAIL GAMMA DISTRIBUTION	PROB
UZKP	COMPLEX INCOMPLETE ELLIPTIC F(x, k)	THETA
V1F1	ONE–VECTOR–ELEMENT OPERATIONS	VSAG
V2F1	TWO–VECTOR–ELEMENT OPERATIONS	VSAG
VABS	VECTOR ELEMENT ABS	VSAG
VADD	VECTOR SCALAR ADDITION	VSAG
VALOG	VECTOR ELEMENT ALOG	VSAG
VARG	VECTOR ELEMENT ARG	VSAG
VARM	STATISTICAL MEAN AND VARIANCE	STAT
VCDEL	VECTOR CONSTANT ELEMENT DELETE	VSAG
VCEIL	VECTOR ELEMENT CEILING	VSAG
VCMB	COMBINE TWO VECTORS INTO ONE VECTOR	STAT
VCORR	VECTOR CORRELATION WITHOUT FFT	PROC
VDEC	VECTOR DECIMATE	PROC
VDEL	DELETE A PORTION OF A VECTOR	VECTR
VDVD	VECTOR DECONVOLUTION WITHOUT FFT	PROC
VECTD	VECTOR ELEMENT ÷	VSAG
VECTX	VECTOR ELEMENT ×	VSAG
VEXP	VECTOR ELEMENT EXP	VSAG
VFLOR	VECTOR ELEMENT FLOOR	VSAG
VHOUS	COMPUTE HOUSEHOLDER VECTOR	LINAG
VINST	INSERT VECTOR INTO VECTOR	VECTR

ALPHABETIC COMMAND SUMMARY AND MENU INDEX

MATHLIB COMMAND	DESCRIPTION	DEFINITION MENU (ALL ARE IN MAIN)
VINTP	VECTOR INTERPOLATE	PROC
VLN	VECTOR ELEMENT LN	VSAG
VLOG	VECTOR ELEMENT LOG	VSAG
VMAX	VECTOR MAXIMUM VALUE	VSAG
VMIN	VECTOR MINIMUM VALUE	VSAG
VMOD	VECTOR ELEMENT MOD	VSAG
VMPY	VECTOR CONVOLVE WITHOUT FFT	PROC
VOAX	PARABOLIC CYLINDER V(a, x)	PCLDR
VREPL	REPLACE VECTOR SUBSET	VECTR
VREV	VECTOR ELEMENT REVERSE	VECTR
VRND	VECTOR ELEMENT RND	VSAG
VROT	VECTOR ROTATE BY ± N	PROC
VSINV	VECTOR ELEMENT INV	VSAG
VSPLT	SPLIT VECTOR INTO TWO VECTORS	STAT
VSQ	VECTOR ELEMENT SQ	VSAG
VSRT	VECTOR ELEMENT √	VSAG
VSUB	VECTOR SCALAR SUBTRACTION	VSAG
VSUBS	EXTRACT VECTOR SUBSET	VECTR
VTRUD	PHASE UNWRAP COMPLEX VECTOR IN DEGREES	PROC
VTRUR	ARGUMENT UNWRAP COMPLEX VECTOR IN RADIANS	VSAG
VΣ	SUM OF VALUES IN DATA VECTOR	PROC
V→DB	VOLTS TO dB	WIND
V←DB	dB TO VOLTS	WIND
V→L	CONVERT VECTOR TO LIST	VECTR
V←L	CONVERT LIST TO VECTOR	VECTR
WARP	PREWARP FREQUENCY FOR z TRANSFORM	FILTR

