HP 48SX MATH LIBRARY GENERAL APPLICATION PROGRAMS

John F. Holland, Ph.D.

HP 48SX MATH LIBRARY GENERAL APPLICATION PROGRAMS

John F. Holland, Ph.D.

Applications Software for the

HP 48SX Engineering Mathematics Library

An Introduction to Symbolic and Complex Computation with Applications

Published by Academic Press @ 1 (800) 321–5068

TABLE OF CONTENTS

PREFACE

This applications package, like MATHLIB itself, is motivated by a desire to make serious symbolic and complex computation both understandable and affordable. It contains about 370 examples from the MATHLIB manual which show how to apply MATHLIB to your specific applications without searching for typing errors. The moments and evaluation of over 40 probability distributions are also included.

In addition, it contains over 200 symbolic function definitions with defined derivatives. Thirteen additional probability distributions are included as well as 17 Weierstrass elliptic function programs. The large number of Legendre function special cases provide starting functions for the various recurrence formulas. They also demonstrate how to program these functions so as to stay on the correct branch as discussed on pages 37 and 140 of the MATHLIB manual.

As promised, this manual contains a detailed tutorial on the subject of wavelets and related engineering topics including the M–band case. Wavelets seem to be the hottest research topic of interest to the mathematics and engineering communities today. Over 70 related programs are included. The wavelet material is rich with symbolic and numerical algorithms and techniques. Many thanks to Howard Resnikoff and Peter Heller at Aware, Inc. and Ramesh Gopinath at Rice University for many fruitful discussions on this subject.

A number of symbolic and computational techniques are also covered. These include advanced MATHLIB plotting, continued fraction expansions, polynomial greatest common divisor, rounding algebraic equations, advanced symbolic derivative techniques, simulating difficult computations, large integer arithmetic, and various symbolic tricks.

A very fast machine language FFT became available this summer and it is included in this package.

John F. Holland
November, 1992

1

OVERVIEW AND MENU MAPS

INTRODUCTION

The enclosed ROM card contains over 700 MATHLIB application, example, test, and symbolic function programs. The symbolic functions have defined derivatives. This software is broken up into 11 directories so that any one of them may be transferred to your VAR directory for editing and incorporation in your application programs without requiring an additional memory RAM card to be purchased.

By distributing this software on ROM card, it saves you the expense of buying a 128 K RAM card and the PC interface kit. You can load the directory containing the programs of interest into the approximately 30 K of RAM which comes with your calculator and purge the directory when you finish to make room for other programs. Then when you need those programs at a later time, you can again load them from the ROM card. The MAN7 directory is the largest one and is under 21 K bytes in size. If your software came on disk, see Appendix B.

GETTING STARTED

See the HP 48SX owner's manual. The HP 48SX has two ports for installing plug-in cards. This application card 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 push LIBRARY followed by either PORT1 or PORT2 depending upon which port you plugged the ROM into. Now 11 directory names plus one program name are displayed on two menu pages. Program MVER displays the version number and the copyright notice.

USING THE ROM CARD PROGRAMS

In order to use the programs on the ROM card, they must first be transferred to your VAR directory. Push the VAR key and enter the directory in which you wish to store the ROM program or directory. Then push LIBRARY, PORT1 or PORT2, and NXT if required. Now the object you wish to transfer is displayed above the white keys.

Push the white key below the directory (program) you wish to transfer (MAN1 for example). Now that directory has been recalled to the stack as an HP 48 object. To store it in the current VAR directory, you must give it a name. Push the ' key on the HP 48 keyboard and type the name MAN1 for example. Push ENTER and the object will appear on Level 2 of the stack while the name 'MAN1' appears on Level 1 of the stack. Push the STO key and the object will be stored in the VAR directory as MAN1. Now push the VAR key and you will see MAN1 displayed in the first page of the current directory. You may now treat MAN1 as you would any other HP 48 directory (program).

EDITING THE APPLICATIONS PROGRAMS

If you look at the programs without the MATHLIB library attached, calls to the library appear as: XLIB 857 XXX where XXX is a number. The HP 48 editor will not let you edit programs which contain calls to the MATHLIB library unless the library is attached. However, once the library is attached, then the XLIB 857 XXX is replaced by the more familiar command name and you now can edit the programs.

When the MATHLIB ROM is installed, it automatically attaches to the HOME directory. Commands to manually attach or detach MATHLIB are on the second page of the HP 48 MEMORY menu.

RUNNING THE PROGRAMS

The MATHLIB library must be attached to run the programs which use the library. Once MATHLIB is attached, you may run these application programs as you would any other HP 48 programs.

If only one program is of interest, copy the program to another directory and then purge the MATHLIB application directory, so you have more memory available for running your applications.

PURGING THE DIRECTORY

When you have finished with a directory, you may purge it with the **PGDIR** command on the third page of the HP 48 MEMORY menu.

ORGANIZATION OF THE DIRECTORIES

The below table summarizes the 11 directories on the ROM card. The first eight directories follow the MATHLIB manual and include examples from the manual. The three symbolic function and defined derivative directories SD1, SD2, and SD3 follow.

DIRECTORY OVERVIEW	
DIRECTORY	**SUBJECT**
MANUAL RELATED DIRECTORIES	
MAN1	CHAPTERS 1 - 8 PLUS INTAG AND WEF
MAN2	CHAPTERS 9 - 14
LGDR	CHAPTER 15
MAN3	CHAPTERS 16 - 19
MAN4	CHAPTERS 20 - 21
MAN5	CHAPTERS 22 - 25
MAN6	CHAPTERS 26 - 28
MAN7	CHAPTER 29 - APPENDIX H
SYMBOLIC FUNCTION DIRECTORIES	
SD1	ALL BUT ELLIPTIC FUNCTIONS
SD2	ELLIPTIC RELATED FUNCTIONS
SD3	ALTERNATE BESSEL FUNCTIONS

APPLICATION PROGRAM MENU MAPS

Menu maps are given on the next four pages for the programs on the MATHLIB GENERAL APPLICATIONS ROM card.

MENU	SUB	KEY 1	KEY 2	KEY 3	KEY 4	KEY 5	KEY 6
MATHLIB APPLICATIONS 1		*MAN1* *MAN6*	*MAN2* *MAN7*	*LGDR* *SD1*	*MAN3* *SD2*	*MAN4* *SD3*	*MAN5* MVER
MAN1		PLINT PROD2	*PLOT* COMBF	*PLTD* *ERROR*	*INTAG* H1NX	Kδ H2NX	PROD1 *WEF*
MAN1	PLOT	PLTT1	PLTT2	PLTT3	PLTT4		
MAN1	PLTD	DEMY	DEMH	DEMD	DEMI	DEMS	
MAN1	INTAG	ADDS ADDP	SUBS PLOTS	MPYS RNDS	NEGS TYPEA	S→P DERV1	P→S DERV2
MAN1	ERROR	AINTG	CIF	DAWS			
MAN1	WEF	POZG ω→G ω→KP	PPZG ω→E G→Δ	ZOZG G→ω E→Δ	σOZG E→ω ω→Q	G→E G→KP ηOFω	E→G E→KP
MAN2		*ELLIPI*	*JACOB*	*THETA*	*CHYPR*	PCLDR1	*GHYP*
MAN2	ELLIPI	ELLIPI1 ELLIPI6 ZUKP1	KOKP1 ELLIPI7 λXKP1	ELLIPI2 ELLIPI8 ΔZKP1	ELLIPI3 ZXKP UZKP1	ELLIPI4 ZXK_i	ELLIPI5 ZTBL
MAN2	JACOB	JACOB1	JACOB2	UZKP1	KOKP1		
MAN2	THETA	POLYL THETA5 $\theta 3ZK_j$ PZSαN	FX THETA6 $\theta 4ZK_j$ ZSαN	THETA1 THETA7 $T1UK_i$	THETA2 THETA8 $T2UK_i$	THETA3 $\theta 1ZK_j$ $T3UK_i$	THETA4 $\theta 2ZK_j$ $T4UK_i$
MAN2	CHYPR	JVOZ1 GLηρ	YVOZ1 F0F1	IVOZ1 F1F0	KVOZ1	ΔABZ	FLηρ
MAN2	GHYP	INCBH1	GHYP1				
LGDR		PSSV PQUP3 PMMZ	WSSV PQUP4 PMNX	PμVX1 PQUP5 PMNZ	QμVX1 PμVZ1 STRUVE	PQUP1 QμVZ1 *TEST*	PQUP2 PMMX
LGDR	TEST	PTST *X.5X*	*PI*	*QI*	*PIX*	*QIX*	*X.5*
LGDR	TEST PI	P11Z	P12Z	P13Z	P22Z	P23Z	P33Z
LGDR	TEST QI	Q10Z Q12Z	Q20Z Q22Z	Q30Z Q32Z	Q11Z Q13Z	Q21Z Q23Z	Q31Z Q33Z
LGDR	TEST PIX	P11Z	P12Z	P13Z	P22Z	P23Z	P33Z
LGDR	TEST QIX	Q10Z Q12Z	Q20Z Q22Z	Q30Z Q32Z	Q11Z Q13Z	Q21Z Q23Z	Q31Z Q33Z
LGDR	TEST X.5	PP.5	PM.5	QP.5	QM.5	PVVZ	
LGDR	TEST X.5X	PP.5	PM.5	QP.5	QM.5	PVVZ	

MENU	SUB	KEY 1	KEY 2	KEY 3	KEY 4	KEY 5	KEY 6
MAN3		*POLY*	*MISC*	*NUMB*	*ALGB*		
MAN3	POLY	PαβX1	COX1	TOFX1	UOFX1	POFX1	
MAN3	MISC	CSERS1 RATAP1 LSSOV2	TALR1A LREG1 UNIQE1	FEVAL1 LREG2 SORT1	PMAT1 LREG3	PMAT2 LSSOV1	PEQN1 CMAT
MAN3	NUMB	GCD1	FACTR1	PRIME1	FIBON1	BR1	ER1
MAN3	ALGB	TALR1A EVLΣ1 PDERV1 *DERD*	PADD1 SUMΣ RDERV1 CFE	PSUB1 PROOT1 PINTG1 ECFE	PMPY1 AROOT1 PEQN1 PGCD	PDVD1 LROOT1 PLDVD1 PGCDI	FEVAL1 DEFLT1 *IDEN*
MAN3	ALGB IDEN	MTHS CNJE EX3	PARE SREIM EX4	PARED CREIM EX5	POLYE TOEXP EX6	TRIGE EX1 EX7	CEXPE EX2
MAN3	ALGB DERD	GAMMAS DEX1	PSIS DEX2	DNPSIS DEX3	der■!	derGAMMAS	derPSIS
MAN4		*LINAG*	*MATR*	MKSQ			
MAN4	LINAG	LSOV1 SVD2 HBDD1 EIGEN1	LSOV2 B←M1 HTRDD1 CHOLD1	LSOV3 ORDER1 UHESD1 CPOLY1	LSOV4 GLUD1 SCRSD1 SVDE	LSOV5 LDLTD1 SCHRD1 EVSOV	SVD1 HRQR1 EIGNS1
MAN4	MATR	RROLL CSORT1 EPSM1 SROW1 →VTR1 M→CL1 B→M1 DIAG?	CROLL CSRTI1 RPSM1 SCOL1 →VTR2 M←CL1 B←M1	→ROW1 ERWS1 IROW1 MROW1 ESBM1 RNLV1 B←M2	→COL1 ECOLS1 ICOL1 MCOL1 RSBM1 CNLV1 B←M3	RSORT1 RRWS1 DROW1 CROW1 M→RL1 RSPLT1 DIAG1	RSRTI1 RCOLS1 DCOL1 CCOL1 M←RL1 CSPLT1 COMP1
MAN5		*STAT*	*PROB*	LTBN	*SYMB*		
MAN5	STAT	FZT1 AVAR1A MWT1 SRHO1	FZT2 AVAR1B WILCX KS1S	FISHZ AVAR2A WILC1 KS1P	LCNTτ ACOVR1 KWT KS2S	CTA2D1 RANK KWT1 FOFC	SRCTT1 MWT SRHO FOFCS
MAN5	PROB	LOGN WALD	MAXW PRETO	SECHS	FRMID	BOSEE	GEOM
MAN5	SYMB	S2F1 SGRD1 IXFRM1 IXFRM7 SMI1 TEA	SYMB1 SDIV1 IXFRM2 A SMI2 VLINI	SYMB2 SCURL1 IXFRM3 EATN SMI3 VVOLI	SYMB3 DESOL1 IXFRM4 EATE SMI4	SYMB4 DESOL2 IXFRM5 EAT EASOV	SYMB5 DESOL3 IXFRM6 RESOV A2
MAN6		*PROC*	*FILTR*	*WIND*	DFT	IDFT	
MAN6	PROC	DER2A ΔWAV1 TINV MUX TM	LINT1 ICNVN MINλ DMUX H3ON	FFT1 TIRRC MAXλ SPLIT	WL1AP ICNVS DM TSTW	WL2AP TIRRD FFTLC TMIC3	SQWV1 DATA FILTR ITM

MENU	SUB	KEY 1	KEY 2	KEY 3	KEY 4	KEY 5	KEY 6
MAN6	FILTR	FTRV2A τVT2B ESOLV1 L→BP1 BILNT3 IXFRM2 LSS1	FTRV2B FTRVL1 BESF L→BP2 BILNT4 IXFRM3 LSS2	IDEM1 POLEP1 L→LP1 L→BS1 DHBRT1 IXFRM4	IDEM2 POLEP2 L→LP2 L→BS2 GAIN1A RATAPτ	IDEM3 BPOLE1 L→HP1 BILNT1 GAIN1B S→ZPτ	τVT2A CPOLE1 L→HP2 BILNT2 IXFRM1 JORDAN
MAN6	WIND	PLTW E3 ZASOV	FIR E6 SMS	IXFRM1 WF1 IXFRMD	PLDVD1 WF2 HNS	FMAT1 TTOFXL KAISER	DOCAP1 TESTF
MAN7		*VECTR*	*APPH*	*WAVE*	*FFTML*		
MAN7	VECTR	PTOM	REVS	CFG			
MAN7	APPH	PKEY KEYP	TMPM KEY0	CSTM KEYM	KSIN	EXMP	PMENU
MAN7	WAVE	*FFT2D* SINDX	*HAAR* SDEC	*SOLVE* SINTP	*SYMB* SCMB	QMFL LNLV	QMF LDLV
MAN7	WAVE FFT2D	FFT2 IDWT2	IFFT2 IDWT3	WMOM CLEN	SOVJ DWT2.1	DEMO IDWT2.1	DWT2
MAN7	WAVE HAAR	DFTM DHPM R4G8	CHAAR DHP TROW	COSM WD4 TESTV	CBYM WD6 SWD4	HDM R2G4 SWD6	KPRD R4G4 TRWS
MAN7	WAVE SOLVE	D2G DMG CLPCF POFZ	QOFZ COSM LPFIL POMY	POFY CWM WSOV MOMZ	MOZ SOVN SSOV *COPT*	MCOZ VALUE SMAT QOZ1	CLPM CRN SSUM Y→ZT
MAN7	WAVE SYMB	LAURT CMPL SPTLR	PTRN CMPG BEXCO	PPRD1 CLPC BMLT	PPRD2 EOFM P→S	ZINV CGC S→P	ZFRM ROFM Z→ω
MAN7	FFTML	FFTM	IFFTM	ABSV	ANGL	CONVM	
SD1		*TRIG* *SBESL* POCHS KVOZS derJVOZS	*HYP* *HGM* JVOZS der■! derYVOZS	*EXPI* *LGDR* YVOZS derGAMMAS derH1VZS	*GAMA* GAMMAS H1VZS derPSIS derH2VZS	*ERROR* PSIS H2VZS derDNPSIS derIVOZS	*BESLX* DNPSIS IVOZS derPOCHS derKVOZS
SD1	TRIG	CSCS derCSCS	ACSCS derACSCS	SECS derSECS	ASECS derASECS	COTS derCOTS	ACOTS derACOTS
SD1	HYP	CSCHS derCSCHS	ACSCHS derACSCHS	SECHS derSECHS	ASECHS derASECHS	COTHS derCOTHS	ACOTHS derACOTHS
SD1	EXPI	ENOZS CHIZS derSHIZS	SINTS derENOZS derCHIZS	CINTS derSINTS	SINCS derCINTS	SINC2S derSINCS	SHIZS derSINC2S
SD1	GAMA	BETAS PERMFS derINCβS	INCGS COMBFS derPERMFS	INCγS derBETAS derPERM	INCγiS derINCGS derCOMBFS	INCβS derINCγS	INCBS derINCγiS
SD1	ERROR	ERFZS derCOFZS	ERFCZS derSOFZS	COFZS	SOFZS	derERFZS	derERFCZS
SD1	BESLX	COPY SD3 BESLX: YNOXS – KNOXS & EDIT SD1 "V" DERIVATIVES: V → N AND Z → X					

MENU	SUB	KEY 1	KEY 2	KEY 3	KEY 4	KEY 5	KEY 6
SD1	SBESL	SJNZS SKNZS derSH2ZS	SYNZS AIOZS derSI1NZS	SH1ZS BIOZS derSI2NZS	SH2ZS derSJNZS derSKNZS	SI1NZS derSYNZS derAIOZS	SI2NZS derSH1ZS derBIOZS
SD1	HGM	MABZS derUABZS	UABZS derF2F1S	F2F1S derF0F1S	F0F1S derF1F0S	F1F0S	derMABZS
SD1	LGDR	PμVXS derPμVZS	QμVXS derQμVZS	PμVZS	QμVZS	derPμVXS	derQμVXS
SD2		*ELLIPI*	*JACOB*	*THETA*			
SD2	ELLIPI	FXKPS EUKPS derZXKPS	EXKPS ΠNUKPS derEUKPS	BXKPS ZUKPS derΠNUKPS	DXKPS derFXKPS derZUKPS	ΠNXKPS derEXKPS	ZXKPS derΠNXKPS
SD2	JACOB	SNUK$_i$S DCUK$_i$S derSNUK$_i$S derDCUK$_i$S	CNUK$_i$S NCUK$_i$S derCNUK$_i$S derNCUK$_i$S	DNUK$_i$S SCUK$_i$S derDNUK$_i$S derSCUK$_i$S	CDUK$_i$S NSUK$_i$S derCDUK$_i$S derNSUK$_i$S	SDUK$_i$S DSUK$_i$S derSDUK$_i$S derDSUK$_i$S	NDUK$_i$S CSUK$_i$S derNDUK$_i$S derCSUK$_i$S
SD2	THETA	θSUK$_i$S θ3ZQS T1UQS derθDUK$_i$S	θCUK$_i$S θ4ZQS T2UQS derθNUK$_i$S	θDUK$_i$S TSUK$_i$S T3UQS derθ1ZQS	θNUK$_i$S TCUK$_i$S T4UQS derθ2ZQS	θ1ZQS TDUK$_i$S derθSUK$_i$S derθ3ZQS	θ2ZQS TNUK$_i$S derθCUK$_i$S derθ4ZQS
SD3		*BESLX* "IVOZS" derIVOZS	*SBESL* "KVOZS" derKVOZS	"JVOZS" derIVOZS	"YVOZS" derYVOZS	"H1VZS" derH1VZS	"H2VZS" derH2VZS
SD3	BESLX	YNOXS derH2NXS	H1NXS derKNOXS	H2NXS	KNOXS	derYNOXS	derH1NXS
SD3	SBESL	"SJNZS" "SKNZS" derSH2ZS	"SYNZS" "AIOZS" derSI1NZS	"SH1ZS" "BIOZS" derSI2NZS	"SH2ZS" derSJNZS derSKNZS	"SI1NZS" derSYNZS "derAIOZS"	"SI2NZS" derSH1ZS "derBIOZS"
"PROGRAM" MEANS AFTER COPYING PROGRAM FROM SD1 DIRECTORY TO SD3 DIRECTORY							

EXAMPLE - LARGE INTEGER ARITHMETIC

Suppose you wish to perform large integer exact arithmetic. Push HOME. Then push LIBRARY, PORT1 or PORT2, and MAN1 to recall the MAN1 directory to the stack. Now type ' M A N 1 ENTER. Now the directory is on Level 2 of the stack and the name 'MAN1' is on Level 1 of the stack. Push STO to store the directory and push the VAR key to display it. The MAN1 directory is now displayed above the left-most white key on the HP 48. Pushing the MAN1 key (the white key below the MAN1), enters the directory. Now the first six menu names of the MAN1 directory are displayed. Push the fourth key under INTAG. Now the twelve programs in the INTAG subdirectory are displayed on two pages (push NXT key) and you are ready to compute.

These programs use the string object as a way of representing large integers on the HP 48. Arithmetic is performed by converting the string to a polynomial with S→P, performing the operation, and converting the resulting polynomial back to a string with P→S. The programs S→P, P→S, and ADDP are internal programs and should not be used directly. The operations available in this directory are:

1. ADDS provides addition
2. SUBS provides subtraction
3. MPYS provides multiplication
4. NEGS provides negation

Suppose we wish to add the base 10 numbers 123456789123456789 and 1111. Type " 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 9 ENTER " 1 1 1 1 ENTER. Now the stack contains:

2: "123456789123456789"
1: "1111"

and we are ready to add. Push the ADDS key for the result "00123456789123457900", which can be viewed by pushing the EDIT (left-shifted +/–) key. Note that the left most 00 is not significant while the right most 00 is significant. Now suppose we need to negate the result. Push NEGS and observe that the result is that a minus sign has been added to the string giving "–00123456789123457900" for the result. The NEGS program is convenient because the HP 48 editor will not let you type a minus sign in an empty string. However, " 1 2 +/– ENTER does put "–12" on the stack.

To cube the number 123456789, put "123456789" on Level 1 of the stack and press ENTER twice to copy it up to Level 3 of the stack. Two pushes of the MPYS key yields the exact cube of 123456789 which equals "1881676371789154860897069".

Now when you are finished with the MAN1 directory, you may purge it. Push HOME since we stored MAN1 in the HOME directory. Now push MEMORY NXT NXT ' M A N 1 ENTER PGDIR to purge MAN1 from the HOME directory.

The last five programs in the INTAG subdirectory are discussed in Chapter 7.

2

MANUAL PROGRAM SUMMARIES

INTRODUCTION

The below 8 tables summarize the application programs contained in this manual and its related program directories. Page references enclosed with brackets [ ] refer to the MATHLIB manual, not to this manual. When the comment applies to all of the programs contained in a subdirectory, then only the subdirectory name is given.

MAN1 DIRECTORY

The MAN1 directory contains programs related to Chapters 1 through 8 of the MATHLIB manual. It also contains programs for performing large unlimited precision arithmetic. The operation of these integer arithmetic programs is discussed in Chapter 3 of this manual.

Seventeen Weierstrass elliptic function and related programs are also given in this directory. These are discussed in Chapter 8 of this manual.

The MAN1 INTAG subdirectory also contains advanced graphics software and programs for rounding numbers in symbolic equations. Programs for advanced derivative techniques are also given. These programs are discussed in Chapter 7.

MAN1 DIRECTORY		
SUB DIR	PROGRAM	COMMENTS
	PLINT	LAGRANGE INTERPOLATION [14]
PLOT		PLOTTING DEMONSTRATIONS [19 - 23]
PLTD		PLOTTING DEMONSTRATIONS [21, 24]
	Kδ	KRONECKER DELTA FUNCTION δ_{mn} [39]
	PROD1, PROD2	PRODUCT EVALUATION [40]
INTAG	ADDS	LARGE INTEGER ADDITION
INTAG	SUBS	LARGE INTEGER SUBTRACTION
INTAG	MPYS	LARGE INTEGER MULTIPLICATION
INTAG	NEGS	LARGE INTEGER NEGATION
INTAG	S→P	CONVERT TO POLYNOMIAL
INTAG	P→S	CONVERT FROM POLYNOMIAL
INTAG	ADDP	ADDITION IN POLYNOMIAL FORM
INTAG	PLOTS	SPECIAL PLOT PROGRAMS
INTAG	RNDS	ROUND SYMBOLIC EQUATIONS
INTAG	TYPEA	ALGEBRAIC TYPE PROGRAM
INTAG	DERV1	ADVANCED DERIVATIVE TECHNIQUES
INTAG	DERV2	ADVANCED DERIVATIVE TECHNIQUES
	COMBF	COMPLEX COMBINATIONS [55]
ERROR	AINTG	COMPLEX PLANE INTEGRATION [58, 62]
ERROR	CIF	CAUCHY'S INTEGRAL FORMULA DEMO [58, 62]
ERROR	DAWS	DAWSON'S INTEGRAL [62]
	H1NX, H2NX	HANKEL FUNCTIONS [72]
WEF		WEIERSTRASS ELLIPTIC FUNCTION PROGRAMS

MAN2 DIRECTORY

The MAN2 directory covers Chapters 9 through 14 in the MATHLIB manual. The majority of these are examples from the manual.

The polylogarithm function, Lerch's transcendent, and the Riemann generalized zeta function are also included in this directory.

MAN2 DIRECTORY		
SUB DIR	PROGRAM	COMMENTS
ELLIPI	ELLIPI1 - ELLIPI8	MANUAL EXAMPLES [83 - 85]
ELLIPI	ZXKP, ZXK$_i$	JACOBI ZETA FUNCTION [86]
ELLIPI	ZTBL	COMPUTES TABLE 17.7 IN AMS 55 [86]
ELLIPI	ZUKP1 - UZKP1	MANUAL EXAMPLES [87 - 89]
JACOB		MANUAL EXAMPLES [94, 95]
THETA	POLYL, FX	POLYLOGARITHM FUNCTIONS [112]
THETA	THETA1 - THETA8	MANUAL EXAMPLES [101 - 110]
THETA	θ1ZK$_i$ - T4UK$_i$	ALTERNATE THETA FUNCTIONS [99, 111]
THETA	PZSαN	LERCH'S TRANSCENDENT $\Phi(z,s,\alpha)$ [112]
THETA	ZSαN	RIEMANN ZETA FUNCTION $\zeta(s,\alpha)$ [112, 190]
CHYPR	JVOZ1 - KVOZ1	MANUAL EXAMPLES [117, 118]
CHYPR	ΔABZ	U(a,b,z) AT THE BRANCH CUT [120]
CHYPR	FL$\eta\rho$, GL$\eta\rho$	COULOMB WAVE FUNCTIONS [120 - 122]
CHYPR	F0F1, F1F0	HYPERGEOMETRIC FUNCTIONS [124]
	PCLDR1	TEST PROGRAM
GHYP		MANUAL EXAMPLES [133]

LGDR DIRECTORY

This is the Legendre and Struve function directory. In addition to the examples in the manual, it contains a large number of special cases, listed in the MATHLIB manual. Deriving all these special cases is harder than one might think because it is surprisingly easy to make errors. These special cases provide starting functions for the various recurrence relations given in Chapter 15 of the MATHLIB manual. They also demonstrate how to program these functions so as to stay on the correct branch as discussed on pages 37 and 140 of the MATHLIB manual. The three X suffix MAN2 LGDR TEST subdirectories are the X "on the cut" programs and the other three are the general complex plane programs.

LGDR DIRECTORY		
SUB DIR	PROGRAM	COMMENTS
	PSSV, WSSV	$P_n(z)$ AND $Q_n(z)$ SYMBOLIC RECURRENCE [140]
	PμVX1, QμVX1	MANUAL EXAMPLES [150]
	PQUP1 – PQUP5	MANUAL EXAMPLES [150, 151]
	PμVZ1, QμVZ1	MANUAL EXAMPLES [152]
	PMMX – PMNZ	$P_n^m(x)$ AND $P_n^m(z)$ [152, 153]
	STRUVE	STRUVE FUNCTION EXAMPLES [154]
TEST PI		EXAMPLE $P_n^m(z)$ [139, 140]
TEST QI		EXAMPLE $Q_n^m(z)$ [141]
TEST PIX		EXAMPLE $p_n^m(x)$ [142]
TEST QIX		EXAMPLE $q_n^m(x)$ [142, 143]
TEST X.5		EXAMPLES [144]
TEST X.5X		EXAMPLES [144]

MAN3 DIRECTORY

The MAN3 directory covers the examples in Chapters 16 through 19 in the MATHLIB manual. Included are the algebraic identity and symbolic calculus programs discussed in Chapter 19. The symbolic function directories discussed in the next chapter contain a far more extensive set of derivative definitions.

MAN3 DIRECTORY		
SUB DIR	PROGRAM	COMMENTS
POLY		TEST PROGRAMS [157 – 158]
MISC		MANUAL EXAMPLES [161 – 183]
NUMB		MANUAL EXAMPLES [186 – 189]
ALGB		MANUAL EXAMPLES [194 – 206]
ALGB	CFE – ECFE	CONTINUED FRACTION EXPANSIONS
ALGB	PGCD – PGCDI	POLYNOMIAL GREATEST COMMON DIVISOR
ALGB IDEN	MTHS – TOEXP	MANUAL PROGRAMS [207 – 211]
ALGB IDEN	EX1 – EX7	MANUAL EXAMPLES [211 – 214]
ALGB DERD	GAMMAS–derPSIS	MANUAL PROGRAMS [215]
ALGB DERD	DEX1 – DEX3	MANUAL EXAMPLES [216]

MAN4 AND MAN5 DIRECTORIES

These directories cover the examples in Chapters 20 through 25 of the MATHLIB manual. It is surprisingly easy to make typos when typing in a matrix. Consequently, these directories provide them. MAN5 also gives 13 additional probability distributions not in MATHLIB.

MAN4 DIRECTORY		
SUB DIR	PROGRAM	COMMENTS
LINAG	LSOV1 - CPOLY1	MANUAL EXAMPLES [228 - 252]
LINAG	SVDE	SVDE COMPUTES VALUES RELATED TO THE SVD OF INPUT MATRIX A USING 2 SCHUR DECOMPOSITIONS. COMPARE WITH SVD.
LINAG	EVSOV	EIGENVECTOR PROGRAM [254]
MATR	RROLL, CROLL	MANUAL PROGRAMS [255]
MATR	→ROW1 - COMP1	MANUAL EXAMPLES [257 - 269]
MATR	DIAG?	MANUAL PROGRAM [270]
	MKSQ	MAKES MATRIX WITH R < C SQUARE

MAN5 DIRECTORY		
SUB DIR	PROGRAM	COMMENTS
STAT	FZT1 - LCNTτ	LINEAR CORRELATION PROGRAMS [282]
STAT	CTAD1 - MWT1	MANUAL EXAMPLES [284 - 295]
STAT	WILCX - SRHO1	MANUAL EXAMPLES [471, 472]
STAT	KS1S	LOWER TAIL KS 1 SAMPLE D_N DIST
STAT	KS1P	UPPER TAIL KS 1 SAMPLE D_N^+ DIST
STAT	KS2S	LOWER TAIL KS 2 SAMPLE D_{MN} DIST
STAT	FOFC	$c = \|\rho\|^2$ DIST FOR SAMPLE CORRELATION COEF
STAT	FOFC2	$C = \|P\|^2$ DISTRIBUTION FOR NON-ZERO MEANS
PROB	LOGN	LOWER TAIL LOG-NORMAL DISTRIBUTION
PROB	MAXW	LOWER TAIL MAXWELL DISTRIBUTION
PROB	SECHS	LOWER TAIL SECH-SQUARED DISTRIBUTION
PROB	FRMID	LOWER TAIL FERMI-DIRAC DISTRIBUTION
PROB	BOSEE	LOWER TAIL BOSE-EINSTEIN DISTRIBUTION
PROB	GEOM	LOWER TAIL GEOMETRIC DISTRIBUTION
PROB	WALD	LOWER TAIL WALD DISTRIBUTION

MAN5 DIRECTORY		
SUB DIR	PROGRAM	COMMENTS
PROB	PRETO	UPPER TAIL PARETO DISTRIBUTION
	LTBN	BIVARIATE NORMAL DISTRIBUTION [314]
SYMB	S2F1	SYMBOLIC MATRIX ELEMENT OPERATIONS
SYMB	SYMB1 - TEA	MANUAL EXAMPLES AND PROGRAMS [328 - 364]
SYMB	VLINI, VVOLI	VECTOR INTEGRATION EXAMPLES [366]

MAN6 AND MAN7 DIRECTORIES

These directories contain example programs from Chapter 26 through the end of the MATHLIB manual. MAN7 also contains example wavelet programs which are discussed in more detail in Chapters 9 through 11 of this manual. The machine language FFT directory is discussed in Chapter 12.

MAN6 DIRECTORY		
SUB DIR	PROGRAM	COMMENTS
PROC	DER2A - ΔWAV1	MANUAL EXAMPLES [372 - 380]
PROC	ICNVN - H3ON	MANUAL APPLICATION PROGRAMS [382 - 390]
FILTR		MANUAL EXAMPLES [398 - 427]
WIND		EXAMPLE PROGRAMS [432 - 447]
WIND	KAISER	EXAMPLE WINDOW PROGRAM
	DFT	DISCRETE FOURIER TRANSFORM (DFT) [390]
	IDFT	INVERSE DFT PROGRAM [390]

MAN7 DIRECTORY		
SUB DIR	PROGRAM	COMMENTS
VECTR		EXAMPLE PROGRAMS [450]
APPH		EXAMPLE PROGRAMS [575 - 577]
WAVE		WAVELET PROGRAMS [566 - 572]
FFTML		DONATED MACHINE LANGUAGE FFT

3

SYMBOLIC FUNCTION PROGRAMS

INTRODUCTION

This chapter introduces the three symbolic function and defined derivative directories. SD1 is the largest symbolic function directory on the ROM. This was necessary in order to keep together the necessary functions. The elliptic functions are to some degree separable and are in SD2. Alternative defined derivative definitions for the Bessel functions are given in SD3 which have derivative formulas which are defined at zero. If these directory layouts do not meet your needs, create your own directory using the SD1, SD2, and SD3 programs.

THEORY OF OPERATION

While in a given directory or subdirectory, the HP 48 will automatically evaluate all the programs in that directory. If it does not find a particular function, then it searches the higher level directories for it. This is discussed in Chapter 7 of the HP 48 owner's manual. Consequently, to access and use the symbolic function and defined derivative programs, you simply enter the directory where the function of interest is located and proceed to use its definition. Examples are given below and at the end of Chapter 19 of the MATHLIB manual. *Please note that with the exception of **BETA, PERM, PERMF, COMBF,** and **POCH**, derivatives are only defined with respect to the single major argument, and not with respect to degree, order, or modulus.*

FINDING FUNCTIONS OF INTEREST

The symbolic functions are generally laid out in directories by function type. Exceptions for SD1 are that GAMMAS, PSIS, DNPSIS, POCHS, JVOZS, YVOZS, H1VZS, H2VZS, IVOZS, and KVOZS are in the top level of the SD1 directory so that they are available to all the lower level directories. See the menu maps. Note that the symbolic function names are the MATHLIB names with an S suffix. Some copying and editing is required to fill out the SD1 and SD3 directories. See menus.

EXAMPLE SESSION

Suppose you need to evaluate the complex derivative of the equation

$$\frac{z^3 \cos(4z)\,\mathrm{erf}(2z)}{\Gamma(z^2)}$$

where erf(z) is the error function and Γ(z) is the gamma function. Push HOME and PURGE the object z if it exists in the HOME directory. The keystrokes are ' z ENTER PURGE. Now push LIBRARY, either PORT1 or PORT2, NXT, and SD1. Now the SD1 directory is on Level 1 of the stack. To name it type ' S D 1 ENTER, and push STO to store the SD1 directory. Now push VAR and the SD1 directory will appear under the first white key. Push that key and then push the ERROR key to enter the error function subdirectory. Note that the error function is in this directory and the gamma function is in the parent directory. Thus all the required functions are defined. Type

« 'z^3×COS(4×z)×ERFZS(2×z)/GAMMAS(z^2)' z ∂ EXCO »

and enter this program on Level 1 of the stack. Now push EVAL for the result

```
'-(2×ERFZS(2×z)×PSIS(z^2)/GAMMAS(z^2)×COS(4×z)×z^4)
+2.25675833418/GAMMAS(z^2)×COS(4×z)×.135335283237^z^(2×z)
×z^3+3×ERFZS(2×z)/GAMMAS(z^2)×COS(4×z)×z^2
-4×ERFZS(2×z)/GAMMAS(z^2)×SIN(4×z)×z^3'
```

I suggest you use EDIT to view the equation since the HP 48 equation editor takes a long time and you may run out of memory. To evaluate this derivative at z = (1, 0.2), put { z (1,.2) } on Level 1 of the stack and push | (α right-shift VAR on the HP 48 keyboard) followed by ENTER. The answer is (-4.37712900266, 6.65709260038).

4

LERCH'S TRANSCENDENT AND RELATED FUNCTIONS

INTRODUCTION

There are a number of useful functions which are related to Lerch's transcendent defined on page 112 of the MATHLIB manual. However, definitions do vary and in this chapter we discuss the various definitions.

LERCH'S TRANSCENDENT

Lerch's transcendent is defined by the equations

$$\Phi(z,\ s,\ \alpha) = \sum_{n=0}^{\infty} (\alpha + n)^{-s}\ z^{n} \qquad |z| < 1$$

$$\Phi(z,\ s,\ \alpha) = \frac{1}{\Gamma(s)} \int_0^{\infty} \frac{t^{s-1}\ e^{-\alpha t}}{1 - z e^{-t}}\ dt = \frac{1}{\Gamma(s)} \int_0^{\infty} \frac{t^{s-1}\ e^{(1-\alpha)t}\ dt}{e^{t} - z}$$

RE $\alpha > 0$, and either z is not on the branch cut from 1 to ∞ with RE $s > 0$ or $z = 1$ and RE $s > 1$.

The generalized Riemann zeta function $\zeta(s, \alpha) = \Phi(1, s, \alpha)$ is a special case which is evaluated by ZSαN in the MAN2 directory. In general, PZSαN evaluates Lerch's transcendent. The N argument of these programs sets the integration precision. N = 7 usually works. Φ is useful in evaluating certain integrals in statistical quantum mechanics. See Tolman, for example. POLYL evaluates the polylogarithm function.

POLYLOGARITHM FUNCTION

There are several definitions of the polylogarithm function currently in use. The MATHLIB definition follows that of Erdelyi and Magnus (see the Chapter 11 references in the MATHLIB manual). We define the polylogarithm as $\mathfrak{L}_s(z) = z\Phi(z, s, 1)$. In Chapter 27, AMS 55 only discusses Euler's dilogarithm. Their definition is $f(x) = \mathfrak{L}_2(1 - x)$. Spanier proposes the definition $diln(x) = -\mathfrak{L}_2(1 - x)$. The dilogarithm is also called Spence's integral for n = 2 and is related to the Debye function. The connection between the dilogarithm function and Lerch's transcendent may be seen by comparing the power series expansions. AMS 55 defines the dilogarithm as

$$f(x) = -\int_1^x \frac{\ln t}{t - 1}\, dt = \sum_{n=1}^{\infty} \frac{(1 - x)^n}{n^2}$$

where the sum converges for $0 \le x \le 2$. We also have the sum

$$z\Phi(z, 2, 1) = \mathfrak{L}_2(z) = \sum_{n=1}^{\infty} \frac{z^n}{n^2} \qquad |z| \le 1$$

From Lerch's transformation, one obtains Jonquiere's relation

$$\mathfrak{L}_s(z) + e^{is\pi}\, \mathfrak{L}_s(1/z) = \frac{(2\pi)^s}{\Gamma(s)} e^{i\pi s/2}\, \zeta\left(1 - s, \frac{\ln z}{2\pi i}\right)$$

For $m = 1, 2, 3, \ldots$, $\mathfrak{L}_{-m}(z) = (-1)^{m+1}\mathfrak{L}_{-m}(1/z)$ and

$$\mathfrak{L}_m(z) + (-1)^m\, \mathfrak{L}_m(1/z) = -\frac{(2\pi i)^m}{m!}\, B_m\left(\frac{\ln z}{2\pi i}\right)$$

for $m = 2, 3, 4, \ldots$. This corrects equation 18 on page 31 of Erdelyi, Volume I. These equations provide analytic continuation of the basic series for $|z| > 1$. Similarly, $f(x) + f(1/x) = -[\ln x]^2/2$ for $0 \le x \le 1$ and $f(x) + f(1 - x) = -\ln x \ln(1 - x) + \pi^2/6$ for $0 \le x \le 1$. While $\mathfrak{L}_s(z)$ has the same branch cut as Φ, f has the negative real axis for its branch cut. All of these definitions are easily programmed with MATHLIB and programs for POLYL and FX are given in the MAN2 THETA directory. FX(.35, 7) = .80608268951, POLYL(.5, 1.7, 7) = .605426124605, and PZSαN(1, .5, 2.6, 7) = ZSαN(.5, 2.6, 7) = -2.90496492738. Also we have ZSαN(3, 1, 7) = PZSαN(1, 3, 1, 7) = RZETA(3).

Spanier, J., and Oldham, K., *An Atlas of Functions*, New York, Hemisphere Publishing Corp., 1987.

Tolman, R., *The Principles of Statistical Mechanics*, New York, Oxford University Press, 1938.

See Chapter 11 of the MATHLIB manual for the other references.

5

PROBABILITY AND CHARACTERISTIC FUNCTIONS

INTRODUCTION

This chapter overviews probability functions and related moments such as characteristic functions. It provides numerous formulas for various probability functions and their moments. Programs for the log–normal, Maxwell, sech–squared, Fermi–Dirac, Bose–Einstein, Wald, Pareto, and geometric distributions are given in the MAN5 directory. The others are on the MATHLIB ROM card.

PROBABILITY FUNCTIONS

A real-valued function F(x) is called a (univariate) *cumulative distribution function* if

1. F(x) is non-decreasing, that is, $F(x_1) \leq F(x_2)$ whenever $x_1 \leq x_2$
2. F(x) is continuous from the right
3. $F(-\infty) = 0$ and $F(\infty) = 1$

The cumulative distribution function can be represented as the integral of the *probability density function* f(x) by

$$F(x) = \int_{-\infty}^{x} f(\alpha)\, d\alpha = \int_{-\infty}^{x} dF(\alpha)$$

To avoid more formal Lebesgue counting measure, we allow f(x) to contain distribution functions such as the Dirac delta function discussed in Appendix F of the MATHLIB manual. As discussed on page 547, we now define the unit step function u(x) such that u(0) = 1, so that it will be continuous from the right (see page 476 of the MATHLIB manual). With this formalism, we can now treat discrete and continuous distributions together. For example, consider the lower tail of the Poisson distribution (see page 305 in the MATHLIB manual)

$$F(x) = \sum_{n=0}^{x-1} e^{-m} \frac{m^n}{n!} = \sum_{n=0}^{\infty} e^{-m} \frac{m^n}{n!} u(x-n-1) \qquad x = 0, 1, 2, \ldots$$

Thus, the Poisson probability density function is

$$f(x) = \sum_{n=0}^{\infty} e^{-m} \frac{m^n}{n!} \delta(x-n-1)$$

EXPECTATIONS AND MOMENTS

The expected value of a function g(x) is

$$E\{g(x)\} = \int_{-\infty}^{\infty} g(x)\, f(x)\, dx = \int_{-\infty}^{\infty} g(x)\, dF(x)$$

Examples include the nth moment about the origin

$$\mu'_n = \int_{-\infty}^{\infty} x^n f(x)\, dx$$

where $\mu'_1 = m = \mu$ is the mean, the nth central moment

$$\mu_n = \int_{-\infty}^{\infty} [x - m]^n f(x)\, dx$$

where $\mu_2 = \mu'_2 - m^2 = \sigma^2$ is the variance, and the characteristic function

$$\Phi(i\omega) = \phi(\omega) = \int_{-\infty}^{\infty} e^{i\omega x} f(x)\, dx$$

where $\Phi(y)$ is called the moment generating function and $\phi(\omega)$ is the characteristic function. Observe that from the Maclaurin series expansion of $e^{i\omega x}$ that the moments about the origin equal

$$\mu'_n = i^{-n} \phi^{(n)}(0) = \Phi^{(n)}(0)$$

where $\Phi^{(n)}(x)$ denotes the nth derivative of $\Phi(x)$. The cumulant function is the natural logarithm of the characteristic function

$$\Psi(i\omega) = \psi(\omega) = \ln \phi(\omega) = \sum_{n=0}^{\infty} \kappa_n \frac{(i\omega)^n}{n!}$$

where κ_n is the nth cumulant. $\kappa_1 = m$, $\kappa_2 = \sigma^2$, $\kappa_3 = \mu_3$, and $\kappa_4 = \mu_4 - 3\mu_2^2$. The coefficient of skewness is $\gamma_1 = \kappa_3/(\kappa_2)^{3/2} = \mu_3/\sigma^3$ and the coefficient of excess (Kurtosis) is $\gamma_2 = \kappa_4/(\kappa_2)^2 = \mu_4/\sigma^4 - 3$.

In general, for j = 1, 2, . . . $\gamma_j = \kappa_{j+2}(\kappa_2)^{j/2+1}$. The central moments are related to the moments about the origin by

$$\mu_n = \sum_{j=0}^{n} \binom{n}{j} (-1)^j \mu'_{n-j} m^j$$

and conversely

$$\mu'_n = \sum_{j=0}^{n} \binom{n}{j} \mu_{n-j} m^j$$

The remainder of this chapter provides data on 34 of the MATHLIB probability distributions. They are listed in alphabetic order.

BETA DISTRIBUTION

The beta distribution is defined by

$$P(x|a, b) = \frac{1}{B(a, b)} \int_0^x t^{a-1}(1 - t)^{b-1} dt \qquad a, b > 0, \quad 0 \le x \le 1$$

with mean a/(a + b), variance ab/[(a + b)^2 (a + b + 1)], skewness 2(a − b)/(a + b + 2), excess

$$\sqrt{\frac{a + b + 1}{ab}} \left[\frac{3(a + b + 1)[2(a + b)^2 + ab(a + b - 6)]}{ab(a + b + 2)(a + b + 3)} - 3 \right]$$

and the nth moment about the origin $\mu'_n = B(a + n, b)/B(a, b)$. The characteristic function is M(a, a + b, iω).

BINOMIAL DISTRIBUTION

The binomial distribution is defined by

$$\sum_{s=0}^{n} \binom{N}{s} p^s (1 - p)^{N-s} \qquad 0 \le n \le N, \quad 0 < p < 1$$

where n and N are non-negative integers and let q = 1 − p. It has mean Np, variance Npq, skewness (q − p)/$\sqrt{}$(Npq), excess (1 − 6pq)/(Npq), and cumulants κ_1 = Np, and κ_{n+1} = pq $d\kappa_n/(dp)$ for n = 1, 2, ... The characteristic function is $(q + pe^{i\omega})^N$.

BIVARIATE NORMAL DISTRIBUTION

Command **BIVN** evaluates the bivariate normal distribution given by

$$F(x, y) = \int_{-\infty}^{x}\int_{-\infty}^{y} f(\alpha, \beta)\, d\alpha d\beta$$

where f(x, y) is the bivariate normal probability density function

$$f(x, y) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1 - r^2}}\, e^{-\frac{1}{2(1-r^2)}\left(\frac{(x-\mu_1)^2}{\sigma_1^2} - \frac{2r(x-\mu_1)(y-\mu_2)}{\sigma_1\sigma_2} + \frac{(y-\mu_2)^2}{\sigma_2^2}\right)}$$

with means μ_1 and μ_2, variances σ_1^2 and σ_2^2, and correlation coefficient r. All the cumulants of order 3 and higher are zero. The characteristic function is

$$\phi(\omega_1, \omega_2) = e^{-(\sigma_1^2\omega_1^2 + r\sigma_1\sigma_2\omega_1\omega_2 + \sigma_2^2\omega_2^2)} e^{i(\mu_1\omega_1 + \mu_2\omega_2)}$$

See the related distributions discussed in Chapter 6.