ALPHABETIC COMMAND SUMMARY AND MENU INDEX

MATHLIB COMMAND	DESCRIPTION	DEFINITION MENU (ALL ARE IN MAIN)
WEIB	UPPER TAIL WEIBULL DISTRIBUTION	PROB
WELC	WELCH WINDOW	WIND
WKμZ	WHITTAKER'S FUNCTION $W_{\mu,\kappa}(z)$	CHYPR
WL1	WIENER–LEVINSON SOLUTION TYPE 1	PROC
WL2	WIENER–LEVINSON SOLUTION TYPE 2	PROC
WM	COMPUTE TIME IN SYSTEM	QUE
WOAX	PARABOLIC CYLINDER W(a, x)	PCLDR
WQM	COMPUTE TIME IN QUEUE	QUE
WSQR	SYMMETRIC EIGEN DECOMPOSITION STEP	LINAG
XEQN	PRETTY POLYNOMIAL EQUATION IN z^n	MISC
Y0OX	BESSEL FUNCTION $Y_0(x)$	BESEL
Y1OX	BESSEL FUNCTION $Y_1(x)$	BESEL
YNOX	BESSEL FUNCTION $Y_n(x)$	BESEL
YNOZ	BESSEL FUNCTION $Y_n(z)$	BESEL
YVOZ	GENERAL BESSEL $Y_v(z)$	CHYPR
ZERO	CREATE A VECTOR WITH ELEMENTS EQUAL TO 0	VSAG
ZFIL1	ZERO–FILLS TO TWICE THE LENGTH – 2	PROC
ZFILL	ZERO–FILLS TO TWICE THE LENGTH	PROC
ZFILN	ZERO FILL VECTOR TO LENGTH N	PROC
ZOFZ	DERIVATIVE OF ERROR FUNCTION Z(z)	ERROR
ZSORT	SORT ACCORDING TO ABS(z) > = < 1	MISC
ZUKP	JACOBI'S ZETA FUNCTION Z(u, k)	ELLIPI
→COL	VECTOR TO COLUMN VECTOR	MATR
→ROW	VECTOR TO ROW VECTOR	MATR
→SOB	STACK TO SYMBOLIC MATRIX	SYMB
→VTR	ROW OR COLUMN VECTOR TO VECTOR	VECTR

ALPHABETIC COMMAND SUMMARY AND MENU INDEX

MATHLIB COMMAND	DESCRIPTION	DEFINITION MENU (ALL ARE IN MAIN)
αNOZ	EXPONENTIAL INTEGRAL $\alpha_n(z)$	EXPIN
α→Kj	CONVERT MODULAR ANGLE α TO k'	THETA
α←Kj	CONVERT k' TO MODULAR ANGLE α	THETA
βNOZ	EXPONENTIAL INTEGRAL $\beta_n(z)$	EXPIN
γ	EULER'S CONSTANT	EXPIN
δη→A	CONVERSION FROM δ_1 δ_2 TO a_{max} a_{min}	WIND
δη←A	CONVERSION FROM a_{max} a_{min} TO δ_1 δ_2	WIND
ΔWAV	GENERATE TRIANGULAR WAVE	PROC
ΔZKP	ELLIPTIC DELTA FUNCTION Δ(z, k)	ELLIPI
εα→U	CONVERSION FOR AMS 55 TABLES	THETA
εα←U	CONVERSION FOR AMS 55 TABLES	THETA
θ1UKj	JACOBI THETA FUNCTION $\Theta_1(u, k)$	THETA
θ1ZQ	THETA FUNCTION $\vartheta_1(z\|q)$	THETA
θ2ZQ	THETA FUNCTION $\vartheta_2(z\|q)$	THETA
θ3ZQ	THETA FUNCTION $\vartheta_3(z\|q)$	THETA
θ4ZQ	THETA FUNCTION $\vartheta_4(z\|q)$	THETA
θCUKj	NEVILLE'S THETA FUNCTION $\vartheta_c(u, k)$	THETA
θDUKj	NEVILLE'S THETA FUNCTION $\vartheta_d(u, k)$	THETA
θFCFS	COMPUTE ELLIPTIC MODULAR ANGLE	FILTR
θNUKj	NEVILLE'S THETA FUNCTION $\vartheta_n(u, k)$	THETA
θSUKj	NEVILLE'S THETA FUNCTION $\vartheta_s(u, k)$	THETA
θUKP	JACOBI THETA FUNCTION Θ(u, k)	THETA
λ?	TEST IF EIGENVALUE	MATR
λV?	TEST IF EIGENVECTOR	MATR
λXKP	HEUMAN'S LAMBDA FUNCTION $\Lambda_0(x, k)$	ELLIPI
μ	STATISTICAL MEAN	STAT