BOSE-EINSTEIN DISTRIBUTION

The Bose–Einstein distribution is defined by

$$P(x|\alpha, \beta) = \int_0^x \frac{-\alpha\beta\, dt}{\ln(1 - \beta)[\exp(\alpha t) - \beta]} \qquad x \geq 0$$

with mean $m = -\mathfrak{L}_2(\beta)/[\alpha \ln(1 - \beta)]$ and variance $-2\mathfrak{L}_3(\beta)/[\alpha^2 \ln(1 - \beta)] - m^2$. In general, $\mu'_n = -n!\mathfrak{L}_{n+1}(\beta)/[\alpha^n \ln(1 - \beta)]$. $\mathfrak{L}_n(z)$ is the polylogarithm function discussed in Chapter 4. See the program BOSEE in MAN5 for the closed form distribution function.

CAUCHY DISTRIBUTION

The Cauchy distribution is defined by

$$P(x|\theta, \alpha) = \int_{-\infty}^{x} \frac{dt}{\pi\alpha[1 + (t - \theta)^2/\alpha^2]} \qquad \alpha > 0$$

with no moments or cumulants defined. The characteristic function is $e^{i\theta\omega - \alpha|\omega|}$.

CHI-SQUARE DISTRIBUTION

The chi-square distribution is defined by

$$P(\chi^2|\nu) = \left[2^{\nu/2}\Gamma(\nu/2)\right]^{-1} \int_0^{\chi^2} t^{\nu/2-1} e^{-t/2}\, dt \qquad \nu > 0,\ \chi^2 \geq 0$$

with mean ν, variance 2ν, skewness $2^{3/2}\nu^{-1/2}$, excess $12\nu^{-1}$, and cumulants $\kappa_{n+1} = 2^n n!\nu$ for $n = 0, 1, \ldots$. The characteristic function is $(1 - 2i\omega)^{-\nu/2}$.

DIRAC DELTA DISTRIBUTION

The Dirac delta function can be considered a probability distribution with zero mean and zero variance. As discussed in Appendix F of the MATHLIB manual, the Dirac delta can be defined as the limit in the mean of a zero mean normal distribution as $\sigma \to 0$.

EXPONENTIAL DISTRIBUTION

The exponential distribution is defined by

$$P(x|\mu, \sigma) = \int_{\mu-\sigma}^{x} \sigma^{-1} e^{-(t-\mu+\sigma)/\sigma}\, dt \qquad x \geq \mu - \sigma,\ \sigma > 0$$

with mean μ, variance σ^2, skewness 2, excess 6, and cumulants $\kappa_1 = \mu$ and $\kappa_n = (n-1)!\sigma^n$ for $n = 2, 3, \ldots$. The characteristic function is given by $e^{i\omega(\mu-\sigma)}(1 - i\sigma\omega)^{-1}$.

EXTREME VALUE DISTRIBUTION

The Type I extreme value distribution is defined by

$$P(x|\xi, \theta) = \theta^{-1} \int_{-\infty}^{x} e^{-(t-\xi)/\theta} \exp[-e^{-(t-\xi)/\theta}]\, dt \qquad \theta > 0$$

with mean $\xi - \theta\psi(1) = \xi + \gamma\theta$, variance $\pi^2\theta^2/6$, skewness $-6^{3/2}\psi^{(2)}(1)\pi^{-3}$, excess $36\psi^{(3)}(1)\pi^{-4} = 2.4$, and cumulants $\kappa_n = (-\theta)^n \psi^{(n-1)}(1)$ for $n = 1, 2, \ldots$. The characteristic function is given by $e^{i\omega\xi}\Gamma(1 - i\omega\theta)$.

F DISTRIBUTION

The F distribution is defined for v_N, $v_D > 0$ by

$$P(F|v_N, v_D) = \frac{v_N^{v_N/2}\, v_D^{v_D/2}}{B\left(\frac{1}{2}v_N, \frac{1}{2}v_D\right)} \int_0^F t^{(v_N-2)/2} (v_D + v_N t)^{-(v_N+v_D)/2}\, dt \qquad F \geq 0$$

with mean $v_D/(v_D - 2)$ for $v_D > 2$ and variance

$$\sigma^2 = \frac{2v_D^2(v_N + v_D - 2)}{v_N(v_D - 2)^2(v_D - 4)} \qquad v_D > 4$$

The characteristic function is $\phi(\omega) = E\{e^{i\omega F}\} = M(v_N/2, -v_D/2, -i\omega v_D/v_N)$.

FERMI–DIRAC DISTRIBUTION

The Fermi–Dirac distribution is defined by

$$P(x|\alpha, \beta) = \int_0^x \frac{\alpha\beta\, dt}{\ln(1 + \beta)[\exp(\alpha t) + \beta]} \qquad x \geq 0$$

with mean $m = -\mathfrak{L}_2(-\beta)/[\alpha \ln(1 + \beta)]$ and variance $-2\mathfrak{L}_3(-\beta)/[\alpha^2 \ln(1 + \beta)] - m^2$. In general, $\mu'_n = -n!\mathfrak{L}_{n+1}(-\beta)/[\alpha^n \ln(1 - \beta)]$. $\mathfrak{L}_n(z)$ is the polylogarithm function discussed in Chapter 4. See the program FRMID in MAN5 for the closed form distribution function.

GAMMA DISTRIBUTION

The gamma distribution is defined by

$$P(x|\nu) = [\Gamma(a)]^{-1} \int_0^x t^{a-1} e^{-t}\, dt \qquad a > 0, \quad x \geq 0$$

with mean a, variance a, skewness $2a^{-1/2}$, excess 6/a, and cumulants $\kappa_n = (n - 1)!a$ for $n = 1, 2, \ldots$. The characteristic function is $(1 - i\omega)^{-a}$.

GEOMETRIC DISTRIBUTION

The geometric distribution, evaluated by GEOM in MAN5, is defined by

$$\sum_{n=0}^{x} p(1 - p)^n \qquad 0 < p < 1 \quad x = 0, 1, \ldots$$

with mean $(1 - p)/p$, variance $(1 - p)p^{-2}$, skewness $(2 - p)[1 - p]^{-1/2}$, and excess $6 + p^2/(1 - p)$. The characteristic function is $p[1 - (1 - p)e^{i\omega}]^{-1}$.

HYPERGEOMETRIC DISTRIBUTION

The hypergeometric distribution is defined by

$$\sum_{k=0}^{c} \frac{\binom{x}{k}\binom{N-x}{n-k}}{\binom{N}{n}} \qquad 0 \le c \le \text{MIN}(x,n) \qquad n + x \le N \qquad n, x, N, c \in \mathbf{N}$$

with mean np, variance npq(N – n)/(N – 1), and skewness

$$\frac{q - p}{\sqrt{npq}}\left(\frac{N - 1}{N - n}\right)^{1/2}\left(\frac{N - 2n}{N - 2}\right)$$

where p = x/N and q = 1 – p = (N – x)/N. The characteristic function is

$$\frac{\binom{N - x}{n}}{\binom{N}{n}} F(-n, -x, N - x - n + 1, e^{i\omega})$$

KOLMOGOROV–SMIRNOV DISTRIBUTION

The Kolmogorov–Smirnov distribution is defined by

$$1 - 2\sum_{n=1}^{\infty}(-1)^{n-1} e^{-2n^2\lambda^2} \qquad \lambda \ge 0$$

with the corresponding probability density function

$$8\lambda\sum_{n=1}^{\infty}(-1)^{n-1} e^{-2n^2\lambda^2} \qquad \lambda \ge 0$$

The mean is $\sqrt{(\pi/2)}\ln 2$ and the variance is $\pi^2/8 - \pi[\ln 2]^2/2$. The nth moment about zero is

$$\mu'_n = \frac{\Gamma(n/2 + 1)}{2^{n/2-1}}\sum_{m=1}^{\infty}(-1)^{m-1} m^{-n}$$

This is the asymptotic distribution. The exact distribution for a small number of observations is discussed in Chapter 6.

LAPLACE DISTRIBUTION

The Laplace distribution is defined by

$$P(x|\alpha, \beta) = \int_{-\infty}^{x} [2\beta]^{-1} e^{-|t-\alpha|/\beta} dt$$

with mean α, variance $2\beta^2$, skewness 0, excess 3, and cumulants $\kappa_1 = \alpha$, $\kappa_2 = 2\beta^2$, $\kappa_{2n+1} = 0$, and $\kappa_{2n} = (2n)!\beta^{2n}/n$ for $n = 1, 2, \ldots$. The characteristic function is $e^{\alpha i\omega}(1 + \beta^2\omega^2)^{-1}$.

LOG-NORMAL DISTRIBUTION

The log-normal distribution is defined by

$$P(x|\theta, \sigma) = \int_{\theta}^{x} [(t - \theta)\sqrt{2\pi}\,\sigma]^{-1} \exp[-\tfrac{1}{2}\{\ln(t - \theta) - \zeta\}^2/\sigma^2] dt$$

with mean $\exp(\zeta + \sigma^2/2)$ and variance $\alpha(\alpha - 1)e^{2\zeta}$ where $\alpha = \exp(\sigma^2)$. In general, $\mu'_n = \exp(n\zeta + n^2\sigma^2/2)$. See the program LOGN in MAN5 for the distribution function.

MARCUM P_N DISTRIBUTION

The Marcum P_N distribution is defined for $x \geq 0$ by

$$\int_0^x \left(\frac{t}{NR}\right)^{(N-1)/2} e^{-(t+NR)} I_{N-1}(\sqrt{4NRt})\, dt \qquad R \geq 0 \quad N = 1, 2, \ldots$$

with mean $N(R + 1)$, variance $N(2R + 1)$, skewness $2N(3R + 1)[N(2R + 1)]^{-3/2}$, excess $6(4R + 1)[N(2R + 1)^2]^{-1}$, and cumulants $\kappa_n = N(n - 1)!(nR + 1)$ for $n = 2, 3, \ldots$. The characteristic function is

$$\frac{e^{-NR}\, e^{NR/(1 + i\omega)}}{(1 + i\omega)^N}$$

This distribution is related to the non-central χ^2 distribution by the identity $MPN(N, R, x) = U\tau\eta C(2NR, 2N, 2x)$ within numerical error.

MARCUM Γ_N DISTRIBUTION

This is the special case of the Marcum P_N distribution with $R = 0$.

$$\int_0^x \frac{t^{N-1} e^{-t}}{(N - 1)!}\, dt \qquad x \geq 0, \quad N = 1, 2, \ldots$$

MAXWELL DISTRIBUTION

The Maxwell distribution is defined by

$$P(x|\alpha) = \int_0^x 4\sqrt{\alpha^3/\pi}\ t^2\ e^{-\alpha t^2}\ dt \qquad \alpha > 0, \quad x \geq 0$$

with mean $2[\pi\alpha]^{-1/2}$, and variance $3/(2\alpha) - 4/(\pi\alpha)$. In general, $\mu'_{2n-1} = 2n!\alpha^{-n+1/2}/\sqrt{\pi}$ for n = 1, 2, . . ., and $\mu'_{2n-2} = 2\alpha(2n-1)!!(2\alpha)^{-n}$ for n = 2, 3, ... See the program MAXW in MAN5 for the distribution function.

NEGATIVE BINOMIAL DISTRIBUTION

The negative binomial distribution is defined for n, $x \in \mathbf{N}$ by

$$\sum_{k=x}^{\infty}\binom{n+k-1}{k} p^k q^n \qquad q = \frac{1}{Q} \qquad p = \frac{P}{Q} = 1 - q \qquad 0 < p < 1$$

with mean nP, variance nPQ, skewness $(P + Q)[nPQ]^{-1/2}$, excess $(1 + 6PQ)[nPQ]^{-1}$. The characteristic function is $(Q - Pe^{i\omega})^{-1}$.

NORMAL DISTRIBUTION (UNIVARIATE)

The univariate normal (Gaussian) distribution is defined by

$$P(x|\mu, \sigma) = \int_{-\infty}^{x} \frac{1}{\sigma\sqrt{2\pi}} e^{-(t-\mu)^2/(2\sigma^2)}\ dt \qquad \sigma > 0$$

with mean μ, variance σ^2, skewness 0, excess 0, and cumulants $\kappa_1 = \mu$, $\kappa_2 = \sigma^2$, and $\kappa_n = 0$ for $n > 0$. The characteristic function is

$$e^{-\sigma^2\omega^2/2}\ e^{i\mu\omega}$$

From the relations for conditional probability density functions,

$$f(y|x) = \frac{f(x, y)}{f(x)}$$

so if f(x, y) is the bivariate normal density discussed on page 22, then the conditional density of y given x is also normal

$$f(y|x) = \frac{e^{-\frac{[y-\mu_2-r\sigma_2(x-\mu_1)/\sigma_1]^2}{2\sigma_2^2(1-r^2)}}}{\sigma_2\sqrt{2\pi(1-r^2)}}$$

NON-CENTRAL F DISTRIBUTION

The non-central F distribution is defined for $F \geq 0$ by

$$P(F|v_N, v_D, \lambda) = \int_0^F f(t|v_N, v_D, \lambda)\, dt \qquad v_N, v_D > 0, \quad \lambda \geq 0$$

where the probability density function $f(t|v_N, v_D, \lambda)$ is

$$\sum_{n=0}^{\infty} e^{-\lambda/2} \frac{(\lambda/2)^n}{n!} \frac{(v_N+2n)^{(v_N+2n)/2}}{B[(v_N+2n)/2,\ v_D/2]} t^{(v_N+2n-2)/2} [v_D+(v_N+2n)t]^{-(v_N+2n+v_D)/2}$$

with mean $v_D(v_N + \lambda)[v_N(v_D - 2)]^{-1}$ and variance

$$2\left(\frac{v_D}{v_N}\right)^2 \frac{(v_N + \lambda)^2 + (v_N + 2\lambda)(v_D - 2)}{(v_D - 2)^2(v_D - 4)}$$

The characteristic function is

$$\sum_{n=0}^{\infty} \left[\frac{(\lambda/2)^n}{n!}\right] e^{-\lambda/2}\, M(v_N/2 + n,\ -v_D/2,\ -i\omega v_D/v_N)$$

NON-CENTRAL T DISTRIBUTION

The non-central T distribution is defined for $v > 0$ and δ, $t \geq 0$ by

$$P(t|v, \delta) = \frac{1}{\sqrt{v}\, B(\tfrac{1}{2}, v/2)} \int_{-\infty}^{t} \left(\frac{v}{v+x^2}\right)^{(v+1)/2} \exp\left(\frac{-v\delta^2}{2(v+x^2)}\right) Hh_v\left(\frac{-\delta x}{\sqrt{v+x^2}}\right) dx$$

where the function $Hh_n(z\sqrt{2}) = (2^{n-1}\pi)^{1/2}\, i^n \operatorname{erfc} z$. It has the mean $m = (v/2)^{1/2}\, \delta\, \Gamma(\tfrac{1}{2}(v-1))/\Gamma(v/2)$ and the variance $v(1 + \delta^2)/(v - 2) - m^2$.

NON-CENTRAL χ^2 DISTRIBUTION

The non-central chi-square distribution is defined for $\chi^2 \geq 0$ by

$$P(\chi^2|v, \lambda) = \int_0^{\chi^2} \frac{1}{2}\left(\frac{t}{\lambda}\right)^{(v-2)/4} e^{-(t+\lambda)/2}\, I_{(v-2)/2}\left(\sqrt{\lambda t}\right) dt \qquad v > 0, \quad \lambda \geq 0$$

with mean $v + \lambda$, variance $2(v + 2\lambda)$, skewness $2\sqrt{2}(v + 3\lambda)[v + 2\lambda]^{-3/2}$, excess $12(v + 4\lambda)[v + 2\lambda]^{-2}$, and cumulants $\kappa_n = 2^{n-1}(n - 1)!(v + n\lambda)$. The characteristic function is $(1 - 2i\omega)^{-v/2}\exp[i\omega\lambda(1 - 2i\omega)^{-1}]$.

PARETO DISTRIBUTION

The Pareto distribution is defined by $1 - (k/x)^a$ for $k > 0$, $a > 0$, $x \geq k$ with probability density function ak^a/x^{a+1}, mean $ak/(a - 1)$, variance $ak^2(a - 1)^{-2}(a - 2)^{-1}$ for $a > 2$, and $\mu'_n = ak^n/(a - n)$. See PRETO.

POISSON DISTRIBUTION

The Poisson distribution is defined by

$$\sum_{n=0}^{x} e^{-m} \frac{m^n}{n!} \qquad m > 0, \quad x = 0, 1, \ldots$$

with mean m, variance m, skewness $m^{-1/2}$, excess m^{-1}, and cumulants $\kappa_n = m$ for $n = 1, 2, \ldots$. The characteristic function is $\exp[m(e^{i\omega} - 1)]$.

RAYLEIGH DISTRIBUTION

This is the special case of the Marcum P_N distribution with $R = 0$ and $N = 1$. It is also the Marcum Γ_1 distribution.

RICIAN DISTRIBUTION

This is the special case of the Marcum P_N distribution with $N = 1$.

SECH-SQUARED DISTRIBUTION

The sech–squared (logistic) distribution is defined for $x \geq 0$ by

$$P(x|\mu, \alpha) = \int_{-\infty}^{x} (\alpha/2)\operatorname{sech}^2[\alpha(t - \mu)]\, dt \qquad \alpha > 0$$

with mean μ, variance π^2 /(12α^2), and skewness 0. The characteristic function for $y = 2\alpha(x - \mu)$ is $\pi\omega \operatorname{csch} \pi\omega$ so with $x = y/(2\alpha) + \mu$, the characteristic function of x is

$$\phi(\omega) = \frac{\pi\omega}{2\alpha} \operatorname{csch}\left(\frac{\pi\omega}{2\alpha}\right) e^{i\omega\mu}$$

The odd cumulants of y are all zero and for n even, the cumulants of y $\kappa_n(y) = 6(2^2 - 1)B_n$. Thus, the odd cumulants of x are zero and the even ones are $\kappa_n(x) = \kappa_n(y)[2\alpha]^{-n}$. See the program SECHS in MAN5 for the distribution function.

T DISTRIBUTION

The T distribution is defined by

$$A(t|v) = \left[\sqrt{v}\ B(\tfrac{1}{2}, v/2)\right]^{-1} \int_{-t}^{t} \left(1 + \frac{x^2}{v}\right)^{-(v+1)/2} dx \qquad t \geq 0 \quad v > 0$$

with mean 0, variance $v/(v-2)$ for $v > 2$, skewness 0, and excess $6/(v-4)$ for $v > 4$. The characteristic function is

$$\phi(\omega) = E\left\{\exp\left(i\omega \frac{X}{\sqrt{\chi^2/v}}\right)\right\} = \left(\frac{|\omega|}{2\sqrt{v}}\right)^{v/2} [\pi\Gamma(v/2)]^{-1}\ Y_{v/2}\left(\frac{|\omega|}{\sqrt{v}}\right)$$

UNIFORM DISTRIBUTION

The uniform distribution is defined by

$$P(x|\mu, h) = \int_{\mu - h/2}^{x} h^{-1}\, dt \qquad \mu - h/2 \leq x \leq \mu + h/2$$

with mean μ, variance $h^2/12$, skewness 0, excess -1.2, cumulants $\kappa_{2n+1} = 0$ for $n = 1, 2, \ldots$, and $\kappa_{2n} = h^{2n}B_{2n}/(2n)$ where B_r is the rth Bernoulli number. The characteristic function is $2\sin(h\omega/2)e^{i\mu\omega}/(h\omega)$.

WALD DISTRIBUTION

The Wald (inverse Gaussian) distribution is defined by

$$P(x|\mu, \lambda) = \int_0^x \sqrt{\lambda/(2\pi t^3}\ e^{-\lambda(t-\mu)^2/(2\mu^2 t)}\, dt \qquad \mu, \lambda > 0, \qquad x \geq 0$$

with mean μ, variance μ^3/λ, skewness $3\sqrt{(\mu/\lambda)}$, excess $15\mu/\lambda$, and cumulants $\kappa_n = (2n-3)!!\mu^{2n-1}\lambda^{1-n}$ for $n = 2, 3, \ldots$. See the program WALD in MAN5 for the distribution function.

WEIBULL DISTRIBUTION

The Weibull distribution is defined by

$$P(x|c) = \int_0^x ct^{c-1}\exp(-t^c)\, dt \qquad c > 0, \quad x \geq 0$$

with mean $m = \Gamma(c^{-1} + 1)$ and variance $\Gamma(2c^{-1} + 1) - m^2$.

6

ADVANCED PROBABILITY DISTRIBUTIONS

INTRODUCTION

There are a number of useful probability distributions which are generally avoided in textbooks because they are too hard to compute. An example is the exact distribution of the correlation coefficient which I avoided discussing on pages 281 and 282 of the MATHLIB manual. There are many important probabilities used in engineering and statistics which involve advanced functions, but nevertheless are easy to evaluate with MATHLIB, given an understanding of the higher transcendental functions. Some of these are covered in this chapter.

KOLMOGOROV-SMIRNOV ONE-SAMPLE STATISTIC

Probability command **UTKS** computes the upper tail of the asymptotic Kolmogorov–Smirnov distribution. This is generally used when there are over 35 observations. For smaller numbers of observations, the exact finite sample size distribution is preferred.

Define the Kolmogorov–Smirnov one-sample statistic

$$D_N = \sup_x |S_N(x) - F_x(x)|$$

where $S_N(x)$ is the empirical distribution function, $F_x(x)$ is the true distribution function, and sup is the supremum (least upper bound or maximum). Also define the one-sided Kolmogorov–Smirnov statistics

$$D_N^+ = \sup_x [S_N(x) - F_x(x)] \qquad D_N^- = \sup_x [F_x(x) - S_N(x)]$$

The statistics D_N, D_N^+, and D_N^- are independent of the continuous cumulative distribution $F_x(x) = \text{Prob}\{X \le x\}$. Let the N observations be $X_1 \le X_2 \le \ldots \le X_N$ and define the empirical distribution $S_N(x)$ by

$$S_N(x) = \begin{cases} 0 & \text{for } x < X_1 \\ \dfrac{n}{N} & \text{for } X_n \le x < X_{n+1} \qquad n = 1, 2, \ldots N-1 \\ 1 & \text{for } X_N \le x \end{cases}$$

Commands **HIST** and **CUMΣ** can be used to create the $S_N(x)$ vector. For large N, the distribution of D_N is given by

$$\lim_{N\to\infty} \text{Prob}\left\{D_N < \frac{z}{\sqrt{N}}\right\} = 1 - 2\sum_{m=1}^{\infty} (-1)^{m-1} e^{-2m^2z^2} = L(z)$$

and $1 - L(z)$ is evaluated by command **UTKS**. For $c = 0, 1, 2, \ldots N$ the exact distribution is given by

$$\text{Prob}\left\{D_N < \frac{c}{N}\right\} = \frac{N!}{N^N} e^N R_{0,N}(c)$$

where $R_{j,k}(c)$ is defined for all integers j, all non-negative integers k, $c = 1, 2, \ldots N$, and $R_{j,k}(c)$ satisfies the recurrence formula

$$R_{j,k+1}(c) = e^{-1} \sum_{s=0}^{2c-1} R_{j+1-s,k}(c) \frac{1}{s!} \qquad \text{for } |j| \le c - 1$$

with the initial values of $R_{0,0}(c) = 1$, $R_{j,0}(c) = 0$ for $j \ne 0$, and $R_{j,k}(c) = 0$ for $|j| \ge c$. Program KS1S in the MAN5 directory implements these equations which are Kolmogorov's original equations. Program KS1S reproduces the probability tables in Birnbaum's paper.

Birnbaum's and Tingey's equations reproduce the numbers in L. Miller's tables for the D_N^+ statistic. The upper tail distribution is

$$\text{Prob}\left\{D_N^+ \ge \frac{c}{N}\right\} = \frac{c}{N^N} \sum_{s=0}^{IP[N-c]} \binom{N}{s} (N - c - s)^{N-s}(c + s)^{s-1}$$

which is evaluated by program KS1P in the MAN5 directory. The corresponding asymptotic distribution for D_N^+ is

$$\lim_{N\to\infty} \text{Prob}\left\{D_N^+ < \frac{z}{\sqrt{N}}\right\} = 1 - e^{-2z^2}$$

D_N^+ and D_N^- have identical distributions because of symmetry.

NON-CENTRAL BETA DISTRIBUTION

Let x_1^2 be non-central χ^2 with 2a degrees of freedom and non-centrality parameter λ, and let x_2^2 be central χ^2 with 2b degrees of freedom. Define the statistic $x = x_1^2 [x_1^2 + x_2^2]^{-1}$. Then x has the non-central beta density function defined by Rao as

$$f(x|a, b, \lambda) = e^{-\lambda^2/2} M(a + b, a, \lambda^2 x/2) [B(a, b)]^{-1} x^{a-1} (1 - x)^{b-1}$$

where M(a, b, z) is Kummer's confluent hypergeometric function, B(a, b) is the beta function, and λ is the non-centrality parameter. As formidable as it may seem to integrate this density to obtain the cumulative distribution function, there are a number of practical applications wherein the evaluation is surprisingly easy. Suppose b is an integer. By Kummer's transformation

$$M(a + b, a, \lambda^2 x/2) = e^{\lambda^2 x/2} M(-b, a, -\lambda^2 x/2)$$

This reduces the confluent hypergeometric function to a polynomial and **MABZ** will evaluate it very fast. Thus, the distribution function can quickly be evaluated as a numerical integral.

JOINT DISTRIBUTIONS OF REAL RANDOM VARIABLES

Omura and Kailath have compiled distribution, density, and characteristic functions for distributions related to normal variables. Included in their report are the RATIOS: Gaussian/Gaussian, Gaussian/Rayleigh, Gaussian/Rice, Rayleigh/Rayleigh, Rice/Rayleigh, and Rice/Rice; SUMS: central χ^2 + central χ^2, central χ^2 + non-central χ^2, and non-central χ^2 + non-central χ^2; DIFFERENCES: central χ^2 – central χ^2, non-central χ^2 – central χ^2, and non-central χ^2 – non-central χ^2; PRODUCTS: Gaussian × Gaussian, Rayleigh × Rayleigh, Rayleigh × Rice, and Rice × Rice. The Gaussian distribution is the same as the normal distribution. Some of these results are based on K. Miller, *Multidimensional Gaussian Distributions.*

JOINT DISTRIBUTIONS OF COMPLEX RANDOM VARIABLES

K. Miller has derived a useful collection of complex distribution functions for hypothesis testing of complex random variables. We will now focus on a few of his results and evaluation techniques for them.

Let $\mathbf{z}_1, \mathbf{z}_2, \ldots \mathbf{z}_N$ be two dimensional independent complex random vectors with zero mean and positive definite covariance matrix

$$\Sigma^{-1} = \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma^*_{12} & \sigma_{22} \end{bmatrix} \qquad \sigma_{12} = \sigma_1 + i\sigma_2$$

where * denotes complex conjugation. Let $\mathbf{z}_n^T = [u_n \quad v_n]$ for n = 1, 2, . . . N and define the second order statistics

$$w_1 = \sum_{n=1}^{N} |u_n|^2 \qquad w_2 = \sum_{n=1}^{N} |v_n|^2$$

$$r = \sum_{n=1}^{N} u_n v_n^* = r_1 + ir_2$$

Then the joint probability density of these statistics is

$$f(w_1, w_2, r_1, r_2) = \frac{(w_1 w_2 - |r|^2)^{N-2}}{\pi(N-1)!(N-2)!(\text{DET } \Sigma)^N} e^{-2[\sigma_1 r_1 + \sigma_2 r_2 + (\sigma_{11} w_1 + \sigma_{22} w_2)/2]}$$

for $w_1, w_2 > 0$, $w_1 w_2 > |r|^2$, $-\infty < r_1, r_2 < \infty$ and 0 otherwise. Defining

$$\Omega = \begin{bmatrix} w_1 & r \\ r^* & w_2 \end{bmatrix} = \sum_{n=1}^{N} z_n z_n^H$$

where H denotes Hermitian transpose, this can be written as the bivariate complex Wishart density function

$$f(\Omega) = \frac{(\text{DET } \Omega)^{N-2}}{\pi(N-1)!(N-2)!(\text{DET } \Sigma)^N} e^{-\text{Tr } \Omega R^{-1}}$$

where DET denotes determinant and Tr denotes trace. The joint density of w_1 and w_2 is then

$$f(w_1, w_2) = \frac{(w_1 w_2)^{(N-1)/2}}{(N-1)!(\text{DET } \Sigma)^N |r_{12}|^{N-1}} e^{-(\sigma_{11} w_1 + \sigma_{22} w_2)} I_{N-1}\left(2|\sigma_{12}|\sqrt{w_1 w_2}\right)$$

for $w_1, w_2 > 0$ and 0 otherwise.

Consider next the normalized correlation functions $\rho_1 = r_1/w$ and $\rho_2 = r_2/w$ where $w = (w_1 + w_2)/2 = \frac{1}{2}\mathbf{z}^H\mathbf{z}$. Their joint density function is given by $f(\rho_1, \rho_2)$ which equals

$$\frac{(2N-1)2^{2N-1}[1 - (\rho_1^2 + \rho_2^2)]^{(2N-3)/2}}{\pi\alpha^{2N}(\text{DET } \Sigma)^N} F\left(N, N + \tfrac{1}{2}, N - \tfrac{1}{2}, \frac{\beta^2}{\alpha^2}\right)$$

for $\rho_1^2 + \rho_2^2 < 1$ and 0 otherwise. F(a, b, c, z) is the Gaussian hypergeometric function and parameters α and β equal

$$\alpha = 2[\sigma_1\rho_1 + \sigma_2\rho_2 + (\sigma_{11} + \sigma_{22})/2]$$

$$\beta = |\sigma_{11} - \sigma_{22}|\sqrt{1 - (\rho_1^2 + \rho_2^2)}$$

Finally, consider the modulus of the sample correlation coefficient

$$\rho = \frac{r}{\sqrt{w_1 w_2}}$$

Then the density function of $|\rho|$ is given by

$$f(|\rho|) = 2(N - 1)(1 - |\lambda|^2)^N |\rho|(1 - |\rho|^2)^{N-2} F(N, N, 1, |\lambda|^2|\rho|^2)$$

where the complex correlation coefficient λ is defined by

$$\lambda = -\frac{\sigma_{12}}{\sqrt{\sigma_{11}\sigma_{22}}}$$

With $c = |\rho|^2$, the density of c is

$$f(c) = (N - 1)(1 - |\lambda|^2)^N(1 - c)^{N-2} F(N, N, 1, |\lambda|^2 c)$$

This density and f(C) on the next page can be used for exact small sample tests with **LCNT**. See page 281 of the MATHLIB manual.

EVALUATION TECHNIQUES

The difficult part of integrating the above three density functions is the Gaussian hypergeometric function. Using the first linear transformation formula given on page 131 of the MATHLIB manual we have the equalities

$$F(N, N + \tfrac{1}{2}, N - \tfrac{1}{2}, z) = (1 - z)^{-N-1} F(-\tfrac{1}{2}, -1, N - \tfrac{1}{2}, z)$$

$$F(N, N, 1, z) = (1 - z)^{1-2N} F(1 - N, 1 - N, 1, z)$$

Now the $b = -1$ in the upper equation and the $a = b = 1 - N$ in the lower equation reduces F(a, b, c, z) to a polynomial. By evaluating the distribution function with the right side of these equations, the hypergeometric function command **F2F1** simply evaluates a polynomial and thus numerical integration is quick. See program FOFC in the MAN5 directory to evaluate the distribution function of $c = |\rho|^2$.

JOINT FIRST AND SECOND ORDER COMPLEX STATISTICS

K. Miller generalizes the above density functions to include the cases where the means are not zero. Define the complex first order statistics and the complex second order statistics

$$X = \frac{1}{N}\sum_{n=1}^{N} u_n = X_1 + iX_2$$

$$Y = \frac{1}{N}\sum_{n=1}^{N} v_n = Y_1 + iY_2$$

$$W_1 = \frac{1}{N}\sum_{n=1}^{N} |u_n - X|^2 = \frac{w_1}{N} - |X|^2$$

$$W_2 = \frac{1}{N}\sum_{n=1}^{N} |v_n - Y|^2 = \frac{w_2}{N} - |Y|^2$$

$$P = \frac{1}{N}\sum_{n=1}^{N} (u_n - X)(v_n^* - Y^*) = P_1 + iP_2 = \frac{\rho}{N} - XY^*$$

The joint density of W_1 and W_2, $f(W_1, W_2)$, is then

$$\frac{N^N (W_1 W_2)^{(N-2)/2}}{(N-2)!(\text{DET } \Sigma)^{N-1} |r_{12}|^{N-2}} e^{-N(\sigma_{11}W_1 + \sigma_{22}W_2)} I_{N-2}\left(2N|\sigma_{12}|\sqrt{W_1 W_2}\right)$$

Let $P_1 = R_1/W$ and $P_2 = R_2/W$ where $W = (W_1 + W_2)/2$. The joint density function $f(P_1, P_2)$ is identical to $f(\rho_1, \rho_2)$ with N replaced by N – 1. Define the modulus of the sample correlation coefficient

$$P = \frac{R}{\sqrt{W_1 W_2}} \qquad C = |P|^2$$

The density of $|P|$ and C are $f(|\rho|)$ and f(c) with N replaced by N–1.

$$f(C) = (N-2)(1 - |\lambda|^2)^{N-1}(1 - C)^{N-3} F(N-1, N-1, 1, |\lambda|^2 C)$$

See program FOFC2 in the MAN5 directory for the distribution function.

KOLMOGOROV–SMIRNOV TWO-SAMPLE STATISTIC

The Kolmogorov–Smirnov one-sample test compared an empirical distribution with the true distribution. The two-sample test compares two empirical distributions $S_M(x)$ and $S_N(x)$. Define the two-sample Kolmogorov–Smirnov statistic

$$D_{MN} = \sup_x |S_M(x) - S_N(x)|$$

Let the first M observations be $X_1 \le X_2 \le \ldots \le X_M$ and define the empirical distribution $S_M(x)$ by

$$S_M(x) = \begin{cases} 0 & \text{for } x < X_1 \\ \dfrac{m}{M} & \text{for } X_m \le x < X_{m+1} \qquad m = 1, 2, \ldots M - 1 \\ 1 & \text{for } X_M \le x \end{cases}$$

Let the second N observations be $Y_1 \le Y_2 \le \ldots \le Y_N$ and define the empirical distribution $S_N(x)$ by

$$S_N(x) = \begin{cases} 0 & \text{for } x < Y_1 \\ \dfrac{n}{N} & \text{for } Y_n \le x < Y_{n+1} \qquad n = 1, 2, \ldots N - 1 \\ 1 & \text{for } Y_N \le x \end{cases}$$

Commands **HIST** and **CUMΣ** can be used to create $S_M(x)$ and $S_N(x)$ vectors.

Smirnov first introduced this statistic in 1939. It took over a decade before its distribution was found even for equal sample sizes. Command **KST2** performs the M = N asymptotic test.

$$\lim_{M,N \to \infty} P\left(\sqrt{\frac{MN}{M + N}}\ D_{MN} \le d\right) = L(d)$$

where L(d) is defined on page 32 and 1 – L(d) is evaluated by command **UTKS**. In general, the lower tail distribution can be evaluated as

$$P(D_{MN} < d) = \frac{A(M, N)}{\binom{M + N}{M}}$$

where A(M, N) is computed by the recursion

$$A(m + 1, n + 1) = A(m + 1, n) + A(m, n + 1)$$

with the boundary conditions

$$A(0, n) = A(m, 0) = 1$$

where these formulas only apply for m and n non-negative integers and

$$\frac{N}{M}(m - Md) < n < \frac{N}{M}(m + Md)$$

with strict inequality, and otherwise A(m, n) = 0. This method of

computation is due to Hodges and is explained in Gibbons. Program KS2S in MAN5 evaluates this distribution which reproduces most the numbers in Gibbon's table taken from Kim (table errors are random).

NONLINEAR CONTROL THEORY DISTRIBUTIONS

Nonlinear communication and control analysis such as Fokker–Planck solutions give rise to many rather nasty probability density functions involving various higher transcendental functions. MATHLIB has been designed to make the evaluation of such densities possible without the researcher having to make a large effort to write special software. An example is the phase error density for a phase lock loop

$$f(\phi) = \frac{\exp(\beta\phi + \alpha\cos\phi)}{4\pi^2 \exp(-\pi\beta)\,|I_{i\beta}(\alpha)|^2} \int_{\phi}^{\phi+2\pi} \exp(-\beta y - \alpha\cos y)\, dy$$

where the argument and order of the Bessel function is complex. Lindsey gives derivations and summaries of these distributions.

Birnbaum, Z., "Numerical Tabulation of the Distribution of Kolmogorov's Statistic For Finite Sample Size", *Journal of the American Statistical Assoc.*, 47, 1952, 425–41.

Birnbaum, Z., and Tingey, F., "One-sided Confidence Contours for Probability Distribution Functions", *Annals of Mathematical Statistics*, 22, 1951, 592–96.

Gibbons, J., and Chakraborti, S., *Nonparametric Statistical Inference*, New York, Marcel Dekker, Inc., 1992.

Hodges, J., "The Significance Probability of the Smirnov Two-Sample Test," *Arkivfoer Matematik, Astronomi och Fysik*, 3, 1958, 469–486.

Kim, P., and Jennrich, R., "Tables of the Exact Sampling Distribution of the Two-Sample Kolmogorov–Smirnov Criterion $D_{m,n}$ ($m \leq n$)," *Selected Tables in Mathematical Statistics*, Vol I, Providence, RI, American Mathematical Society, 1973, 79–170.

Lindsey, W., *Synchronization Systems in Communication and Control*, Englewood Cliffs, NJ, Prentice-Hall, 1972.

Miller, L., "Table of Percentage Points of Kolmogorov Statistics", *Journal of the American Statistical Assoc.*, 51, 1956, 111–121.

Miller, K., *Hypothesis Testing with Complex Distributions*, Huntington, N.Y., Krieger Publishing Co., 1980.

Miller, K., *Multidimensional Gaussian Distributions*, Wiley, 1964.

Omura, J., and Kailath, T., *Some Useful Probability Distributions*, Tech. Rpt. No. 7050–6, Stanford Electronics Lab., Stanford, Ca., 1965.

Rao, C., *Linear Statistical Inference and Its Applications*, Wiley, 1965.

7

ADVANCED GRAPHICS AND OTHER TECHNIQUES

INTRODUCTION

This chapter explains some useful techniques that can be used with MATHLIB and the HP 48 calculator. The programs discussed in this chapter are located in the MAN1 INTAG and MAN7 WAVE SOLVE subdirectories. We begin by explaining how to extend the MATHLIB plotting commands to custom plot applications. Then we cover advanced ↑**MATCH** techniques which provide the capability to perform rounding of algebraic equations on the HP 48. Finally, we discuss doing difficult symbolic computations with MATHLIB and the HP 48.

ADVANCED MATHLIB PLOTTING

The MATHLIB plot commands make plotting of functions and arrays very easy, but it is not obvious how to incorporate this capability into custom applications. MATHLIB provides the HP 48 with special compiled library functions which linearly interpolate the input array and thus behave as if a continuous function was input to the HP 48 function plot software. MATHLIB automatically stores these functions in the variable EQ in the current VAR directory. While these binary functions cannot be edited on the HP 48, they still can be used in your applications. There are three unique plot functions – the ones associated with **PLT1**, **PLT2**, and **PLT3**. **PLTC** uses the same function as **PLT2** and the plot with label commands also use the same functions. We illustrate by example how to use these functions.

Program PLOTS creates a special directory called PLOT and proceeds to create a user version of **PLT1** and **PLT3**. In particular, PT1 and PT3 create double sized 90° rotated plots which are ideal for printing large sized plots with the HP 82240B printer. The size of plot, # 131d by # 262d, is a good size for the printer. Since the plot is rotated by 90°, the # 262d can be increased within HP 48 memory limitations to very long X axis plots.

Move program PLOTS to the HOME or other directory where you want the PLOT directory created. PLOTS begins by doing simple plots to get MATHLIB to store the special plot functions into EQ. Then it moves these functions to a new variable so they are user accessible. When you run PLOTS, each time you plot the straight line, simply push ATTN to leave the graphics window and allow the program to continue running (pushing ATTN outside the graphics window stops the program and you will have to start again). When PLOTS finishes, the user plot functions PT1, PT3, and PREST will be created along with the other required variables and functions. PREST simply changes the size of the plot window back to the default size # 131d by # 64d. PLOTS can be generalized for other user plot applications.

ROUNDING ALGEBRAIC EQUATIONS

The MAN1 INTAG subdirectory has two other programs which you may find useful. TYPEA is simply an algebraic version of the HP 48 command **TYPE**. It is used by program RNDS. RNDS uses the conditional capability of the HP 48 command ↑**MATCH** to round the numerical values in equation E to N places. This technique can also be extended to lists and symbolic matrices.

ADVANCED SYMBOLIC DERIVATIVE TECHNIQUES

This section presents by example techniques for doing difficult symbolic computations without hanging the HP 48 memory management. Some techniques have already been discussed on pages 16 and 211 through 214 of the MATHLIB manual. The results of this example are required to compute rank M wavelet matrices, but here we simply focus on the evaluation problem of computing the nth symbolic derivative of $[d(z)]^{-N-1}$ for some collection of functions d(z). The first derivative is easy and equals $(-N-1)[d(z)]^{-N-2}d'(z)$. The next derivative is a little messier and equals $(N+1)(N+2)[d(z)]^{-N-3}[d^{(1)}(z)]^2 + (-N-1)[d(z)]^{-N-2}d^{(2)}(z)$.

In problems like this and the hydrodynamic flow problem discussed on page 211 of the MATHLIB manual, it is surprisingly easy to make little mistakes doing these computations by hand. It is also surprisingly easy to hang the memory manager on any symbolic computer performing computations such as the problem we are now considering. Program DERV1 in the MAN1 INTAG subdirectory creates a directory for doing these derivatives. Move this program to a convenient directory (or leave it where it is) and evaluate it. It creates a directory called DDIR which contains 5 programs. Recall DHH1 to the stack.

'-(D(0,Z)^-N(2)×D(1,Z)×N(1))'

This is a *pseudofunction* representing the first derivative of $[d(z)]^{-N-1}$ which has a number of interesting properties. The first is invariance to the **EXCO** command. You can test this by evaluating **EXCO** with this equation on the stack. This property is important because as we clean up the equation after each successive differentiation, we want to minimize the size of the expression (relative to the memory manager) and keep the expression compact for fast evaluation. Pseudofunction D(N,X) is defined by the program D(N, X)

« → N X « 'D(α,β)' α N β X 4 →LIST EVLΣ » »

which symbolically represents the Nth derivative of d(z) with respect to argument X. Its defined derivative program is derD(N, X, dN, dX).

« → N X dN dX « 'D(N+1,X)' EVAL » »

The definition of N is contained in PARTZ and has the defined derivative derN(N, dN) = 0: « → N dN « 0 » ».

Pseudofunction N(n) = N + n for some to be specified value N. This notation keeps **EXCO** from expanding and collecting terms like (N + n). Now place pseudofunction DHH1 on Level 1 of the stack. Pressing PARTZ will then create a pseudofunction representing its derivative which should be stored in DHH2. By this means successive derivatives can be evaluated. The seventh derivative will require several hours to compute and is a very large equation.

Pressing DERV2 creates the directory COPT which contains two programs and the variable N. Program COVT converts the pseudofunctions DHHk into equations for evaluation which must then be stored in COPT as Dk for k = 1, 2, . . . for the wavelet application.

SIMULATING DIFFICULT COMPUTATIONS

Consider efficiently evaluating the dilation equation

$$s(x/m^{n}) = \sum_{k=0}^{gm-1} a_k \, s(x/m^{n-1} - k)$$

given $s(x) \neq 0$ for $x = 1, 2, \ldots S$ where x, m, g, and n are integers with m, g = 2, 3, . . ., and n = 1, 2, ... For example, let m = 3, g = 2, and S = 2. Then the first set of terms correspond to n = 1 and are

$s(1/3) = a_0 \, s(1)$ $\quad$ $s(2/3) = a_0 \, s(2) + a_1 \, s(1)$
$s(4/3) = a_2 \, s(2) + a_3 \, s(1)$ $\quad$ $s(5/3) = a_3 \, s(2) + a_4 \, s(1)$
$s(7/3) = a_5 \, s(2)$ $\quad$ $s(8/3) = 0$

where we observe that s(3/3), s(6/3), and s(9/3) = 0 are given and thus not recomputed. After interleaving the values, we have s(x/3) for x = 1, 2, . . . 9 and we are ready to compute the values s(x/9) for x = 1, 2, 4, 5, 7, 8, . . . 25, and 26 where again we do not recompute existing values such as s(3/9). The first few are

$s(1/9) = a_0 \, s(1/3)$ $\quad$ $s(2/9) = a_0 \, s(2/3)$
$s(4/9) = a_0 \, s(4/3) + a_1 \, s(1/3)$ $\quad$ $s(5/9) = a_0 \, s(5/3) + a_1 \, s(2/3)$

It is clear that if one can decipher the patterns, dilation equations can be efficiently evaluated. The trick is of course to get the equations correct. Program SSUM in the MAN7 WAVE SOLVE directory *simulates* these computations using the symbolic capability of MATHLIB and the HP 48. Equation simulation is a very powerful technique for debugging difficult evaluation problems.

Given m, g, and n, SSUM first builds a symbolic coefficient list for the a_k coefficients for k = 0, 1, . . . gm – 1. Program CRN provides the value of S. Next, the symbolic functions s(x) represented by Sx are created. Now we are ready to run the simulation. Counter j starts at zero so $m^j = 1$. The terms of the dilation equation are then computed on the stack and placed in a symbolic matrix. The tricky operations involving the **STRN** command perform the interleaving operation. The program then does a **HALT** so you can examine what has been computed and see if it is correct. Press CONT and SSUM will compute the next set of terms up to the input value n. This powerful technique is useful in numerous applications.

CONTINUED FRACTION EXPANSIONS

Continued fraction expansions are useful. They look like

$$b_0 + \cfrac{a_1}{b_1 + \cfrac{a_2}{b_2 + \cfrac{a_3}{b_3 + \cfrac{a_4}{b_4 + \cdots}}}}$$

and are often written as

$$b_0 + \frac{a_1}{b_1 +} \; \frac{a_2}{b_2 +} \; \frac{a_3}{b_3 +} \; \frac{a_4}{b_4 +} \cdots$$

Program CFE in the MAN3 ALGB directory will compute a continued fraction expansion (CFE) of any rational polynomial function. The inputs are the numerator and denominator polynomial lists N and D, respectively. The expansion is computed by the sequence **PDVD SWAP PDVD SWAP** • • • until $|a_k| < 1E{-}10$. Observe that the sequence always terminates. The output is a list of polynomial lists on Level 2 corresponding to the b_k polynomials and a list of numbers corresponding the numbers a_k. For example, consider realizing an electrical network for the Laplace transform impedance function

$$Z(s) = \frac{s^4 + 10s^2 + 9}{s^3 + 4s}$$

$$\text{CFE}(\{9\ \ 0\ \ 10\ \ 0\ \ 1\},\{0\ \ 4\ \ 0\ \ 1\}) = \begin{cases} 2: & \{\{0\ \ 1\}\ \{0\ \ 1/6\} \\ & \ \ \{0\ \ 2.4\}\ \{0\ \ 5/18\}\ \{9\}\} \\ 1: & \{1\ \ 1\ \ 1\ \ 1\ \ 0\} \end{cases}$$

which represents the equation

$$Z(s) = s + \cfrac{1}{\cfrac{s}{6} + \cfrac{1}{2.4s + \cfrac{18}{5s}}}$$

Now the CFE corresponds to removing poles at infinity of first the impedance function, then the reciprocal of the remaining impedance function, ... The result is a series inductance of 1H, a shunt capacitance of 1/6F, another series inductance of 2.4 = 12/5H, and finally a shunt capacitance of 5/18F. For general circuit design, CFE must be modified so it does not divide out inverse resistances or compute negative circuit component values.