ALPHABETIC COMMAND SUMMARY AND MENU INDEX		
MATHLIB COMMAND	DESCRIPTION	DEFINITION MENU (ALL ARE IN MAIN)
ΠNXKP	INCOMPLETE ELLIPTIC INTEGRAL Π(n, x, k)	ELLIPI
ρ←AX	a_{MIN} TO REFLECTION COEFFICIENT	FILTR
ρ→AX	REFLECTION COEFFICIENT TO a_{MIN}	FILTR
σ	STATISTICAL STANDARD DEVIATION	STAT
τOFω	EVALUATE GROUP DELAY AT A FREQUENCY	FILTR
τVT1	ANALOG AND DIGITAL GROUP DELAY	FILTR
τVT2	ANALOG AND DIGITAL GROUP DELAY	FILTR
ϕ→X	CONVERT AMPLITUDE φ TO x	THETA
ϕ←X	CONVERT x TO AMPLITUDE φ	THETA
ΩMAX	LOCATION OF PASSBAND MAX ATTENUATION	FILTR
ΩMIN	LOCATION OF STOPBAND MIN ATTENUATION	FILTR
¡	GIVES THE VALUE OF i = (0, 1)	NUMB
¡¡	DOUBLE FACTORIAL FUNCTION (n)!!	GAMA

SYMBOL INDEX FOR MATHEMATICAL FUNCTIONS

SYMBOL	FUNCTION	MENU	COMMAND
ABS	Absolute value	HP	ABS
Ai(z)	Airy function	SBESL	AIOZ
am(u, k)	Amplitude φ = arcsin(sn(u, k))	JACOB	AMUK¡
arccos z	Inverse of cos z	HP	ACOS
arccosh z	Inverse of cosh z	HYP	ACOSH
arccot z	Inverse of cot z	TRIG	ACOT
arccoth z	Inverse of coth z	HYP	ACOTH
arccsc z	Inverse of csc z	TRIG	ACSC
arccsch z	Inverse of csch z	HYP	ACSCH
arcsec z	Inverse of sec z	TRIG	ASEC
arcsech z	Inverse of sech z	HYP	ASECH
arcsin z	Inverse of sin z	HP	ASIN
arcsinh z	Inverse of sinh z	HYP	ASINH
arctan z	Inverse of tan z	HP	ATAN
arctanh z	Inverse of tanh z	HYP	ATANH
arg z	Argument of z	HP	ARG
B(x, k)	Elliptic integral	ELLIPI	BXKP
B(z, w)	Beta function	GAMA	BETA
bei_v x	Kelvin function	BESEL	BEN&
ber_v x	Kelvin function	BESEL	BEN&
Bi(z)	Airy function	SBESL	BIOZ
B_n	Bernoulli number	NUMB	BR
$B_n(x)$	Bernoulli polynomial	NUMB	BPOLY
$B_x(a, b)$	Incomplete beta function	GAMA	INCβ
C(z)	Fresnel integral	ERROR	COFZ
$C_1(x)$	Fresnel integral	ERROR	C1OZ
$C_2(x)$	Fresnel integral	ERROR	C2OZ
cd(u, k)	Jacobi elliptic function	JACOB	CDUK¡