To evaluate a CFE, one implements the recurrence

$$A_n(s) = b_n(s)A_{n-1}(s) + a_nA_{n-2}(s) \qquad B_n(s) = b_n(s)B_{n-1}(s) + a_nB_{n-2}(s)$$

for n = 1, 2, . . ., N starting with the values

$$A_{-1}(s) = 1, \quad A_0(s) = b_0(s), \quad B_{-1}(s) = 0, \quad B_0(s) = 1$$

The resulting rational function is then given by $Z(s) = A_N(s)/B_N(s)$. Program ECFE performs this algorithm and is the functional inverse of CFE. These techniques are called the *Euclidean algorithm.*

POLYNOMIAL GREATEST COMMON DIVISOR

The *greatest common divisor* (GCD) of polynomials a(s) and b(s) is the highest degree polynomial d(s) which exactly divides both a(s) and b(s). If the degree of d(s) is zero (d is a constant), then a(s) and b(s) are said to be *coprime.* In the above example, the GCD of the numerator and denominator polynomials was 9, so these polynomials are coprime. The first step in the continued fraction expansion of a(s)/b(s) is

$$a(s) = b(s)q_1(s) + r_1(s)$$

where $q_1(s)$ is the first quotient (s in the above example) and $r_1(s)$ is the first remainder ($6s^2 + 9$ in the above example). By single stepping through program CFE, you can follow along. Observe that from the above equation, any divisor of a(s) and b(s) must also be a divisor of $r_1(s)$. Hence, the GCD of a(s) and b(s) must also be the GCD of b(s) and $r_1(s)$. Now we invert and divide again

$$\begin{aligned} b(s) &= r_1(s)q_2(s) + r_2(s) \\ r_1(s) &= r_2(s)q_3(s) + r_3(s) \\ r_2(s) &= r_3(s)q_4(s) + r_4(s) \end{aligned}$$

$$\vdots$$

$$\begin{aligned} r_{N-2}(s) &= r_{N-1}(s)q_N(s) + r_N(s) \\ r_{N-1}(s) &= r_N(s)q_{N+1}(s) \end{aligned}$$

Since the degree of the remainder continually decreases, it must eventually become zero. The last equation shows that the desired GCD must also be the GCD of $r_{N-1}(s)$ and $r_N(s)$. Since $r_{N+1}(s) = 0$, the last non-zero remainder $r_N(s)$ must be the GCD of a(s) and b(s). Program PGCD calls CFE for the continued fraction expansion and selects the last non-zero remainder. Then it calls **PSCAL** to scale the final result.

PGCD({9 0 10 0 1}, {0 4 0 1}) = { 1 }

so we see that the numerator and denominator polynomials in the above example are coprime. For $a(s) = 12s^2 + 10s + 2$ and $b(s) = 3s^2 + 10s + 3$, then the GCD is $3s + 1$.

PGCD({2 10 12}, {3 10 3}) = {1 3}

Observe that the order of the arguments of PGCD is reversible.

The GCD of polynomials a(z) and b(z), say d(z), can be written as a linear combination of a(z) and b(z) since

$$r_1(z) = a(z) - b(z)q_1(z) = a(z) + [-q_1(z)]b(z)$$
$$r_2(z) = b(z) - q_2(z)r_1(z) = [-q_2(z)]a(z) + \{1 + [-q_2(z)][-q_1(z)]\}b(z)$$

up to $r_N(z)$. Thus, there exist polynomials s(z) and t(z) such that

$$d(z) = s(z)a(z) + t(z)b(z)$$

Program PGCDI also computes the GCD, but in addition it performs the above recurrence to compute s(z) and t(z).

$$\text{PGCDI}(\{2 \quad 10 \quad 12\}, \{3 \quad 10 \quad 3\}) = \begin{cases} 3\text{: } \{-10 \quad -30\} \\ 2\text{: } \{-6 \quad -10 \quad -12\} \\ 1\text{: } \{4 \quad 10 \quad 3\} \end{cases}$$

where the unnormalized GCD is {-10 -30},

$$d(z) = -30z - 10, \quad t(z) = -12z^2 - 10z - 6, \quad s(z) = 3z^2 + 10z + 4.$$

Using commands **PMPY** and **PADD**, we find that

$$s(z)a(z) = 36z^4 + 150z^3 + 154z^2 + 60z + 8$$
$$t(z)b(z) = -36z^4 - 150z^3 - 154z^2 - 90z - 18$$

Adding these two equations gives the unnormalized $d(z) = -30z - 10$.

Observe that the order of the arguments for PGCDI is not reversible.

$$\text{PGCDI}(\{\text{a(z) LIST}\}, \{\text{b(z) LIST}\}) = \begin{cases} 3\text{: \{GCD LIST\}} \\ 2\text{: \{t(z) LIST\}} \\ 1\text{: \{s(z) LIST\}} \end{cases}$$

SHIFTING THE CENTER OF COMPUTATION

There are a number of situations where efficient use of the MATHLIB polynomial commands requires a few tricks. Suppose f(z) is a polynomial in z and we wish to compute the product $f(z)f(z^{-1})$. It is clear how to do it symbolically, but this is slow. For example, consider the product

$$[1 + z + z^2] [1 + z^{-1} + z^{-2}] = z^{-2} + 2z^{-1} + 3 + 2z + z^2$$

Since the polynomial commands only consider positive (negative) powers of z, they cannot directly compute this product. However, suppose we redefine the problem as follows

$$[z^2 + z^3 + z^4] [z^2 + z + 1] = z^2 + 2z^3 + 3z^4 + 2z^5 + z^6$$

Now this is a product which **PMPY** can compute since it contains only non-negative powers of z.

$$\text{PMPY}(\{1\ \ 1\ \ 1\}, \{1\ \ 1\ \ 1\}) = \{1\ \ 2\ \ 3\ \ 2\ \ 1\}$$

and we simply need to note that the result should be multiplied by z^{-2}, not z^{-4}, since we used another trick, omitting the two leading zeros in the first argument and thus shifting it back by z^{-2}. Thus,

$$z^{-2}\{ [1 + z + z^2] [z^2 + z + 1] \} = z^{-2} + 2z^{-1} + 3 + 2z + z^2$$

This is called *shifting the center of computation.* The minimum shift is the minimum of the degree in z and the degree in z^{-1}.

A more sophisticated example is computing the composition of the polynomial f(w) and the polynomial $w = z + 2 + z^{-1}$. Basic polynomial composition computations was covered on page 450 of the MATHLIB manual. Program Y→ZT in the MAN7 WAVE SOLVE directory provides an easy example.

SYMBOLIC TRICKS

There are a number of tricks which can be useful in large symbolic computations. Ultimately, computational speed on any computer depends on the cleverness of the user. See MAN7 WAVE SYMB.

Suppose we have computed a large symbolic polynomial in z^n containing dozens of symbolic coefficients. We would like to collect all the coefficients of each power of z together. In particular, we would like to create a symbolic coefficient list. Maclaurin series command **CLIST** is one approach, but because the coefficients are symbolic and the equation is large, the best one can hope for is not to run out of memory. It is very slow. Consider the following program SPTLR

```
« → P N « P { 'z^&' 0 '&≠0 } ↑MATCH DROP EVAL DUP
  { } + → E L « 1 N FOR J 'L' P 'z' '&N' ^ 0 '&N' J ≠ 3
    →LIST ↑MATCH DROP EVAL E - COLCT 'z' J ^ 1 2
      →LIST ↑MATCH DROP EVAL STO+ NEXT L » » »
```

where P is the symbolic polynomial in z^n for n = 0, 1, . . . N, and N is the degree of the polynomial. The program begins by computing the coefficient of $z^0 = 1$. The **↑MATCH** command replaces all occurrences of z^n for $n > 0$ with the value of zero. **EVAL** then gets rid of all those terms leaving the ones we want. These are then stored in coefficient list L and also in E. Then we play the same trick for J = 1, 2, . . . N to isolate the coefficients with the additional steps:

1. The coefficients of z^0 will always be included in the isolated equation. E - COLCT gets rid of them.

2. After isolating the terms we want, we then must get rid of the z^n factor. We accomplish this by substituting 1 for z^n.

The result is the desired polynomial coefficient list. For those of you who have not gotten around to reading the HP 48 manual, & and & followed by other characters define wildcard names relative to the **↑MATCH** command. The above program only works with the wildcard symbols as written.

To quickly multiply a large polynomial in z by z^m, isolate the z^0 terms and multiply by $(z^m - 1)$. Push **EXCO**. Use « { 'z^&' 'z^(&+m)' } ↑MATCH DROP EVAL » for the other terms and combine. Now use **COLCT** to get rid of the unwanted z^0 terms which are not times z^m.

One of the general principles discussed on page 16 of the MATHLIB manual, is that of minimizing what is on the stack and in local memory. The following version of MULTI (see page 569 of the HP 48 owner's manual) that we call BMLTI uses global memory instead of stack memory, to apply program P to a stack object.

```
« → P « '□□□□' STO DO □□□□ P EVAL □□□□ OVER '□□□□'
    STO UNTIL SAME END □□□□ '□□□□' PURGE » »
```

where □ is the α LEFTSHIFTED 6 key. Still applying the same principle, program BEXCO is like **SEXCO**, but may be faster in large applications. Input M is a symbolic matrix.

```
« → M « SSIZE DUP EVAL × → S N « M SOB→ DROP
N →LIST 'TEMP' STO 1 N FOR J TEMP J GET "EXPAND "
  J + 3 DISP « EXPAN » BMLTI "COLLECT " J + 3 DISP «
  COLCT » BMLTI 'TEMP' J ROT PUT NEXT TEMP 'TEMP'
    PURGE OBJ→ DROP S →SOB » » »
```

The program converts M to a list which it stores in the global variable TEMP. Then it recalls M to the stack, one element at a time, and first expands it completely, then collects it completely using BMLTI. In some applications, we only want to get rid of parentheses, and not to expand z^8 into zxzxzxzxzxzxzxz. The following two programs can be useful. With equation E on the stack, P→S

```
« → E « E { 'z^&' 'Z(&)' } ↑MATCH DROP » »
```

will substitute pseudofunction Z(n) for z^n so that **EXPAN** cannot expand it. The back substitution is performed by S→P.

```
« → E « E { 'Z(&)' 'z^&' } ↑MATCH DROP » »
```

By using « P→S » as the program argument to **L1F1** and **S1F1**, these substitutions can be applied to lists and symbolic matrices.

Divide and conquer is another basic principle. Use these tricks to break your problem into pieces. An example of breaking large symbolic integrals into pieces is given on page 212 of the MATHLIB manual. Store the pieces in global memory as BEXCO does, and operate on each piece at a time. For example, suppose you have a nasty equation and half the terms include the **SIN** function. Then the **↑MATCH** substitution { 'SIN(&)' 0 } isolates the terms without the **SIN** function.

8

WEIERSTRASS ELLIPTIC FUNCTIONS

INTRODUCTION

This chapter explains the basics of the Weierstrass theory of elliptic integrals. It is equivalent to the Legendre theory discussed in Chapter 9 of the MATHLIB manual. The seventeen Weierstrass elliptic function and parameter conversion programs in the MAN1 subdirectory WEF are also discussed.

REDUCTION OF ELLIPTIC INTEGRALS

Consider the elliptic integral $I = \int R(x, y)\,dx$ where $y^2 = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$. Legendre's theorem states that integral I can be expressed as a linear combination (with constant coefficients) of integrals of the following types:

$$I_1 = \int \frac{dx}{y}, \qquad I_2 = \int \frac{a_0x^2/2 + a_1x}{y}\,dx, \qquad I_3 = \int \frac{dx}{(x - c)y}$$

where c is a constant.

LEGENDRE ELLIPTIC INTEGRALS

If y^2 has the Legendre form $y^2 = (1 - x^2)(1 - k^2x^2)$, then $I_1 = F$, $I_2 = \frac{1}{2}(F - E)$, and

$$I_3 = \int \frac{d(x^2)}{2(x^2 - c^2)[(1 - x^2)(1 - k^2x^2)]^{1/2}} - \frac{1}{c}\Pi$$

where the first integral on the right can be evaluated in terms of elementary functions and F, E, and Π are the Legendre elliptic integrals of the first, second, and third kind, respectively. Elliptic integrals of the first and second kinds have exactly two independent periods. Elliptic integrals of the third kind have three independent periods.

WEIERSTRASS ELLIPTIC INTEGRALS

In the Weierstrass form, $y^2 = 4x^3 - g_2x - g_3$, and the elliptic integrals of the three kinds are

$$I_1 = \int \frac{dx}{(4x^3 - g_2x - g_3)^{1/2}}$$

$$I_2 = \int \frac{x\,dx}{(4x^3 - g_2x - g_3)^{1/2}}$$

$$I_3 = \int \frac{dx}{(x - c)(4x^3 - g_2x - g_3)^{1/2}}$$

Let ω and ω' denote a pair of complex numbers with $IM(\omega'/\omega) > 0$. We call ω and ω' the half-periods of an elliptic function $f(z) = f(z + 2M\omega + 2N\omega')$ where M and N are integers. The study of elliptic functions f(z) can thus be reduced to the study of their properties over one period in two dimensions, which is called the *fundamental period parallelogram*. Let $W = 2M\omega + 2N\omega'$. Then the sums over all M and N excluding the M = N = 0 values are $g_2 = 60\Sigma W^{-4}$ and $g_3 = 140\Sigma W^{-6}$ which defines the *invariants* g_2 and g_3. The *discriminant* Δ is the number $\Delta = g_2^3 - 27g_3^2$. We consider only the cases where g_2 and g_3 are real, which covers most applications. Thus, Δ is real and the formulas are dichotomized by $\Delta > 0$ and $\Delta < 0$. The $\Delta = 0$ case is discussed by AMS 55, but the software does not have this case programmed since it can be approximated by Δ equals a small number. Homogeneity relations also allow the restriction to non-negative g_3.

Now $y^2 = 0 = 4e^3 - g_2e - g_3$ has three roots which we denote by e_n for n = 1, 2, 3. We label them to be unique by defining e_2 to be real and non-negative unless $g_3 = 0$ when $e_2 = 0$. Root $e_1 = -\alpha + i\beta$ where $\alpha \geq 0$ and $\beta > 0$. Root $e_3 = \text{CONJ}(e_1) = -\alpha - i\beta$.

Now that we have the basic parameter definitions in place, I can tell you the good news - that you probably will not need to understand them. Let ω denote the parameters ω, ω', SIGN(Δ); G denote the invariants g_2 and g_3; and E denote the roots e_1, e_2, and e_3. MATHLIB offers the following programs for parameter conversion between sets: G→E, E→G, ω→G, ω→E, G→ω, and E→ω. Thus, given the parameters of either set, you can convert to the other sets.

In addition, conversion to the complementary modulus k' is available with the programs ω→KP, G→KP, and E→KP. Calculation of the discriminant Δ is performed by G→Δ and E→Δ. Given ω and ω', ω→Q computes the complex nome q. Program ηOFω is an internal program used in the calculation of the Weierstrass sigma function.

THE WEIERSTRASS FUNCTIONS

The Weierstrass sigma function $\sigma(z|\omega, \omega') = \sigma(z; g_2, g_3)$ is evaluated by σOFZ. The Weierstrass zeta function $\zeta(z|\omega, \omega') = \zeta(z; g_2, g_3) = \sigma'(z)/\sigma(z)$ is the logarithmic derivative of $\sigma(z|\omega, \omega')$ and is evaluated by ZOZG. The Weierstrass $\wp(z|\omega, \omega') = \wp(z; g_2, g_3) = -\zeta'(z|\omega, \omega')$ is evaluated by POZG. The fourth function is the derivative $\wp'(z|\omega, \omega') = \wp'(z; g_2, g_3)$ which is evaluated by PPZG. These programs use the equations on pages 649 and 650 of AMS 55.

Returning to the above integral $I = \int R(x, y)dx$, let $x = \wp(z; g_2, g_3)$ and $y = \wp'(z; g_2, g_3)$. Since $[\wp'(z)]^2 = 4\wp^3(z) - g_2\wp(z) - g_3$, the integral $I = \int R[\wp(z), \wp'(z)]\wp'(z)dz$ where the integrand is a rational function of $\wp(z)$ and $\wp'(z)$ and may be evaluated in terms of Weierstrass zeta functions and their derivatives.

For more details, see Chapter 18 of AMS 55 and the references of Chapters 9, 10, and 11 of the MATHLIB manual. Erdelyi has a good explanation of how one evaluates elliptic integrals with these functions. Gradshteyn and Ryzhik give a nice summary of the properties of elliptic functions.

EXAMPLE COMPUTATIONS

The input-output stack conventions for the set conversion programs G→E, E→G, ω→G, ω→E, G→ω, and E→ω is

3: ω	3: e_1	
2: ω'	2: e_2	2: g_2
1: SIGN(Δ) or Δ	1: e_3	1: g_3

These are also the input conventions for G→KP, E→KP, G→Δ, E→Δ, and ω→Q. The arguments for ω→KP(ω, ω') are ω and ω'. Some numerical examples are:

Given ω = 10 and ω' = 11i, ω→E(10, (0,11), 1) gives the values e_1 = 1.68430411404E-2, e_2 = -2.16625762695E-3, and e_3 = -1.46767835133E-2 where the 1 defines the sign(Δ) > 0. As a check, now push E→ω which gives ω = 10, ω' = (0,11), and Δ = 8.99023187289 E-10. Now push ω→G to get g_2 = 1.00757736252E-3 and g_3 = 2.14200 999934E-6. Now as an accuracy check push G→ω to get ω = 10.0000000446, ω' = (0, 10.9999999044), and Δ = 8.99023219043E-10.

Similarly, for Δ < 0, ω→E(10, (0,11), -1) gives e_1 = (-2.16625762695E-3, 3.08425890166E-2), e_2 = 4.3325152539E-3, and e_3 = (-2.16625762695E-3, -3.08425890166E-2). Pushing E→G yields g_2 = -3.74874912371E-3 and g_3 = 1.65668099372E-5. Now G→Δ gives the value for Δ = -6.00920194277E-8.

The stack conventions for POZG, PPZG, ZOZG, and σOZG are all the same. For example, POZG(z, g_2, g_3) evaluates $\wp(z; g_2, g_3)$. Thus, POZG((.07,.1), 10, 2) = (-22.9745001048, -63.0532328498) and POZG ((.1,.03), -10, 2) = (76.5883327117, -50.5037916911).

ω→Q(10, 20i, 1) = 1.86744273171E-3, ω→Q(5, 7i, -1) = (0, .110901278363, PPZG((3, 2), ω→G(5, 7i, -1)) = (-6.91819875372E-4, .047713052692), σOZG((.4, 1.3), 8, 4) = (.278079995729, 1.27278494989), σOZG((.8, .4), 7, 6) = (.814657646488, .388194735751), ZOZG((9, 19), ω→G(10, (0, 20), 1)) = (.0743949200186, -.000468795788), ω→KP(10, (0, 20)) = .985171431018, and ω→KP(5, (0, 7)) = .906274901297.

Please note the abuse in notation in the above PPZG and ZOZG examples. If you enter what I have written algebraically, the HP 48 will complain over wrong argument count. If you simply enter the arguments and push the WEF menu keys, everything is fine.

9

WAVELETS AND FILTER BANKS

INTRODUCTION

The subject of wavelets has been studied for several years by the mathematics community, as an alternative to the more traditional Fourier based analysis techniques. The close connection between multirate filter bank theory and wavelets has resulted in much interest in wavelets by the signal processing community. While the ink is not dry on hundreds of research papers, this and the next two chapters attempt to present a partial overview of the subject with as little mathematical eloquence as possible, and with as many detailed steps in the derivations as seems reasonable.

Wavelets can be a difficult subject to learn and use if one does not have the required symbolic, polynomial, and digital processing tools in place. Results are often stated in the most complicated way possible and some emerging texts on wavelets, while claiming to be introductions to the subject, read more like highly specialized research monographs.

Before studying Fourier approximation theory, one normally learns what a sine function is. This is our approach in the next 3 chapters. Over 70 wavelet related programs are presented.

One of the mistruths that pervades the wavelet literature is that they are easy to compute. Compared with even a hypergeometric function, they are not. Classical orthogonal approximation theory involves functions such as sines, cosines, and polynomials in z^n which in today's computer technology are easy to compute using power series and other expansions. Now we have to deal with translations and dilations of functions which are evaluated by a new set of digital algorithms. Sines and cosines are numerically orthogonal to more than 10 digits using simple Chebyshev approximations. This is observed with DFTs. Getting 10 digit orthogonality between a numerically approximated Daubechies scaling and wavelet function, by comparison, requires an extraordinary amount of computation. See pages 66 and 101.

All orthogonal and biorthogonal wavelet bases with compactly supported wavelets are special cases of FIR filter bank theory. The orthogonal case is the Daubechies phase nonlinear approach and the biorthogonal one is the phase linear approach wherein much of the orthogonality may be given up. Cosine modulated filter banks and other techniques are currently being investigated.

This chapter continues the discussion of wavelets which began in Appendix G of the MATHLIB manual. The software we will be discussing is located in the WAVE subdirectory of the MAN7 directory. The first subdirectory in WAVE is FFT2D which contains the two dimensional FFTs discussed on page 565 of the MATHLIB manual. The next three directories are the wavelet program directories.

We begin by completing the discussion of wavelet coefficient matrices. In particular, we focus on the 2-band case in this chapter. Since the scaling and wavelet function orthogonality for this case is built on the concept of quadrature mirror filters, we review it in detail. Then we examine methods of computing scaling and wavelet functions. Next we consider orthogonal and biorthogonal perfect reconstruction filter banks.

The 2-band case is, in essence, a basic lowpass-highpass frequency decomposition. The scaling function is the lowpass filter and the wavelet is the highpass filter. Chapter 10 then expands the theory to the M-band case, where the scaling function remains the lowpass filter, but now we have a collection of wavelets providing bandpass filters. Chapter 11 then addresses wavelets and approximations.

The following discussions assume intimate knowledge of z transforms. We list numerous relationships which are foundational to the theory.

QUADRATURE MIRROR FILTERS

Let $h_1(n)$ be a FIR filter with N real non-zero coefficients. We define the corresponding *quadrature mirror filter* (QMF) as $h_2(n) = (-1)^n h_1(n)$ for $n = 0, 1, \ldots N-1$. The z transform of $h_2(n)$ is

$$H_2(z) = \sum_{n=-\infty}^{\infty} z^{-n} h_2(n) = \sum_{n=-\infty}^{\infty} z^{-n}(-1)^n h_1(n) = H_1(-z)$$

The Fourier transform of these filters can be computed by evaluating z on the unit circle as $z = e^{i\omega}$. When plotted over $\omega \in [0, \pi]$, $H_2(e^{i\omega}) = H_1(-e^{i\omega})$ is the mirror image of $H_1(e^{i\omega})$. Define $h_3(n) = h_1(N-1-n)$ to be a time reversed $h_1(n)$. Then its z transform is

$$\begin{aligned} H_3(z) &= \sum_{n=0}^{N-1} z^{-n} h_3(n) = \sum_{n=0}^{N-1} z^{-n} h_1(N-1-n) \\ &= z^{-(N-1)} \sum_{m=0}^{N-1} z^{m} h_1(m) = z^{-(N-1)} H_1(z^{-1}) \end{aligned}$$

which apart from the delay $z^{-(N-1)}$, is like $H_1(z)$, but a function of z^{-1} instead of z. Also define $h_4(n) = (-1)^{N-1-n} h_1(N-1-n)$. Then

$$\begin{aligned} H_4(z) &= \sum_{n=0}^{N-1} z^{-n} h_4(n) = \sum_{n=0}^{N-1} z^{-n}(-1)^{N-1-n} h_1(N-1-n) \\ &= z^{-(N-1)} \sum_{m=0}^{N-1} z^{m}(-1)^{m} h_1(m) = z^{-(N-1)} H_1(-z^{-1}) \end{aligned}$$

which apart from the delay $z^{-(N-1)}$, is like $H_2(z)$, but a function of z^{-1} instead of z. Thus,

$$H_3(e^{i\omega}) = e^{-i\omega(N-1)} H_1(e^{-i\omega}) \quad \text{and} \quad H_4(e^{i\omega}) = e^{-i\omega(N-1)} H_1(-e^{-i\omega}).$$

Noting that $-1 = e^{i\pi}$, we also have that $H_2(e^{i\omega}) = H_1(e^{i(\omega+\pi)})$,

$$e^{i\omega(N-1)} H_3(e^{i\omega}) = H_1(e^{-i\omega}) = H_1^*(e^{i\omega}),$$

$$e^{i\omega(N-1)} H_4(e^{i\omega}) = H_1(e^{i(-\omega+\pi)}) = H_2^*(e^{i\omega}).$$

Hence, $|H_3(e^{i\omega})|^2 = |H_1(e^{i\omega})|^2$ and $|H_4(e^{i\omega})|^2 = |H_2(e^{i\omega})|^2$.