SYMBOL INDEX FOR MATHEMATICAL FUNCTIONS

SYMBOL	FUNCTION	MENU	COMMAND
CEIL	Smallest integer greater or equal to argument	HP	CEIL
Chi(z)	Hyperbolic cosine integral	EXPIN	CHIZ
Ci(z)	Cosine integral	EXPIN	CINT
$C_n^{(\alpha)}(x)$	Gegenbauer ultraspherical polynomial	POLY	CαOX
cn(u, k)	Jacobi elliptic function	JACOB	CNUK¡
$C_n(x)$	Chebyshev polynomial of the second kind	POLY	COFX
COMB	Binomial coefficient: COMB(a, b) = a!/[b!(a – b)!]	HP	COMB
CONJ	Complex conjugate often denoted by z^* = CONJ(z)	HP	CONJ
cos z	Cosine function	HP	COS
cosh z	Hyperbolic cosine function	HP	COSH
cs(u,k)	Jacobi elliptic function	JACOB	CSUK¡
cot z	Cotangent function	TRIG	COT
coth z	Hyperbolic cotangent function	HYP	COTH
csc z	Cosecant function	TRIG	CSC
csch z	Hyperbolic cosecant function	HYP	CSCH
D(x, k)	Elliptic integral	ELLIPI	DXKP
dc(u, k)	Jacobi elliptic function	JACOB	DCUK¡
DET	Determinant of a matrix	HP	DET
dn(u, k)	Jacobi elliptic function	JACOB	DNUK¡
ds(u, k)	Jacobi elliptic function	JACOB	DSUK¡
$D_v(z)$	Parabolic cylinder function	PCLDR	DVOZ
e	Euler's constant e = 2.71828182846	HP	e
E	Complete elliptic integral of the second kind	ELLIPI	EOKP
E(x, k)	Incomplete elliptic integral of the second kind	ELLIPI	EXKP
$E_1(z)$	Exponential integral	EXPIN	E1OZ
Ei(x)	Exponential integral	EXPIN	EIOX
E_n	Euler number	NUMB	ER
$E_n(x)$	Euler polynomial	NUMB	EPOLY

SYMBOL INDEX FOR MATHEMATICAL FUNCTIONS

SYMBOL	FUNCTION	MENU	COMMAND
$E_n(z)$	Exponential integral	EXPIN	ENOZ
erf z	Error function	ERROR	ERFZ
erfc z	Complementary error function	ERROR	ERFCZ
$E_v^{(0)}(z)$	Weber's parabolic cylinder function	PCLDR	EV0Z
$E_v^{(1)}(z)$	Weber's parabolic cylinder function	PCLDR	EV1Z
exp z = e^z	Exponential function	HP	EXP
F(a, b, c, z)	Gaussian hypergeometric function $_2F_1(a; b; c; z)$	GHYP	F2F1
$_pF_q(\alpha_1,...;\beta_1...;z)$	Generalized hypergeometric function	CHYPR	NONE
$f_n(x)$	Discrete orthogonal Chebyshev polynomial	WIND	CHEBY
F_n	nth Fibonacci number	NUMB	FIBON
$F_k(\eta)$	Fermi–Dirac function	THETA	PROGRAM
F(x, k)	Incomplete elliptic integral of the first kind	ELLIPI	FXKP
$F_L(\eta, \rho)$	Regular Coulomb wave function FL$\eta\rho$	CHYPR	PROGRAM
FLOOR	Greatest integer less than or equal to argument	HP	FLOOR
FP	Fractional part of real number	HP	FP
g_2 and g_3	Invariants associated with Weierstrass elliptic functions	ELLIPI	PROGRAM
$G_L(\eta, \rho)$	Irregular Coulomb wave function GL$\eta\rho$	CHYPR	PROGRAM
$G_n(p, q, x)$	Jacobi polynomial	POLY	GPQX
$He_n(x)$	Hermite polynomial	POLY	HEOX
$h_n^{(1)}(z)$	Spherical Bessel function of the third kind	SBESL	SH1Z
$h_n^{(2)}(z)$	Spherical Bessel function of the third kind	SBESL	SH2Z
$H_n(x)$	Hermite polynomial	POLY	HOX
$\mathbf{H}_v(z)$	Struve function	STRUV	HVOZ
I or |	Identity matrix	HP	IDN
i	$(0, 1) = e^{i\pi/2}$	HP	i
I(u, p)	Pearson incomplete gamma function	GAMA	PINCG
IM z	Imaginary part of z	HP	IM
i^n erfc z	Repeated integral of the error function	ERROR	INERFC