PARAUNITARY CONDITIONS

Recall that the discussion in Appendix G presented conditions on the coefficient matrix A such that $P^H(z^{-1})P(z) = cI$. These conditions were summarized at the top of page 571. After deriving a few relations, we will revisit these conditions. Define $H_1(z) = H_{10}(z) + z^{-1}H_{11}(z)$ where

$$H_{10}(z) = \sum_{n=0}^{N/2} h_1(2n)z^{-2n} \qquad H_{11}(z) = \sum_{n=0}^{N/2-1} h_1(2n+1)z^{-2n} \quad (1)$$

where we will assume that N *is an even integer*. Observe that $H_{10}(-z) = H_{10}(z)$ and $H_{11}(-z) = H_{11}(z)$. Then we have the relations

$$\begin{aligned} H_1(z)H_1(z^{-1}) &= H_{10}(z)H_{10}(z^{-1}) + H_{11}(z)H_{11}(z^{-1}) \\ &\quad + zH_{10}(z)H_{11}(z^{-1}) + z^{-1}H_{11}(z)H_{10}(z^{-1}) \end{aligned}$$

$$\begin{aligned} H_1(-z)H_1(-z^{-1}) &= H_{10}(z)H_{10}(z^{-1}) + H_{11}(z)H_{11}(z^{-1}) \\ &\quad - zH_{10}(z)H_{11}(z^{-1}) - z^{-1}H_{11}(z)H_{10}(z^{-1}) \end{aligned}$$

Adding these equations gives

$$\begin{aligned} &H_1(z)H_1(z^{-1}) + H_1(-z)H_1(-z^{-1}) \\ &\quad = 2H_{10}(z)H_{10}(z^{-1}) + 2H_{11}(z)H_{11}(z^{-1}) \end{aligned} \quad (2)$$

so it follows that

$$\begin{aligned} &H_1(z)H_3(z) + H_2(z)H_4(z) \\ &= 2z^{-(N-1)}[H_{10}(z)H_{10}(z^{-1}) + H_{11}(z)H_{11}(z^{-1})] \end{aligned}$$

and that the following identities hold

$$H_1(z) + H_2(z) = 2H_{10}(z) \qquad H_1(z) - H_2(z) = 2z^{-1}H_{11}(z)$$

$$H_3(z) + H_4(z) = 2z^{-(N-1)}H_{10}(z^{-1}) \qquad H_3(z) - H_4(z) = 2z^{-(N-1)}zH_{11}(z^{-1})$$

Now consider the paraunitary conditions on the coefficients of $h_1(n)$. Let $a_n^0 = h_1(n)$ and $a_n^1 = h_4(n)$. From page 571 of the MATHLIB manual

$$\sum_{k=-\infty}^{\infty} \bar{a}_{k+2r}^{s} a_{k+2r'}^{s'} = c\delta_{rr'}\delta_{ss'} \qquad \bar{a} = a^*, \quad a_k^s = 0 \text{ for } k < 0 \text{ and } k \geq 2g$$

We show that these conditions imply that $H_{10}(z)H_{10}(z^{-1}) + H_{11}(z)H_{11}(z^{-1}) = 2$ and thus $|H_{10}(e^{i\omega})|^2 + |H_{11}(e^{i\omega})|^2 = 2$ for real coefficients.

$$\begin{aligned} H_{10}(z)H_{10}(z^{-1}) + H_{11}(z)H_{11}(z^{-1}) &= \\ &\sum_{m=0}^{N/2}\sum_{n=0}^{N/2} h_1(2m)h_1(2n)z^{-2(m-n)} \\ &+ \sum_{m=0}^{N/2-1}\sum_{n=0}^{N/2-1} h_1(2m+1)h_1(2n+1)z^{-2(m-n)} \end{aligned} \tag{3}$$

Let $m - n = k$. Then this equation can be written as

$$\begin{aligned} &\sum_k z^{-2k}\left[\sum_{n=0}^{N/2} h_1(2k+2n)h_1(2n) + \sum_{n=0}^{N/2-1} h_1(2k+2n+1)h_1(2n+1)\right] \\ &= \sum_k z^{-2k}\sum_{n=0}^{N-1} h_1(2k+n)h_1(n) = \sum_k z^{-2k}2\delta_{k0} = 2 \end{aligned} \tag{4}$$

Thus, if the filter coefficients satisfy the paraunitary conditions, we have from (2) that

$$H_1(z)H_1(z^{-1}) + H_4(z)H_4(z^{-1}) = 4 \tag{5}$$

where we have used the identity

$$H_4(z)H_4(z^{-1}) = z^{-(N-1)}H_1(-z^{-1})z^{N-1}H_1(-z) = H_1(-z^{-1})H_1(-z) \tag{6}$$

Consequently, define the filter vector $H^T(z) = [H_1(z) \;\; H_4(z)]$. Then we have proved that $H^H(z^{-1})H(z) = H^H(-z^{-1})H(-z) = 4$.

Now $h_1(n)$ is the scaling coefficient vector and $h_4(n)$ is the wavelet coefficient vector. Observe that by construction for all $k \in \mathbf{I}$

$$\begin{aligned} &\sum_{n=0}^{N-1} h_4(2k+n)h_4(n) \\ &= \sum_{n=0}^{N-1} (-1)^{2k+N-1-n}h_1(2k+N-1-n)(-1)^{N-1-n}h_1(N-1-n) \\ &= \sum_{m=0}^{N-1} (-1)^{2k+m}h_1(2k+m)(-1)^m h_1(m) = \sum_{m=0}^{N-1} h_1(2k+m)h_1(m) = 2\delta_{k0} \end{aligned}$$

$$\sum_{n=0}^{N-1} h_1(2k + n)h_4(n) = \sum_{n=0}^{N-1} h_1(2k + n)(-1)^{N-1-n}h_1(N - 1 - n) = 0$$

so the wavelet coefficients also satisfy the paraunitary conditions.

Hence, the orthogonality condition on the scaling vector $h_1(n)$ gives rise to paraunitary filter banks with perfect reconstruction. The wavelet vector is derived from the scaling vector in a way that the orthogonality is preserved. It is the nature of the QMF that makes the scaling and wavelet vectors orthogonal. When we get to the biorthogonal case, you will observe that this latter orthogonality due to the QMF is all that is left.

Now we depart from the theory and learn to compute wavelet matrices as well as scaling and wavelet functions. Then we will return to it.

DISCRETE HERMITE TRANSFORM

The first page of the WAVE HAAR menu contains the programs given in Appendix G of the MATHLIB manual. On the second page of the menu, DHPM uses DHP to build the discrete Hermite transform matrix.

EXAMPLE WAVELET MATRICES

Programs WD4 and WD6 construct 4 and 6 coefficient wavelet matrices. See page 571 of the MATHLIB manual for definitions. The arguments are angles in radians and the equations are from Tewfik [10]. The resulting matrices always satisfy the wavelet matrix conditions, but not necessarily the Daubechies regularity condition which we will examine below. WD4 constructs a genus 2 matrix and WD6 constructs a genus 3 matrix. Genus is Daubechies regularity plus one. Programs R2G4, R4G4, R4G8 construct example flat wavelet matrices with the indicated Rank and Genus.

Program QMFL in the WAVE directory computes the QMF coefficients of $h_4(n)$ given the scaling vector list $h_1(n)$. Program QMF uses QMFL to construct the filter wavelet matrix A whose first row is the $h_1(n)$ coefficients and whose second row is the $h_4(n)$ coefficients. The sign of the second row is arbitrary. By using $-h_4(n)$, the plots of the wavelet functions are the same as in Daubechies' paper [3].

SINDX is a symbolic version of command **EINDX**. SDEC, SINTP, and SCMB are symbolic vector versions of the commands **VDEC, VINTP,** and **VCMB**. LNLV and LDLV are special list interleave and deinterleave programs used internally by the *Pollen product* program discussed below.

Returning to the third page of the WAVE HAAR subdirectory, program TROW will test the orthogonallity conditions of any wavelet matrix. The inputs are the matrix, the first row number, the second row number, and the offset. TESTV tests certain regularity conditions. SWD4 and SWD6 are symbolic versions of the first row of WD4 and WD6, respectively, except for a normalization factor. TRWS is a symbolic version of TROW. Examples are given below.

DAUBECHIES' D_4 AND D_6 COEFFICIENTS

Daubechies' coefficients, denoted by D_{2g}, have the property that for each g, $H_1(z) = [1 + z^{-1}]^g Q_g(z)$. This is required by her regularity condition. Thus, the polynomial {1 2 1} must divide the D_4 coefficients and {1 3 3 1} must divide the D_6 coefficients. As our first example, let us compute angle θ such that WD4 will compute the D_4 coefficients given on page 571 of the MATHLIB manual. The program

« SWD4 {1 2 1} PDVD DROP »

symbolically divides the output list from SDW4. Since the remainder must be zero, we immediately obtain the equation 2 – 4C = 0 where C = cos θ and S = sin θ in the SWD4 coefficient list. Thus, cos θ = 0.5 so θ = π/3. Now let us try it. « p 3 / WD4 » computes Daubechies' D_4 coefficients. The coefficients in her paper are these divided by √2.

The orthogonality conditions may be tested using TROW. The programs: « M 1 1 O TROW », « M 1 2 O TROW », and « M 2 2 O TROW » test the orthogonality of the first row with the first row, the first row with the second row, and the second row with the second row, respectively. Matrix M is the output of WD4 and O defines the offset equal to R×O where R is the rank of M (the number of rows) and O is an integer. Other example wavelet matrices can be tested similarly. TRWS allows you to test the symbolic coefficients output by SWD4 and SWD6. **RSUM** can be used to verify the linear conditions stated on page 571 of the MATHLIB manual.

Except for division by a factor of 4, SWD6 computes the first row of the genus 3 wavelet matrix output from WD6 using the *Pollen product* program in the SYMB subdirectory. It may not run if you change the structure of the WAVE directory. For this 6 coefficient system, there are two angles which must be specified. Define $C = \cos \theta_1$, $S = \sin \theta_1$, $D = \cos \theta_2$, and $T = \sin \theta_2$. The program

```
« SWD6 {1 3 3 1} PDVD DROP »
```

symbolically divides the output list from SWD6. Since the remainder must be zero in the case of Daubechies' coefficients, we have the equations:

$$\begin{aligned} -4 + 8CT - 8SD - 8C + 4S + 8D - 4T &= 0 \\ -12 + 16CT - 16SD - 24C + 8S + 24D - 8T &= 0 \\ -8 + 8CT - 8SD - 16C + 4S + 16D - 4T &= 0 \end{aligned}$$

Subtracting the third equation from the first gives $4 + 8C - 8D = 0$ so we obtain the relation $D = C + 0.5$. With the remainder on Level 1 of the stack, type in {D 'C+.5'} ENTER MAT↑ SEXCO to make this substitution and obtain the simplified equations

$$\begin{aligned} -8CS + 8CT - 4T &= 0 \\ -16CS + 16CT - 8T &= 0 \\ -8CS + 8CT - 4T &= 0 \end{aligned}$$

which are the same equation, $CS = (C - .5)T$, repeated three times. Now

$$T = \sqrt{1 - D^2} = \sqrt{1 - (C + .5)^2} \qquad S = \sqrt{1 - C^2}$$

so $[CS]^2 = C^2(1 - C^2) = (C - .5)^2[1 - (C + .5)^2]$ which reduces to the quadratic equation $C^2 - 2C + 0.375 = 0$. The program

```
« {.375 -2 1} 0 1E-10 100 AROOT »
```

will compute the root vector [.209430584958 1.79056941504]. The first root is the only feasible answer so $\theta_1 = 1.35980373244$ and $\theta_2 = -.782106384744$. Using these angles, WD6 will compute Daubechies' coefficients (the ones in her paper are divided by $\sqrt{2}$). Use TROW and **RSUM** to check the conditions. Tewfik's entire paper is on computing good scaling vectors without Daubechies' regularity condition.

DAUBECHIES' REGULARITY CONDITION

Daubechies' regularity condition is that the filter $H_1(z)$ have as many zeros at -1 as possible. We have already exploited this fact in the above examples where we noted that $H_1(z) = [1 + z^{-1}]^g Q_g(z)$. In Fourier transform notation this is $H_1(e^{i\omega}) = [1 + e^{-i\omega}]^g Q_g(e^{i\omega})$ so $H_1(e^{i\omega})$ has a zero of order g at $\omega = \pi$. This is equivalent to the sum (see page 106)

$$\sum_{n=0}^{2g-1} (-1)^n n^m h_1(n) = 0 \qquad m = 0, 1, \ldots g - 1$$

Program TESTV computes this sum. The D_4 coefficients satisfy this equation for m = 0 and 1. The D_6 coefficients satisfy it for $m \le 2$.

COMPUTING DAUBECHIES' SCALING VECTOR D_{2g}

Now we will compute Daubechies' coefficients in the general case. The program D2G in the WAVE SOLVE subdirectory does this computation. The argument of D2G is the genus g = 2, 3, ... From Daubechies' paper [3], we need to solve the equation

$$P_g(y) = Q_g(z)Q_g(z^{-1}) = \sum_{k=0}^{g-1} \binom{g-1+k}{k} \left[\frac{1}{2} - \frac{1}{4}(z + z^{-1})\right]^k \quad (7)$$

for $Q_g(z)$ where

$$y = \frac{1}{2} - \frac{1}{4}(z + z^{-1}) \qquad 1 - y = \frac{1}{2} + \frac{1}{4}(z + z^{-1}) \quad (8)$$

and $P_g(y)$ is the solution of the Bezout equation

$$(1 - y)^g P_g(y) + y^g P_g(1 - y) = 4 \quad (9)$$

Observe that $y(z = e^{i\omega}) = \sin^2(\omega/2)$ and $1 - y(z = e^{i\omega}) = \cos^2(\omega/2)$. Program QOZ1 illustrates the slow inaccurate brute force solution. It computes the composition of the $P_g(y)$ polynomial with the y polynomial in z. Then it computes the roots and chooses the minimum phase ones. A better approach is to observe that if we solve for the roots of $P_g(y)$

$$P_g(y) = c(y - r_0)(y - r_1) \ldots (y - r_{g-1}) \qquad c \in \mathbf{C}$$

then the roots in z can be computed from

$$y = -\frac{z}{4}(1 - z^{-1})^2 = r_n \qquad n = 0, 1, \ldots g - 1 \qquad (10)$$

for each of the y roots. This is equivalent to solving the quadratic equation $1 + (4r_n - 2)z^{-1} + z^{-2} = 0$. The minimum phase roots are given by the equation

$$z_n^{-1} = 1 - 2r_n + 2\sqrt{r_n^2 - r_n} \qquad n = 0, 1, \ldots g - 1$$

Program QOFZ performs this calculation. Since QOFZ solves a polynomial of degree g – 1, instead of degree 2(g – 1), the solution is both faster and more accurate than the approach in Daubechies' paper which is implemented by QOZ1. Given these roots, we then multiply by $[(1 + z^{-1})/2]^g$ (which is computed by MOZ) and normalize the resulting coefficients. Program QMF will construct the wavelet matrix and you are ready to test the matrix with the WAVE HAAR programs TROW and TESTV.

COMPUTING THE SCALING AND WAVELET FUNCTIONS

Let the scaling function be s(x) and the wavelet function be w(x). Then s(x) and w(x) are completely determined from the wavelet matrix

$$A = \begin{bmatrix} a_0 & a_1 & \cdots & a_{2g-1} \\ b_0 & b_1 & \cdots & b_{2g-1} \end{bmatrix}$$

by solving

$$s(x) = \sum_{k=0}^{2g-1} a_k s(2x - k) \qquad w(x) = \sum_{k=0}^{2g-1} b_k s(2x - k) \; .$$

MATHLIB uses the elegant approach suggested by Strang [9].† The scaling function equation can be written for g = 2 as the equation

† Programs for implementing Daubechies' computational method [3] are given in Chapter 11.

$$\begin{bmatrix} s(1) \\ s(2) \end{bmatrix} = \begin{bmatrix} a_1 & a_0 \\ a_3 & a_2 \end{bmatrix} \begin{bmatrix} s(1) \\ s(2) \end{bmatrix} \qquad s(0) = s(3) = 0$$

and similarly for g = 3 as

$$\begin{bmatrix} s(1) \\ s(2) \\ s(3) \\ s(4) \end{bmatrix} = \begin{bmatrix} a_1 & a_0 & 0 & 0 \\ a_3 & a_2 & a_1 & a_0 \\ a_5 & a_4 & a_3 & a_2 \\ 0 & 0 & a_5 & a_4 \end{bmatrix} \begin{bmatrix} s(1) \\ s(2) \\ s(3) \\ s(4) \end{bmatrix} \qquad s(0) = s(5) = 0$$

Program SMAT sets up the matrix. Now the linear condition on the scaling vector coefficients can be written as $a_0 + a_2 + \ldots a_{2(g-1)} = 1$ and $a_1 + a_3 + \ldots a_{2g-1} = 1$. Thus, the sums of the values in each column of these matrices is 1. It follows that [[1 1 ... 1]] is a left eigenvector of these matrices corresponding to the eigenvalue $\lambda = 1$. Hence, the solution of these equations is simply the right eigenvector for $\lambda = 1$.

Program SSOV solves the SMAT output matrix for the right eigenvector corresponding to $\lambda = 1$ using the power method. If the matrix has several eigenvalues clustered near one, the solution may never converge because the power method tends to have problems with repeated eigenvalues. If SSOV does not converge, you may either use **SCHRD** or try modifying the program to solve for the eigenvector corresponding to $1 \pm \varepsilon$ where ε is a small number less than the required precision.

WSOV calls these two programs and then uses the s(x) solution to compute the initial values of w(x) using for g = 2 the equation

$$\begin{bmatrix} w(1) \\ w(2) \end{bmatrix} = \begin{bmatrix} b_1 & b_0 \\ b_3 & b_2 \end{bmatrix} \begin{bmatrix} s(1) \\ s(2) \end{bmatrix} \qquad w(0)=w(3)=0$$

and similarly for g = 3

$$\begin{bmatrix} w(1) \\ w(2) \\ w(3) \\ w(4) \end{bmatrix} = \begin{bmatrix} b_1 & b_0 & 0 & 0 \\ b_3 & b_2 & b_1 & b_0 \\ b_5 & b_4 & b_3 & b_2 \\ 0 & 0 & b_5 & b_4 \end{bmatrix} \begin{bmatrix} s(1) \\ s(2) \\ s(3) \\ s(4) \end{bmatrix} \qquad w(0) = w(5) = 0$$

Once these initial solutions are computed, SOVN uses recursion to compute the values in between by the formulas (also see page 42)

$$s(x/2) = \sum_{k=0}^{2g-1} a_k s(x - k) \qquad w(x/2) = \sum_{k=0}^{2g-1} b_k s(x - k) .$$

Observe that these are compact support scaling and wavelet functions having the same support as the wavelet matrix scaling and wavelet vectors, $2g - 1$. The inputs to SOVN are the wavelet matrix, $\varepsilon > 0$ which sets the convergence precision of the eigenvector solution, and n which defines the resolution. The output is a $2 \times 2^{n+1}(2g - 1) + 1$ matrix. The first row is the scaling function and the second row is the wavelet. The below program computes the D_{10} case and plots first the scaling function and then the wavelet. Then it plots their spectrums.

```
« 5 D2G QMF 1E-10 4 SOVN M→RL OBJ→ DROP SWAP
DUP2 PLT1 NEG PLT3 128 TWIDL DROP OVER 128 REDN
OVER 128 FFT VABS HALF1 PLT1 3 PICK 128 REDN
SWAP 128 FFT VABS HALF1 PLT3 »
```

WAVELET ORTHOGONALITY RELATIONS

Now that we have computed a scaling and wavelet function, let us examine the consequences of the conditions placed on the scaling and wavelet vectors. Let ℓ^2 denote the space of absolutely square summable real sequences α_n where it is understood that $n \in \mathbf{I}$ and finite sequences are zero filled so that α_n is well defined for all n. Define the norm of that space by

$$||\alpha||_2 = \left[\sum_n |\alpha_n|^2\right]^{1/2} < \infty$$

where it is understood that the summation is over all $n \in \mathbf{I}$. Similarly, define the norm of a real function f by

$$||f||_2 = \left[\int_{-\infty}^{\infty} |f(x)|^2 \, dx\right]^{1/2} < \infty$$

which is commonly called the L^2 norm. The ℓ and L symbols are used because these are Lebesgue integrals and norms.

Consider the consequences of the scaling vector coefficient conditions combined with the dilation equation on the integral

$$\begin{aligned}
&\int_{-\infty}^{\infty} s(x - j)s(x - k)\, dx \\
&= \int_{-\infty}^{\infty} \sum_m a_m \, s(2[x - j] - m) \sum_n a_n \, s(2[x - k] - n)\, dx \\
&= \frac{1}{2}\sum_m \sum_n a_m \, a_n \int_{-\infty}^{\infty} s(2[x - j] - m)\, s(2[x - k] - n)\, 2dx
\end{aligned}$$

for $j,k \in \mathbb{I}$. Define $r = k - j$ and let $n = 2r + m$. Then we have

$$\frac{1}{2}\sum_m \sum_{r=j-k} a_m \, a_{2r+m} \int_{-\infty}^{\infty} s^2(2[x - j] - m)\, 2dx$$

Now if $s(x) \in L^2$, then the integral is finite. Consequently, for $j \neq k$, the original integral is zero since the coefficient sum is zero for $j \neq k$. Thus, the scaling function is orthogonal to its integer translates.

$$\int_{-\infty}^{\infty} s(x - j)\, s(x - k)\, dx = ||s||_2^2 \, \delta_{jk}$$

where δ_{jk} is the Kronecker delta function. Next consider the wavelet.

$$\begin{aligned}
&\int_{-\infty}^{\infty} w(x - j)w(x - k)\, dx \\
&= \int_{-\infty}^{\infty} \sum_m b_m \, s(2[x - j] - m) \sum_n b_n \, s(2[x - k] - n)\, dx \\
&= \frac{1}{2}\sum_m \sum_n b_m \, b_n \int_{-\infty}^{\infty} s(2[x - j] - m)\, s(2[x - k] - n)\, 2dx
\end{aligned}$$

for $j,k \in \mathbb{I}$. Define $r = k - j$ and let $n = 2r + m$. Then we have

$$\frac{1}{2}\sum_m \sum_{r=j-k} b_m \, b_{2r+m} \int_{-\infty}^{\infty} s^2(2[x - j] - m)\, 2dx$$

so if $s(x) \in L^2$, the integral is finite and equals $||s(x)||_2^2$. Noting that

$$\sum_m b_m \, b_{2r+m} = \sum_m (-1)^m a_m \, (-1)^{2r+m} a_{2r+m} = \sum_m a_m a_{2r+m}$$

it follows that the wavelets are also orthogonal to their integer translates.

$$\int_{-\infty}^{\infty} w(x - j)\, w(x - k)\, dx = ||s||_2^2\, \delta_{jk}$$

Finally, the orthogonality between w(x) and s(x) follows from

$$\int_{-\infty}^{\infty} s(x - j)w(x - k)\, dx$$
$$= \int_{-\infty}^{\infty} \sum_m a_m\, s(2[x - j] - m) \sum_n b_n\, s(2[x - k] - n)\, dx$$
$$= \frac{1}{2}\sum_m \sum_n a_m\, b_n \int_{-\infty}^{\infty} s(2[x - j] - m)\, s(2[x - k] - n)\, 2dx$$

so by the same argument

$$\int_{-\infty}^{\infty} s(x - j)\, w(x - k)\, dx = 0$$

LINEAR CONDITIONS

The integral of the dilation equation is consistent with the linear condition placed on the scaling vector coefficients

$$\int_{-\infty}^{\infty} s(x)\, dx = \int_{-\infty}^{\infty} \sum_m a_m\, s(2x - m)\, dx$$
$$= \frac{1}{2}\sum_m a_m \int_{-\infty}^{\infty} s(2x - m)\, d(2x) = \frac{1}{2}\sum_m a_m \int_{-\infty}^{\infty} s(x)\, dx$$

Since $\int s(x)\, dx \neq 0$, we can divide both sides of the equation and obtain the linear condition on the scaling vector coefficients

$$\sum_m a_m = 2$$

Similarly,

$$\int_{-\infty}^{\infty} w(x)\, dx = \int_{-\infty}^{\infty} \sum_m b_m\, s(2x - m)\, dx$$
$$= \frac{1}{2}\sum_m b_m \int_{-\infty}^{\infty} s(2x - m)\, d(2x) = \frac{1}{2}\sum_m b_m \int_{-\infty}^{\infty} s(x)\, dx = 0$$

Since the sum of the wavelet vector coefficients is zero, the integral of the wavelet is zero. As a quick check, compute the D_4 scaling and wavelet functions with SOVN with n = 5. Using command **RSUM**, the scaling vector sums to 64 and the wavelet vector sums to −1.7E−10. These sums approximate the above linear condition integrals. The number of values is 193. Use the sequence: 1 RSPLT DOT to check the orthogonality of the scaling and wavelet functions. The result is .04. Now compute the row norms with **RABS2** to get about 64. Thus, the orthogonality of the approximated functions is $\sqrt{}.04/64 = .025$ (32 dB), which is less than 2 digits. This justifies my comments on page 53.