SYMBOL INDEX FOR MATHEMATICAL FUNCTIONS

SYMBOL	FUNCTION	MENU	COMMAND
$i_n^{(1)}(z)$	Modified spherical Bessel function of the first kind	SBESL	SI1NZ
$i_n^{(2)}(z)$	Modified spherical Bessel function of the second kind	SBESL	SI2NZ
IP	Integer part of a real number	HP	IP
$I_\nu(z)$	Modified Bessel function of the first kind	BESEL	INOZ
$I_x(a, b)$	Incomplete beta function	GAMA	INCB
$j_n(z)$	Spherical Bessel function of the first kind	SBESL	SJNZ
$J_\nu(z)$	Bessel function of the first kind	BESEL	JNOZ
k	Modulus of Jacobian elliptic functions	THETA	SEVERAL
K	Complete elliptic integral of first kind	ELLIPI	KOKP
K '	Complementary complete elliptic integral of first kind	ELLIPI	K¡OK¡
$kei_\nu x$	Kelvin function	BESEL	KEN&
$ker_\nu x$	Kelvin function	BESEL	KEN&
$k_n(z)$	Modified spherical Bessel function of the third kind	SBESL	SKNZ
$K_\nu(z)$	Modified Bessel function of second kind	BESEL	KNOZ
k'	Complementary modulus of Jacobian elliptic functions	THETA	SEVERAL
L(h, k, ρ)	Cumulative bivariate normal probability	BIVN	SEVERAL
Li(x)	Logarithmic integral	EXPIN	LIOX
lim or Lim	Limit	HP	EVAL
ln or Ln	Natural base e logarithm	HP	LN
$L_n^{(\alpha)}(x)$	Generalized Laguerre polynomial	POLY	LαOX
$L_n(x)$	Laguerre polynomial	POLY	LOFX
$\mathfrak{L}_n(z)$	Polylogarithm (Jonquiere) function	THETA	PROGRAM
log or Log	Base 10 logarithm	HP	LOG
$\mathbf{L}_\nu(z)$	Modified Struve function	STRUV	LVOZ
m	Parameter in elliptic functions	THETA	SEVERAL
m_1	Complementary parameter in elliptic functions	THETA	SEVERAL
MAX	Maximum	HP	MAX
MIN	Minimum	HP	MIN

SYMBOL INDEX FOR MATHEMATICAL FUNCTIONS

SYMBOL	FUNCTION	MENU	COMMAND
M(a, b, z)	Kummer's confluent hypergeometric function ${}_1F_1(a; b; z)$	CHYPR	MABZ
$M_{\kappa,\mu}(z)$	Whittaker function	CHYPR	MKμZ
MOD	x MOD y = x – y FLOOR(x/y)	HP	MOD
n	Characteristic of elliptical integral of the third kind	ELLIPI	ΠNXKP
nc(u, k)	Jacobi elliptic function	JACOB	NCUKj
nd(u, k)	Jacobi elliptic function	JACOB	NDUKj
ns(u, k)	Jacobi elliptic function	JACOB	NSUKj
PERM	Permutation: PERM(a, b) = a!/(a – b)!	HP	PERM
$P_n^{(\alpha,\beta)}(x)$	Jacobi polynomial	POLY	PαβX
$P_n(x)$	Legendre spherical polynomial	POLY	POFX
$p_\nu^\mu(x)$	Associated Legendre function on the cut	LGDR	PμVX
$P_\nu^\mu(z)$	Associated Legendre function of the first kind	LGDR	PμVZ
$\wp(z; g_2, g_3)$	Weierstrass elliptic function – also $\wp(z\|\omega, \omega')$	ELLIPI	PROGRAM
q	Nome in theta functions	THETA	KP→Q
q_1	Complementary nome in theta functions	THETA	KP→Q
$q_\nu^\mu(z)$	Associated Legendre function on the cut	LGDR	QμVX
$Q_\nu^\mu(z)$	Associated Legendre function of the second kind	LGDR	QμVZ
RE z	Real part of z	HP	RE
S(z)	Fresnel integral	ERROR	SOFZ
$S_1(z)$	Fresnel integral	ERROR	S1OZ
$S_2(z)$	Fresnel integral	ERROR	S2OZ
sc(u, k)	Jacobi elliptic function	JACOB	SCUKj
sd(u, k)	Jacobi elliptic function	JACOB	SDUKj
sec z	Secant function	TRIG	SEC
sech z	Hyperbolic secant function	HYP	SECH
Shi(z)	Hyperbolic sine integral	EXPIN	SHIZ
Si(z)	Sine integral	EXPIN	SINT
sign(z)	Sign of z	HP	SIGN