COMPUTING INDIVIDUAL VALUES

Program VALUE in the WAVE SOLVE menu will extract values from the scaling and wavelet vectors output by SOLN. The first argument V is the vector. The second argument N is the support which equals 2g - 1. The third argument is the value of x. The program computes the value of the function at x MOD N. For example, given the wavelet for D_4 with n = 5, VALUE(V,3,1) = .3660254037 and VALUE(V,3,2) = 1.36602540374. Since Daubechies effectively divides her wavelet coefficients by $\sqrt{2}$ and negates them, her wavelet values differ by $-\sqrt{2}$.

PERFECT RECONSTRUCTION FILTER BANKS

We now examine in more detail the properties associated with perfect reconstruction. Form the 2 x 2 *aliasing component* matrix as [8]

$$A(z) = [H(z) \quad H(-z)] = \begin{bmatrix} H_1(z) & H_1(-z) \\ H_4(z) & H_4(-z) \end{bmatrix}$$

The determinant of A(z) is (see page 57)

$$\begin{aligned} \text{DET } A(z) &= H_1(z)H_4(-z) - H_1(-z)H_4(z) \\ &= H_1(z)(-z)^{-(N-1)}H_1(z^{-1}) - H_1(-z)z^{-(N-1)}H_1(-z^{-1}) \\ &= -z^{-(N-1)}[H_1(z)H_1(z^{-1}) + H_1(-z)H_1(-z^{-1})] \\ &= -4z^{-(N-1)} \end{aligned}$$

simply a constant times a delay. This is a fundamental property of perfect reconstruction filter banks. Similarly,

$$B(z) = A(z)\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \sqrt{2}\begin{bmatrix} H_{10}(z) & z^{-1}H_{11}(z) \\ -z^{-(N-1)}zH_{11}(z^{-1}) & z^{-(N-1)}H_{10}(z^{-1}) \end{bmatrix}$$

and DET $B(z) = 4z^{-(N-1)}$. Now factor B(z) as

$$B(z) = \sqrt{2}\begin{bmatrix} H_{10}(z) & H_{11}(z) \\ H_{40}(z) & H_{41}(z) \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & z^{-1} \end{bmatrix} = \sqrt{2}\,C(z)\begin{bmatrix} 1 & 0 \\ 0 & z^{-1} \end{bmatrix}$$

where we define similarly to (1)

$$H_{40}(z) = -z^{-(N-1)}\,z\,H_{11}(z^{-1}) \text{ and } H_{41}(z) = z^{-(N-1)}\,z\,H_{10}(z^{-1}) \qquad (11)$$

so that $H_4(z) = H_{40}(z) + z^{-1}H_{41}(z)$ as $H_1(z) = H_{10}(z) + z^{-1}H_{11}(z)$. It follows that DET $C(z) = 2z^{-(N-2)}$ and that the *polyphase matrix* C(z) is only a function of z^{-2}. We thus have the representation of C(z) in terms of A(z)

$$C(z) = A(z)\frac{1}{2}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & z\end{bmatrix}$$

You can demonstrate these properties to yourself as follows: from the WAVE SOLVE directory, use D2G to compute a scaling vector, then from the WAVE directory create a wavelet matrix with QMF, then from the WAVE SYMB directory add z (or any symbol) to Level 1 of the stack and push LAURT to create a symbolic Laurent (polyphase) matrix which is C(z), and finally use **SDET** and **EXCO** to evaluate symbolically the determinant. For the g = 3 case, the result is

'.000000000001xz^-2 + 2xz^-4 + .000000000001xz^-6'

so DET C(z) = $2z^{-4}$. Application of the **SMPY** and **SEXCO** commands allow you to multiply and simplify the inverse transformation

$$C(z)\begin{bmatrix}1 & 0\\ 0 & z^{-1}\end{bmatrix}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} = A(z)$$

The result (rounded to just a few digits) is

$H_1(z)$ = .47 + 1.14z^-1 + .65z^-2 - .19z^-3 - .12x^-4 + .049x^-5
$H_1(-z)$ = .47 - 1.14z^-1 + .65z^-2 + .19z^-3 - .12x^-4 - .049x^-5
$H_4(z)$ = -.049 - .12z^-1 + .19z^-2 + .65z^-3 - 1.14x^-4 + .47x^-5
$H_4(-z)$ = -.049 + .12z^-1 + .19z^-2 - .65z^-3 - 1.14x^-4 - .47x^-5

The coefficients of these polynomials are the D_6 ones as expected.

After Vetterli [12], define $P(z) = H_1(z)H_4(-z)$. Then DET A(z) = P(z) - P(-z). Since DET A(z) = $-4z^{-(N-1)}$, P(z) can have only a single non-zero odd-indexed coefficient. For our above example

P(z) = '-.0234275 + .1953125xz^-2 - 6.E-13z^-3 - 1.171875xz^-4
- 2xz^-5 - 1.171875xz^-6 - 1.E-12xz^-7 + .1953125xz^-8
- .0234375xz^-10'

so the only non-zero odd-indexed coefficient is $-2z^{-5}$. P(z) satisfying this condition is called a *valid product polynomial.* In general, any factorization of a valid $P(z) = P_1(z)P_2(z)$ gives a possible FIR perfect reconstruction filter bank with $H_1(z) = P_1(z)$ and $H_4(z) = P_2(-z)$. Given a FIR filter $H_1(z)$, then any filter $H_5(z)$ such that $P(z) = H_1(z)H_5(-z)$ is a valid product polynomial is called a *complementary filter.*

Consider the filter bank defined by the equation

$$Y(z) = [G_1(z) \;\; G_4(z)]A(z)\begin{bmatrix} X(z) \\ X(-z) \end{bmatrix} = [G_1(z) \;\; G_4(z)]\begin{bmatrix} H_1(z) & H_1(-z) \\ H_4(z) & H_4(-z) \end{bmatrix}\begin{bmatrix} X(z) \\ X(-z) \end{bmatrix}$$

The output equals

$$\begin{aligned} Y(z) &= [G_1(z)H_1(z) + G_4(z)H_4(z)]X(z) \\ &\quad + [G_1(z)H_1(-z) + G_4(z)H_4(-z)]X(-z) \end{aligned}$$

If $G_1(z) = H_1(z^{-1})$ and $G_4(z) = H_4(z^{-1})$ then we have from (5)

$$Y(z) = 4X(z) + [H_1(z^{-1})H_1(-z) + H_4(z^{-1})H_4(-z)]X(-z) = 4X(z)$$

where we have used the identity

$$H_4(z^{-1})H_4(-z) = z^{N-1}H_1(-z)(-z)^{-(N-1)}H_1(z^{-1}) = -H_1(z^{-1})H_1(-z)$$

Thus, perfect reconstruction depends on a very delicate cancellation of terms which places constraints on the coefficients including that N in these equations be an even integer. One final relationship is

$$\begin{aligned} H_1(z)H_4(z^{-1}) &= H_{10}(z)H_{40}(z^{-1}) + H_{11}(z)H_{41}(z^{-1}) \\ &\quad + zH_{10}(z)H_{41}(z^{-1}) + z^{-1}H_{11}(z)H_{40}(z^{-1}) \\ &= -z^{N-1}z^{-1}H_{10}(z)H_{11}(z) + z^{N-1}z^{-1}H_{11}(z)H_{10}(z) \\ &\quad + z^{N-1}H_{10}(z)H_{10}(z) - z^{-(N-1)}z^{2}H_{11}(z)H_{11}(z) \\ &\quad + z^{N-1}\left\{[H_{10}(z)]^2 - z^{-2}[H_{11}(z)]^2\right\} \end{aligned}$$

so we have that $H_1(z)H_4(z^{-1})$ and $H_1(z^{-1})H_4(z)$ are polynomials in z^{-1} with no even terms. Consequently,

$$H_1(z)H_4(z^{-1}) + H_1(-z)H_4(-z^{-1}) = 0 \qquad (12)$$

These relations may be summarized by the equations

$$A^H(z^{-1})A(z) = A(z)A^H(z^{-1}) = 4I \quad \text{and} \quad C^H(z^{-1})C(z) = C(z)C^H(z^{-1}) = 2I$$

C(z) is a 2 × 2, genus g = N/2 example of the Laurent (polyphase) matrix P(z) discussed on page 567 of the MATHLIB manual.

In the past 15 pages, we have steamed through numerous relationships with numerical and symbolic examples so you will have them for reference. Now let me show you why it works. Using the QMF identities, we have

$$G_1(z) = H_1(z^{-1}) = -z^{N-1}H_4(-z) \quad \text{and} \quad G_4(z) = H_4(z^{-1}) = z^{N-1}H_1(-z)$$

Consequently, $G(z)A(z) = [G_1(z) \;\; G_4(z)]\, A(z) = [c \;\; 0]$ where c is given by

$$c = z^{N-1}[\,-H_4(-z)H_1(z) + H_1(-z)H_4(z)\,] = -z^{N-1}\text{DET } A(z) = 4$$

Hence, the paraunitary properties follow directly from the properties of the determinant of A(z). The synthesis filter G(z) is chosen such that, except for a possible delay times a constant, $G(z)H(z) = \text{DET } A(z) = P(z) - P(-z)$ where P(z) is a valid product polynomial.

ORTHOGONAL OR PARAUNITARY FILTER BANKS

The filter banks and associated wavelets we have studied so far are orthogonal ones. After Mallot [5] and Daubechies, the wavelet vector was chosen to be the QMF associated with the scaling vector. The definition of $H_4(z)$ was carefully chosen so the equations would explicitly show the associated delay and we would not have to make statements like "except for a phase factor." However, the results hold for N equal to any even integer such as N = 2 for most of Daubechies equations or N = 2g for our $H_4(z)$ definition. The filter bank orthogonality was proved on page 46 and the resulting scaling function and wavelet orthogonality was proved on page 54. After summarizing the properties of these filter banks, we will then generalize our results to biorthogonal filter banks where the perfect reconstruction property still holds, but we trade orthogonality for other desirable properties such as linear phase.

PROPERTY 1: The product of two paraunitary matrices is paraunitary.

PROPERTY 2: The determinant of a paraunitary matrix is a constant times a delay.

PROPERTY 3: The form of all filters which are complementary to $H_1(z)$ and give rise to a rank 2 paraunitary perfect reconstruction filter bank is $z^{-2k+1}\, H_1(-z^{-1})$ for some integer k.

PROPERTY 4: All of these results depend on N being even.

PROPERTY 5: The statement that A(z) or C(z) is paraunitary is equivalent to the statement that the wavelet coefficient matrix A is paraunitary. The paraunitary property of A(z) and C(z) follows from the orthogonality of the wavelet matrix by construction, see (4). Conversely, symbolically computing $C(z)C^H(z)$ and comparing coefficients of the same powers of z implies the paraunitary property of the wavelet coefficient matrix A. See Chapter 10 for details.

BIORTHOGONAL PERFECT RECONSTRUCTION BANKS

The above wavelet constructions have the disadvantage that they are inherently phase nonlinear. Consequently, let us examine more closely a few relations. Define $H_1(z)$ to be the Daubechies filter

$$H_1(z) = \left[\frac{1 + z^{-1}}{2}\right]^g Q_g(z)$$

Then

$$\begin{aligned} H_1(z)H_1(z^{-1}) &= \left[\frac{1 + z^{-1}}{2}\right]^g Q(z)_g Q_g(z^{-1}) \left[\frac{1 + z}{2}\right]^g \\ &= \left[\frac{1 + z}{2}\right]^{2g} z^{-g} Q_g(z)Q_g(z^{-1}) \\ &= (1 - y)^g P_g(y) \end{aligned}$$

where we have rewritten the definition of y in (8) as

$$y = -z^{-1}\left[\frac{1 - z}{2}\right]^2 \qquad 1 - y = z^{-1}\left[\frac{1 + z}{2}\right]^2$$

Similarly,

$$\begin{aligned} H_1(-z)H_1(-z^{-1}) &= \left[\frac{1 - z^{-1}}{2}\right]^g Q_g(-z)Q_g(-z^{-1}) \left[\frac{1 - z}{2}\right]^g \\ &= \left[\frac{1 - z}{2}\right]^{2g} (-z)^{-g} Q_g(-z)Q_g(-z^{-1}) \\ &= y^g P_g(1 - y) \end{aligned}$$

Using identities (5) and (6) we then have derived the Bezout equation (9) which is equivalent to (5). Thus, $H_1(z)$ is an asymmetric decomposition of the allpass requirement given by (5) which leads to phase nonlinear filters because all of the roots of $Q_g(z)$ are minimum phase.

We can generalize the above equations as follows. Define the filters

$$H_1(z) = \left[\frac{1 + z^{-1}}{2}\right]^{N_1} Q_1(z) z^{-n_1} \tag{13}$$

$$H_1(z) = \left[\frac{1 + z^{-1}}{2}\right]^{N_2} Q_2(z) z^{-n_2} \tag{14}$$

where $N_1 + N_2 = N$ which must be an even integer, $n_2 - n_1 = [N_1 - N_2]/2$ which is an integer, and $Q_1(z)$ and $Q_2(z)$ satisfy the identities

$$Q_1(z)Q_2(z^{-1}) = Q_1(z^{-1})Q_2(z) = P_{N/2}(y)$$

$$Q_1(-z)Q_2(-z^{-1}) = Q_1(-z^{-1})Q_2(-z) = P_{N/2}(1 - y)$$

Then we have that

$$\begin{aligned} H_1(z^{-1})H_1(z) &= \left[\frac{1 + z^{-1}}{2}\right]^{N_1} P_{N/2}(y) \left[\frac{1 + z}{2}\right]^{N_2} z^{n_2 - n_1} \\ &= \left[\frac{1 + z}{2}\right]^{N} P_{N/2}(y)\, z^{n_2 - n_1 - N_1} \\ &= (1 - y)^{N/2} P_{N/2}(y) \end{aligned}$$

$$\begin{aligned} H_1(-z^{-1})H_1(-z) &= \left[\frac{1 - z^{-1}}{2}\right]^{N_1} P_{N/2}(1 - y) \left[\frac{1 - z}{2}\right]^{N_2} (-z)^{n_2 - n_1} \\ &= \left[\frac{1 - z}{2}\right]^{N} P_{N/2}(1 - y)\, (-z)^{n_2 - n_1 - N_1} \\ &= y^{N/2} P_{N/2}(1 - y) \end{aligned}$$

and hence perfect reconstruction

$$H_1(z^{-1})H_1(z) + H_1(-z^{-1})H_1(-z) = 4 \tag{15}$$

Observe that $P(z) = H_1(z^{-1})H_1(z)$ is a valid product polynomial having only a single non-zero odd-indexed (even–indexed) coefficient (see page 75 for explanation). The phase linear constraint requires that each root of $P_{N/2}(y)$ be associated with either $Q_1(z)$ or $Q_2(z)$, that is, the double root in z defined by (10), must not be split between $Q_1(z)$ and $Q_2(z)$. Furthermore, the complex roots of $P_{N/2}(y)$ must not be split between $Q_1(z)$ and $Q_2(z)$, if you want real filter coefficients.

Consider the special case where $N_2 = 0$ and $Q_1(z) = 1$. Then we have

$$H_1(z) = \left[\frac{1 + z^{-1}}{2}\right]^{N} z^{N/2} \qquad H_1(z) = P_{N/2}(y)$$

The following program will test these filters to determine if they are valid product polynomials.

« → N « N 2 / POFZ N MOZ PMPY DUP QMFL PSUB » »

where N must be an even integer.

Vetterli [11], Cohen [1], Daubechies [2], and Feauveau [4], have studied the special case where $n_1 = 0$ and $Q_1(z) = 1$ defined by

$$H_1(z) = \left[\frac{1 + z^{-1}}{2}\right]^{N_1}$$

and its complementary filter

$$H_1(z) = \left[\frac{1 + z^{-1}}{2}\right]^{N_2} z^{-(N_1 - N_2)/2} P_{N/2}(y) = \left[\frac{1 + z}{2}\right]^{N_2} z^{-N/2} P_{N/2}(y)$$

and some more general cases. Program MOZ evaluates $H_1(z)$ and program MCOZ evaluates $H_1(z)$. Given these two lists, CLPM creates wavelet matrices corresponding to both lists. The following program computes the examples in the papers [1] and [2] and plots the results.

« → N1 N2 « N1 MOZ N1 N2 MCOZ CLPM → M1 M2 «
M1 1E-10 3 SOVN DUP 1 EROW PLT1 2 EROW PLT3 M2
1E-10 3 SOVN DUP 1 EROW PLT1 2 EROW PLT3 » » »

Remember that $N_1 + N_2 = N$ must be an even integer. If N1 = 1, then N2 = 1, 3, ... If N1 = 2, then N2 = 0, 2, ... For N1 = 2, you will find that the solution gets to the step where *COMPUTE λV FOR λ = 1* is displayed, but the solution never converges. This is because the eigenvector matrix is not full rank and SSOV can find no solution. When you tire of waiting carefully push once any key except ATTN. This will cause the program to HALT with a beep and the message *ENTER DIMENSION k λV AND PUSH CONT*, where k is some integer. You now have several options. Since you are in a HALT state, all the local variables are still defined and if you wish to verify that the dilation matrix T minus the identity matrix has a zero determinant, you can with the program sequence (T k IDN - DET DROP where DROP removes the determinant from the stack). However, to continue with the scaling and wavelet computations, you must now place a k dimensional vector on Level 1 of the stack without disturbing the other numbers sitting on the stack. Using a priori knowledge that the N1 = 2 case corresponds to a scaling function that is a hat function, a reasonable choice is a vector of the form [0 . . . 0 2 0 . . . 0] with dimension k. You can use the matrix editor to create this vector and place it on Level 1 of the stack. Then push CONT to continue the computation. The N1 = 2 case seems to be the only one with this problem.

Program CLPCF in the WAVE SOLVE directory allows you to study all of the cases defined by (13) and (14). Since the n_1 and n_2 are simply polynomial shifts, we can ignore them. The inputs to CLPCF (compute linear phase complementary filters) are N_1 and N_2 where $N_1 + N_2 = N$ must be even. After solving for the roots of $P_{N/2}(y)$, the program displays a temporary menu and HALTs with the root vector on Level 1 of the stack and the message "SPLIT P(y) ROOTS THEN PUSH CONT." When you are finished, the roots of $P_{N/2}(y)$ which you want associated with $Q_1(z)$ should be on Level 2 of the stack and those to be associated with $Q_2(z)$ on Level 1 of the stack.

Following the polynomial root vector conventions defined on page 193 of the MATHLIB manual, if either $Q_1(z)$ or $Q_2(z)$ equals 1, then the associated root vector is { }. The temporary menu gives you a key for this. Keys for the MATHLIB commands **SCOL, CORDR, IORDR,** and **CSPLT** allow you to easily rearrange the root vector so that the roots to be associated with $Q_1(z)$ can be placed at the left side of the row vector and those to be associated with $Q_2(z)$ placed at the right side of the row vector. **CSPLT** then splits the vector for you. By pushing ENTER to copy the root vector to Level 2 of the stack and using the **RND** command, you can more easily view the roots. Once the roots are split, push CONT to complete the computation. $H_1(z)$ is returned on Level 2 of the stack and $H_1(z)$ is returned to Level 1 of the stack.

As an example, we can compute the above example using MOZ and MCOZ by placing N1 and N2 on the stack and pushing the CLPCF key. When the program halts, push the { } key and then the SWAP key. Now push CONT and the same result (within a scale factor) will be computed.

If N_1 is odd (and thus N_2 is odd), then the resulting complementary filters have even length which may be the same length or lengths differing by some factor of 2. If N_1 is even, then both filters are of odd length differing by some factor of 2. The smaller filter can thus be zero filled by adding half of the zeros at the beginning of the list and half of the zeros at the end of the list as to create two phase linear polynomial lists of the same length. LPFIL will do this for you. The arguments of LPFIL are the two lists. LPFIL will make the smaller list larger.

Ignoring phase shifts, define the QMF $H_5(z) = H_1(-z)$ and $H_5(z) = H_1(-z)$ corresponding to $H_1(z)$ and $H_1(z)$, respectively. Then the analysis filter is defined as $H(z) = [H_1(z)\ H_5(z)]^T$ and the synthesis filter by $G(z) = [H_1(z^{-1})\ H_5(z^{-1})]$. Then using (15), the transfer function is

$$H_1(z^{-1})H_1(z) + H_5(z^{-1})H_5(z) = (1 - y)^{N/2}P_{N/2}(y) + y^{N/2}P_{N/2}(1 - y) = 4$$

The two complementary filter coefficients, after zero filing to the same length (see LPFIL), may be transformed into wavelet matrices with the program QMF. Program CLPM creates these two wavelet matrices for you. Program SOVN will use these matrices to compute the scaling and wavelet functions. See also program SOVJ discussed in Chapter 11.

When N_1 is odd, then the phase linear symmetry causes the two wavelet matrices to behave like a paraunitary pair since the inner product (resulting in a 2 × 2 matrix) is the identity matrix times a constant. However, while each wavelet matrix exhibits the usual orthogonality properties between the scaling and wavelet vectors, the two matrices are not mutually orthogonal. Furthermore, none of the vectors are orthogonal to their integer translations. Hence, the name *biorthogonal*. Since CLPM outputs two wavelet matrices, these may be called *dual bases* or *complementary bases*.

PROPERTIES OF FIR SOLUTIONS

We summarize a number of properties of FIR solutions which are proved in Vetterli [12]. Programs for continued fraction expansions and polynomial greatest common divisor are available in the MAN3 ALGB directory and are discussed in Chapter 7. The symbolic programs are in the MAN7 WAVE directory.

Vetterli's definition of *product polynomial* and *complementary filter* given on page 68, does not conveniently extend into the M–band and biorthogonal cases. Consequently, we introduce a new definition which is equivalent to his, but is less confusing as we generalize our results. Observe that $P(z) = H_1(z)H_4(-z) = H_1(z)H_1(z^{-1})z^{-(N-1)}$. Our definition is

$$P(z) = H_1(z)H_5(z^{-1}) \qquad \textbf{(16)}$$

for two filters $H_1(z)$ and $H_5(z)$. Observe that $P(z) = H_1(z)H_1(z^{-1})$ has a single non-zero *even-indexed* value, the coefficient of z^0. With this definition, the Bezout equation (15) can be written as

$$P(z) + P(-z) = 4$$

The below results make use of this definition.

PROPERTY 6: Linear phase perfect reconstruction FIR solutions have either of the following forms:
(a) Both filters are symmetric and of odd lengths, differing by an odd multiple of 2.
(b) One filter is symmetric and the other is antisymmetric, both lengths are even, and are equal or differ by an even multiple of 2.

PROPERTY 7: An even length linear phase filter has a unique same length complementary filter. An odd length N linear phase filter has a unique N – 2 length complementary filter.