SYMBOL INDEX FOR MATHEMATICAL FUNCTIONS

SYMBOL	FUNCTION	MENU	COMMAND
sin z	Sine function	HP	SIN
sinc(z)	Sinc function = sin(z)/z	EXPIN	SINC
sinh z	Hyperbolic sine function	HYP	SINH
$S_n^{(m)}$	Stirling number of the first kind	NUMB	STRL1
$\mathbf{S}_n^{(m)}$	Stirling number of the second kind	NUMB	STRL2
sn(u, k)	Jacobi elliptic function	JACOB	SNUK¡
$S_n(x)$	Chebyshev polynomial of the first kind	POLY	SOFX
tan z	Tangent function	HP	TAN
tanh z	Hyperbolic tangent function	HYP	TANH
Tr	Trace of matrix	LINAG	TRACE
$T_n(x)$	Chebyshev polynomial of the first kind	POLY	TOFX
$T_n^*(x)$	Shifted Chebyshev polynomial of the first kind	POLY	TSFX
U(a, b, z)	Tricomi's confluent hypergeometric function	CHYPR	UABZ
U(a, x)	Parabolic cylinder function	PCLDR	UOAX
$U_n(x)$	Chebyshev polynomial of the second kind	POLY	UOFX
$U_n^*(x)$	Shifted Chebyshev polynomial of the second kind	POLY	USFX
V(a, x)	Parabolic cylinder function	PCLDR	VOAX
W(a, x)	Parabolic cylinder function	PCLDR	WOAX
$W_{\kappa,\mu}(z)$	Whittaker function	CHYPR	WKμZ
$y_n(z)$	Spherical Bessel function of the second kind	SBESL	SYNZ
$Y_\nu(z)$	Bessel function of the second kind	BESEL	YNOZ
Z(u, k)	Jacobi zeta function	ELLIPI	ZUKP
Z(z)	Normal probability density function	ERROR	ZOFZ
α	Modular angle	THETA	SEVERAL
$\alpha_n(z)$	Exponential integral	EXPIN	αNOZ
$\beta_n(z)$	Exponential integral	EXPIN	βNOZ
β(n)	Complementary Riemann zeta function	NUMB	CZETA
γ	Euler's constant γ = 0.577215664902	EXPIN	γ

SYMBOL INDEX FOR MATHEMATICAL FUNCTIONS

SYMBOL	FUNCTION	MENU	COMMAND
$\gamma(a, z)$	Incomplete gamma function	GAMA	INCγ
$\Gamma(a, z)$	Incomplete gamma function	GAMA	INCG
$\Gamma(z)$	Gamma function	GAMA	GAMMA
$\gamma^*(a, z)$	Incomplete gamma function	GAMA	INCγ¡
$\delta(t)$	Dirac delta function	SYMB	NONE
δ_{nm}	Kronecker delta function (= 0 if $m \neq n$; =1 if $m = m$)	TRIG	PROGRAM
$\Delta(z, k)$	Elliptic function $\Delta(\varphi\backslash\alpha) = \Delta(z, k) = dn(u, k)$	ELLIPI	ΔZKP
$\Delta = g_2^3 - 27g_3^2$	Discriminant associated with Weierstrass elliptic functions	ELLIPI	PROGRAM
$\zeta(n)$	Riemann zeta function	NUMB	RZETA
$\zeta(s, \alpha)$	Generalized Riemann zeta function	NUMB	PROGRAM
$\zeta(z; g_2, g_3)$	Weierstrass zeta function – also $\zeta(z\|\omega, \omega')$	ELLIPI	PROGRAM
$H(u, k)$	Jacobi's eta function	THETA	HUKP
$H_1(u, k)$	Jacobi's complementary eta function	THETA	H1UK¡
$\eta(n)$	$(1 - 2^{1-n})\ \zeta(n)$	NUMB	RZETA
$\Theta(u, k)$	Jacobi's theta function	THETA	θUKP
$\Theta_1(u, k)$	Jacobi's complementary theta function	THETA	θ1UK¡
$\vartheta_1(z\|q)$	Theta function for complex nome	THETA	θ1ZQ
$\vartheta_2(z\|q)$	Theta function for complex nome	THETA	θ2ZQ
$\vartheta_3(z\|q)$	Theta function for complex nome	THETA	θ3ZQ
$\vartheta_4(z\|q)$	Theta function for complex nome	THETA	θ4ZQ
$\vartheta_c(z, k)$	Neville's theta function	THETA	θCUK¡
$\vartheta_d(z, k)$	Neville's theta function	THETA	θDUK¡
$\vartheta_n(z, k)$	Neville's theta function	THETA	θNUK¡
$\vartheta_s(z, k)$	Neville's theta function	THETA	θSUK¡
$\theta_n(z, k)$	Theta function for n = 1, 2, 3, 4	THETA	PROGRAM
$\Lambda_0(x, k)$	Heuman's lambda function	ELLIPI	λXKP
$\lambda(n)$	$(1 - 2^{-n})\ \zeta(n)$	NUMB	RZETA
μ	Mean	STAT	μ

SYMBOL INDEX FOR MATHEMATICAL FUNCTIONS

SYMBOL	FUNCTION	MENU	COMMAND
π	Archimede's constant π = 3.14159265359	HP	π
$\Pi(x)$	$\Pi(x) = x!$	HP	!
$\Pi(n, x, k)$	Incomplete elliptic integral of the third kind	ELLIPI	ΠNXKP
σ	Standard deviation	STAT	σ
$\sigma(z; g_2, g_3)$	Weierstrass sigma function – also $\sigma(z\|\omega, \omega')$	ELLIPI	PROGRAM
Σ	Summation	HP	Σ
φ	Amplitude φ = am(u, k)	JACOB	AMUKP
$\Phi(z, s, \alpha)$	Lerch's transcendent	THETA	PROGRAM
$\psi^{(n)}(z)$	Polygamma function	GAMA	DNPSI
$\psi(z)$	Digamma function = logarithmic derivative of $\Gamma(z)$	GAMA	PSI
ω and ω'	Half-periods associated with Weierstrass elliptic functions	ELLIPI	PROGRAM
$\|z\|$	Absolute value of z	HP	ABS
$(a)_n$	Pochhammer's symbol $\Gamma(a + n)/\Gamma(a)$	MISC	POCH
$\binom{a}{b}$	Binomial coefficient = (a)! / [(a – b)! b!]	HP	COMB
!	Factorial function n! = $\Gamma(n + 1)$	HP	!
(n)!!	Double factorial function	GAMA	!!
$\nabla \phi$	Gradient of function ϕ	SYMB	SGRD
$\nabla \bullet A$	Divergence of vector A	SYMB	SDIV
$\nabla \times A$	Curl of vector A	SYMB	SCURL
$\nabla^2 \phi$	Laplacian of function ϕ	SYMB	$\nabla \bullet [\nabla \phi]$
$z^* = CONJ(z)$	Denotes conjugate – exceptions: $\gamma^*(a, z)$, $T_n^*(x)$, and $U_n^*(x)$	HP	CONJ
^	Exponentiation: 3^2 = 9	HP	^
$\sqrt{x} = \sqrt{}(x)$	Square root of x	HP	$\sqrt{}$
$\sqrt[n]{x}$	nth root of x	HP	XROOT
$\int$	Symbolic or real numerical integral	HP	$\int$
$\int$	Complex contour integral over analytic paths	ERROR	CINTG
$\int$	Complex residue integral over closed paths	SYMB	RESDA

A

B

C

D

E

F

G

H

I

J

K

N

O

P

Q

R

S

T

HP 48SX Engineering Mathematics Library

JOHN F. HOLLAND

CONTENTS

The HP 48SX graphics calculator with MATHLIB provides 1,110 user programmable commands and thousands of complex functions covering the spectrum of applied and engineering mathematics. Put your problem on the stack and press a key for the answer. MATHLIB SOFTWARE IS ENCLOSED.

THE MOST COMPREHENSIVE AND INEXPENSIVE MATHEMATICS CAPABILITY AVAILABLE. MATHLIB will solve a 40 x 40 linear system of equations with iterative refinement in under 4 minutes.

Optional serial interface provides 9600 baud rates between the HP 48SX and both PCs and Macintoshes. Use for data, graphics, and program transfers, as well as computer editing and executing of HP 48 programs.

ISBN 0-12-352380-X