Given a linear phase filter of length greater than 3, defined by its polynomial list L, program CMPL will compute the complementary filter. For example, if L = {.25 .75 .75 .25}, then CMPL(L) = {–.5 1.5 1.5 –.5} = C and PMPY(L, C) = {–.125 0 1.125 2 1.125 0 –.125} which is clearly a valid polynomial product. If L = {.125 .5 .75 .5 .125}, then CMPL(L) = {–1 4 –1} and the corresponding product polynomial is {–.125 0 1.125 2 1.125 0 –.125}.

Program MOZ is an easy way of creating a linear phase filter. It is known as the *binomial filter*. For this special case of linear phase filters, MCOZ also computes the complementary filter. In particular, MCOZ(3, 1) = {–.5 1.5 1.5 –.5} and MCOZ(4, 0) = {–1 4 –1} as above.

PROPERTY 8: There is always a complementary filter to the binomial filter.

The *involution* of the polynomial $H_5(z)$ is $H_5(z^{-1})$. Apart from a delay factor $z^{\pm k}$, the involution of any polynomial represented by the polynomial list {1 2 3 4 5} is simply the reversal of the coefficients, {5 4 3 2 1}.

$$H(z) = \sum_{n=0}^{N} a_n z^{-n}, \quad H(z^{-1}) = \sum_{n=0}^{N} a_n z^{n} = z^{N} \sum_{n=0}^{N} a_{N-n} z^{-n}$$

Thus, we compute involutions with the MATHLIB command **LREV** or when appropriate **VREV**. The above examples were phase linear wherein, apart from a delay, the involution of the filter was equal to the filter. Thus, no reversal was required. However, in the general case, we must perform it. See also the discussion on shifting the center of computation given on page 46.

PROPERTY 9: In the general case, a length N filter has N – 2 complementary filters of length N – 2.

Given an arbitrary filter of length N defined by the polynomial list L, program CMPG will compute any of the N – 2 complementary filters of length N – 2. The inputs are the list L and the position p = 1, 2, . . . N – 2. For example, CMPG{1 2 3 4 5}, 3) = {–1/3 2/3 –1/3} with corresponding product polynomial {–1/3 0 0 0 0 2 –4/3} while for p = 1, the complementary filter is {–10/3 8/3 –1/3} with product polynomial {–1/3 2 1 0 –1 0 –50/3} = PMPY({1 2 3 4 5}, {–1/3 8/3 –10/3}).

From polynomial theory (see Chapter 7), given polynomials a(z) and b(z), then

$$a(z)p(z) + b(z)q(z) = c(z)$$

has a solution p(z) and q(z) where the greatest common divisor of a(z) and b(z) divides c(z). Restating (5) and viewing the 4 as the c(z), we have that

$$H_1(z)H_1(z^{-1}) + H_4(z)H_4(z^{-1}) = 4$$

From definitions (1) and (11), it follows that:

PROPERTY 10: For $H_1(z)$ and $H_4(z)$ to form a perfect reconstruction pair, it is necessary and sufficient that any of the pairs $\{H_{10}(z), H_{11}(z)\}$, $\{H_{40}(z), H_{41}(z)\}$, $\{H_{10}(z), H_{40}(z)\}$, or $\{H_{11}(z), H_{41}(z)\}$ be coprime except for factors of z^{-n} for n = 0, 1, ...

For a coefficient list L which represents $H_1(z)$, zero fill on the right so that the size of L is even. Now use program LDLV in the MAN7 WAVE directory to create the polyphase filter lists representing $H_{10}(z)$ and $H_{11}(z)$. For example, from the MAN7 WAVE SOLVE directory, run the program « 4 MOZ 0 + LDLV » which computes the Level 2 list {.125 .75 .125} = a(z) and the Level 1 list {.5 .5 0} = b(z). Now push HOME MAN3 ALGB PRE (left-shifted NXT), and finally PGCDI to compute the greatest common divisor (GCD). The result is that GCD = {–.5} so that the polynomials are coprime. The q(z) polynomial is {–1.375 –1 –.125} and the p(z) polynomial is {1.5 .5}

PROPERTY 11: Given a linear phase filter $H_1(z)$ of odd length N, and its length N – 2 linear phase complement $H_5(z)$, all higher degree linear phase filters complementary to $H_1(z)$ are of the form

$$H_6(z) = z^{-2m}H_5(z) + E(z)H_1(-z) \qquad (17)$$

where

$$E(z) = \sum_{n=1}^{m} \alpha_n \left[z^{-2(n-1)} + z^{-(4m-2n)} \right] \tag{18}$$

Program CLPC computes linear phase complementary filters. The two inputs are the polynomial list L and the value of m. It calls program EOFM to compute E(z). For example CLPC({1 4 6 4 1}, 2) = { A1 '-(4×A1)' '6×A1+A2' '-(4×A1)-4×A2' '-.125+A1+7×A2' '.5-8×A2' '-.125+A1+7×A2' '-(4×A1)-4×A2' '6×A1+A2' '-(4×A1)' A1 }. The corresponding product polynomial is (use **SEXCO** to clean up the algebraic expressions)

{A1 0 '-(4×A1)+A2' 0 '-.125+6×A1-3×A2' 0 '1.125-3×A1+2×A2' 2 '1.125-3×A1+2×A2' 0 '-.125+6×A1-3×A2' 0 '-(4×A1)+A2' 0 A1}

Observe that the complementary filter holds for all values of A1 and A2 and that it produces a valid product polynomial since $P(z) + P(-z) = 4$. CLPC will compute complementary filters for an even number of coefficients, but the result is not phase linear.

PROPERTY 12: All filters $H_6(z)$ of length N + 2m - 2, which are complementary to a length N filter $H_1(z)$, have the form

$$H_6(z) = z^{-2k} H_5(z^{-1}) + R(z) H_1(-z) \tag{19}$$

where $R(z) = R(-z)$ is a polynomial of degree 2(m - 1) defined by

$$R(z) = \sum_{n=0}^{m-1} \beta_n z^{-2n} \tag{20}$$

k = 0, 1, . . . m, and $H_5(z)$ is a length N - 2 complementary filter.

Program CGC computes complementary filters in the general case. The three inputs are the polynomial list L representing $H_1(z)$, m = 1, 2, . . ., and k. It calls program ROFM to compute R(z). For example, CGC({1 2 3 4}, 2, 1) = { '-(4×B1)' '3×B1' '2-4×B0-2×B1' '-1+3×B0+B1' '-(2×B0)' B0 }. Remembering to reverse this polynomial with **LREV**, the corresponding product polynomial after cleanup with **SEXCO** is

{ B0 0 '-1+2×B0+B1' 0 '1-7×B0+2×B1' 2 '8-16×B0-7×B1' 0 '-(16×B1)' }

which is a valid product polynomial with $P(z) + P(-z) = 4$.

POLLEN PRODUCT AND THE PARAMETERIZATION OF COMPACTLY SUPPORTED WAVELETS

In the beginning of this chapter we carefully defined the filters $H_1(z)$ through $H_4(z)$. The paraunitary properties only work between $H_1(z)$ and $H_4(z)$, and not between $H_1(z)$ and $H_3(z)$. If you modify Pollen's papers [6] and [7] to use $H_4(z)$ instead of $H_3(z)$, then you obtain a beautiful parameterization of rank 2 compactly supported wavelets.

Define polyphase matrix $C_3(z) = C_1(z)h^{-1}C_2(z)$ where each of the $C_n(z)$ polyphase matrices has the form defined on page 67

$$C_n(z) = \begin{bmatrix} H_{10}^n(z) & H_{11}^n(z) \\ H_{40}^n(z) & H_{41}^n(z) \end{bmatrix} \qquad (21)$$

and h is a 2 x 2 Haar matrix which we define as

$$h = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad (22)$$

though other choices are possible. We note in this case that $h^{-1} = h/2$. Now the entries of $C_n(z)$ are defined by (1) and (11).

$$\begin{aligned} H_1^n(z) &= H_{10}^n(z) + zH_{11}^n(z) \\ H_4^n(z) &= H_{40}^n(z) + zH_{41}^n(z) \end{aligned} \qquad n = 1, 2, 3 \qquad (23)$$

We define the *Pollen product* of polynomials $H_1^1(z)$ and $H_1^2(z)$ to be $H_1^3(z)$ if and only if $C_3(z) = C_1(z)h^{-1}C_2(z)$. Thus, given $H_1^1(z)$ and $H_1^2(z)$, we compute the polyphase matrices $C_1(z)$ and $C_2(z)$, compute the matrix product $C_1(z)h^{-1}C_2(z)$, and finally extract $H_{10}^3(z)$ and $H_{11}^3(z)$ to compute $H_1^3(z)$. You can actually carry out this procedure symbolically using the program LAURT in the MAN7 WAVE SYMB directory. Its inputs are the wavelet matrix and some independent variable symbol z. A much faster way is to use the Pollen product programs PPRD1 and PPRD2 in the same directory.

The importance of the Pollen product lies in the fact that higher genus scaling vectors can be represented as Pollen products of lower genus scaling vectors. We used this fact on page 60 to symbolically compute a genus 3 coefficient vector from two genus 2 scaling vectors. Program SWD6 simply changes directories, calls PPRD1, and returns to the HAAR directory. The inputs to PPRD1 are the two scaling vector lists. It calls program QMF to convert the second list into a wavelet matrix and then calls PPRD2 for the solution. PPRD2 provides more generality. Pollen products are discussed more generally in the next chapter.

SPECIAL SYMBOLIC PROGRAMS

To compute the symbolic z transform, given a polynomial list, use the command **XEQN**. Program LAURT computes polyphase matrices given the wavelet matrix (which must be an HP array) and the independent variable. To generalize it to symbolic matrices, see the techniques in PPRD2. PTRN performs a non-Hermitian paraunitary transpose of a polyphase matrix. The inputs are the symbolic matrix and the independent variable to be inverted. Given an algebraic symbolic function F(z) and the symbol z, ZINV computes $F(z^{-1})$. SZINV does the same thing, except that the first input is a symbolic array. $Z \rightarrow \omega$ performs the $z = e^{i\omega}$ substitution for you. For more details, edit the programs or print them. The remaining programs in the SYMB directory are discussed on pages 47 and 48.

[1] Cohen, A., "Biorthogonal Wavelets," *Wavelets: A Tutorial in Theory and Applications*, Chui, C., ed., Cambridge, Academic Press, 1992.
[2] Cohen,A., Daubechies, I., and Feauveau, J., "Biorthogonal Bases of Compactly Supported Wavelets," *Comm. Pure and Appl. Math.*, 1991 to appear.
[3] Daubechies, I., Orthonormal Bases of Compactly Supported Wavelets, *Comm. Pure and Appl. Math.* 41, 1988, 909–996.
[4] Feauveau, J., "Nonorthogonal Multiresolution Analysis Using Wavelets," *Wavelets: A Tutorial in Theory and Applications*, Chui, C., ed., Cambridge, Academic Press, 1992.
[5] Mallat, S., "Multiresolution Approximation and Wavelet Orthonormal Bases of L^2," *Trans. Amer. Math. Soc.*, 11(7), 1989, 69–87.
[6] Pollen, D., "$SU_I(F[z,1/z])$ for F a Subfield of **C**," *J. Amer. Math. Soc.*, 1990.
[7] Pollen, D., "Parameterization of Compactly Supported Wavelets," *Aware Technical Report*, Aware Inc., Cambridge, MA, to appear.
[8] Smith, M., and Barnwell, T., "A New Filter Bank Theory for Time-Frequency Representation," *IEEE Trans. ASSP*, 35, No. 3, March 1987, 314–327.
[9] Strang, G., "Wavelets and Dilation Equations: A Brief Introduction," *SIAM Review*, 31, 1989, 614–627.
[10] Tewfik, A., Sinha, D., and Jorgensen, P., "On the Optimal Choice of a Wavelet for Signal Representation," *IEEE PGIT*, Vol 38, No. 2, March 1992, 747–765.
[11] Vetterli, M., "Wavelets and Filter Banks: Relationships and New Results," *Proc. IEEE ICASSP*, April 1990, 1723–1726.
[12] Vetterli, M., and Herley, C., "Wavelets and Filter Banks: Theory and Design," *IEEE ASSP*, 1992, to appear.

10

RANK M WAVELET MATRICES AND WAVELETS

INTRODUCTION

This chapter presents the M-band generalization of the 2-band wavelet software presented in the last chapter. Some of the first examples of perfect reconstruction filter banks were given by Smith and Barnwell [34]. Vaidyanathan then noticed the importance of the unitary property and reinterpreted results from classical scattering theory yielding factorization theorems for rational matrix functions, unitary on the unit circle. For a history see [44] and for excellent surveys see [45] and [50]. I have used the Koilpillai and Vaidyanathan cosine-modulated perfect reconstruction FIR filter bank software to design rank 64 genus 8 filters having over 100 dB subband isolation [25].

This chapter focuses on M-band (rank M or scale M or multiplicity M) Daubechies wavelet matrices and programs to compute them. While this class of filters is not optimum for all filtering problems, it is good for polynomial interpolation. Given the wavelet matrix, program SOVN discussed in Chapter 9, will compute the M-band scaling and wavelet functions. Program SOVJ discussed in Chapter 11 does also.

We begin by generalizing some of the concepts associated with perfect reconstruction filters. Then we more closely examine Daubechies' regularity condition. The computation of M–band wavelet matrices and wavelets is then discussed along with programs which compute them. Finally, we discuss the biorthogonal and other extensions.

RANK M GENERALIZATIONS

Pollen [29] has worked out an explicit formula for all Daubechies genus 2 scaling vector coefficient polynomials with rank m = 2, 3, ...

$$a(z) = \frac{(1 + z^{-1} + z^{-2} + \cdots + z^{-(m-1)})^2 (z^{-1} - r)}{m\,(1 - r)} \qquad (1)$$

based on Daubechies [4] where the root r is given by

$$r = \frac{m^2 + 2 \pm \sqrt{3(2m^2 + 1)}}{m^2 - 1}$$

Note that r is always real and $r_+ = 1/r_-$. Pollen [29], Linden [30], Heller [17], and Resnikoff [31] have worked out explicit formulas for the higher genus cases and in particular, Heller [18], Steffen [39], and Gopinath [12] have developed computational approaches for any rank and genus scaling vector with Daubechies regularity. This is implemented by program DMG in the MAN7 WAVE SOLVE directory.

Observe that the Daubechies maximum number of zeros at π (see page 61) translates in the higher rank case to a maximum number of zeros at $2\pi k/m$ for k = 1, 2, . . . m – 1. The factor

$$\mathcal{H}(z) = 1 + z^{-1} + z^{-2} + \cdots z^{-(m-1)} = \frac{1 - z^{-m}}{1 - z^{-1}} \qquad (2)$$

provides those zeros and the pole at $z^{-1} = 1$ cancels the zero at $z^{-1} = 1$ so that perfect reconstruction is possible. The generalization of (1) is of the form (Daubechies proves $\mathcal{H}^g(z)$ gives rise to regularity [4])

$$\mathcal{H}^g(z)[q_0 + q_1 z^{-1} + \cdots + q_{g-1} z^{-(g-1)}] = \mathcal{H}^g(z) Q_g(z) \qquad (3)$$

where $q_0 + q_1 + \ldots + q_{g-1} = 1$ so that $\mathcal{H}^g(1)Q_g(1) = m$. $\mathcal{H}^g(z)$ provides the Daubechies regularity and $Q_g(z)$ provides the perfect reconstruction filter bank paraunitary property. The $Q_g(z)$ coefficients are determined from the linear and quadratic wavelet matrix conditions. Daubechies regularity is equivalent to a maximally flat scaling vector design. In the orthogonal case, the scaling vector determines the wavelet vectors.

Consider a system of m FIR filters

$$H_s(z) = \sum_{k=0}^{mg-1} a_k^s \, z^{-k} \qquad s = 0, 1, \ldots m-1 \tag{4}$$

where the coefficients are elements of the m x mg wavelet matrix A

$$A = \begin{bmatrix} a_0^0 & a_1^0 & a_2^0 & \cdots & a_{gN-1}^0 \\ a_0^1 & a_1^1 & a_2^1 & \cdots & a_{gN-1}^1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_0^{N-1} & a_1^{N-1} & a_2^{N-1} & \cdots & a_{gN-1}^{N-1} \end{bmatrix} \tag{5}$$

Define the partial sums

$$P[A]_r^s(z) = \sum_{k=0}^{g-1} a_{r+mk}^s \, z^{-mk} \qquad r = 0, 1, \ldots m-1 \tag{6}$$

so that

$$H_s(z) = \sum_{r=0}^{m-1} P[A]_r^s(z) \, z^{-r} \qquad s = 0, 1, \ldots m-1 \tag{7}$$

Observe that this construction is simply the m generalization of definitions (9–1) and (9–11). We now form the M–band generalization of the polyphase matrix C(z) defined on page 67.

$$P[A](z) = \begin{bmatrix} P[A]_0^0(z) & & \cdots & & P[A]_{m-1}^0(z) \\ \vdots & \ddots & \vdots & & \vdots \\ & \cdots & P[A]_r^s(z) & \cdots & \\ \vdots & & \vdots & \ddots & \vdots \\ P[A]_0^{m-1}(z) & & \cdots & & P[A]_{m-1}^{m-1}(z) \end{bmatrix} \tag{8}$$

We require that the wavelet coefficients satisfy the linear constraint

$$\sum_{k=0}^{gm-1} a_k^s = m\delta_{s0} \qquad s = 0, 1, \ldots m-1 \tag{9}$$

and furthermore we will show that

$$P[A]_r^0(1) = 1 \qquad r = 0, 1, \ldots m-1 \tag{10}$$

The paraunitary condition $P[A](z)P[A]^H(z^{-1}) = mI$ where H denotes Hermitian transpose and I is the identity matrix is equivalent to the quadratic coefficient constraint

$$\sum_{k=-\infty}^{\infty} \bar{a}^{s}_{k+mr}\, a^{s'}_{k+mr'} = m\delta_{rr'}\delta_{ss'},\quad \bar{a} = a^{*},\ a^{s}_{k} = 0 \text{ for } k<0,\ k\geq mg. \tag{11}$$

We also have the paraunitary condition $P[A]^{H}(z^{-1})P[A](z) = mI$ which is equivalent to the quadratic constraint

$$\sum_{s=0}^{m-1}\sum_{r=-\infty}^{\infty} \bar{a}^{s}_{k+mr}\, a^{s}_{k'+mr+mn} = m\delta_{kk'}\delta_{n0} \tag{12}$$

Heller [18] and for the m = 2 case Strang [40] relate the paraunitary conditions to on an abstract symbol in order to derive conditions on the scaling vector. We derive those conditions by two alternative means. The first is as a modulated filter bank from which the existence of the $\mathcal{H}(z)$ filter is demonstrated. What we are exploiting in this derivation is the DFT convolution theorem as discussed in Appendix G of the MATHLIB manual and page 104 which leads to paraunitariness. The second means shows that we are not simply designing a DFT modulated bank. We obtain (11) and (12) as a result of these two derivations.

Given the definition of $H_s(z)$ in (4), compute the product

$$\begin{aligned} &H_s(ze^{i2\pi n/m})H^{*}_{s'}(z^{-1}e^{-i2\pi n/m}) \\ &\quad = \sum_{r=0}^{g-1}\sum_{r'=0}^{g-1}\sum_{k=0}^{m-1}\sum_{k'=0}^{g-1} \bar{a}^{s}_{rm+k}a^{s'}_{r'm+k'}z^{(r'-r)m+(k'-k)}e^{i2\pi n(k'-k)/m} \end{aligned} \tag{13}$$

Now only the exponential term depends on n and if we sum from n = 0 to m – 1, the sum equals m if k′ = k, and equals zero otherwise. Hence we have

$$\sum_{n=0}^{m-1} H_s(ze^{i2\pi n/m})H^{*}_{s'}(z^{-1}e^{-i2\pi n/m}) = m\sum_{r=0}^{g-1}\sum_{r'=0}^{g-1}\sum_{k=0}^{m-1} \bar{a}^{s}_{rm+k}a^{s'}_{r'm+k}z^{(r'-r)m} \tag{14}$$

Next define $\alpha = r' - r$ so $r' = r + \alpha$. Then we have

$$\sum_{n=0}^{m-1} H_s(ze^{i2\pi n/m})H^{*}_{s'}(z^{-1}e^{-i2\pi n/m}) = m\sum_{\alpha} z^{\alpha m}\sum_{r=0}^{g-1}\sum_{k=0}^{m-1} \bar{a}^{s}_{rm+k}a^{s'}_{rm+k+\alpha m} \tag{15}$$

Now if the left side of (15) equals $m^2\delta_{ss'}$, then clearly (11) must be true. Conversely, if (11) holds, then (15) must equal $m^2\delta_{ss'}$. Consequently, the wavelet matrix orthogonality condition (11) is directly related to a pseudo-QMF bank and (15) is the M–band generalization of (9–5) with (9–6), remembering that $|H_4(e^{i\omega})|^2 = |H_2(e^{i\omega})|^2$ as noted at the bottom of page 55. In the above construction we used the DFT kernel to exploit the convolution theorem. Taking the real part, yields a cosine modulated bank. Banks of this type were discussed by Smith and Barnwell [34] and are still used today [25].

$$\sum_{n=0}^{m-1} H_s(ze^{i2\pi n/m})H_{s'}^*(z^{-1}e^{-i2\pi n/m}) = m^2\delta_{ss'} \tag{16}$$

For s′ = s, (16) is a statement that each of the $H_s(z)$ is a spectral factor of an Mth band filter [27]. See the polyphase filter equations (38) and (39). A number of interesting relationships immediately follow. Linear condition (10) implies that $H_s(1) = m\delta_{s0}$. Consequently,

$$\sum_{n=0}^{m-1} |H_0(e^{i2\pi n/m})|^2 = |H_0(1)|^2 + \sum_{n=1}^{m-1} |H_0(e^{i2\pi n/m})|^2 = m^2 \tag{17}$$

so we have the result

$$H_0(e^{i2\pi n/m}) = 0 \qquad n = 1, 2, \ldots m-1 \tag{18}$$

which is a constraint on the scaling vector leading to the pseudo–QMF bank perfect reconstruction properties. This constraint implies that the scaling vector must contain a multiplicative factor of the form $\mathcal{H}(z)$. Define the partial sum

$$c_r = \sum_{p=0}^{g-1} a_{r+pm}^0$$

Then

$$\sum_{k=0}^{mg-1} a_k^0 e^{-i2\pi kn/m} = \sum_{r=0}^{m-1}\sum_{p=0}^{g-1} a_{r+pm}^0 e^{i2\pi rn/m}$$
$$= \sum_{r=0}^{m-1} c_r e^{-i2\pi rn/m} = m\delta_{n0}$$

which is simply the DFT of c_r having only a DC term. This proves that $c_r = 1$ for all r, which is stated by (10), and that the dilation matrix constructed by program SMAT will for all m again have the left eigenvector [1 . . . 1] corresponding to an eigenvalue of 1. Thus, SOVN will compute the scaling and wavelet functions for all ranks m.

Our second derivation exploits (12) and does not involve any modulation. Compute the sum

$$\sum_{s=0}^{m-1} H_s(z)H_s^*(z^{-1})$$
$$= \sum_{s=0}^{m-1}\sum_{r=0}^{g-1}\sum_{r'=0}^{g-1}\sum_{k=0}^{m-1}\sum_{k'=0}^{g-1} \bar{a}_{rm+k}^s a_{r'm+k'}^s z^{(r'-r)m+(k'-k)} \tag{19}$$

By (12), the sum over s on the right is zero unless k′ = k and r′ = r. Alternatively, for P[A](z) to be paraunitary, (19) must not depend on z so (12) must be true and we have

$$\sum_{s=0}^{m-1} H_s(z)H_s^*(z^{-1}) = \sum_{s=0}^{m-1}\sum_{r=0}^{g-1}\sum_{k=0}^{m-1} \bar{a}^s_{rm+k} a^s_{rm+k} = \sum_{k=0}^{m-1} m = m^2 \tag{20}$$

Observe that (20) is the generalization of (9–5) without (9–6). As in (17) we can write

$$\sum_{s=0}^{m-1} |H_s(1)|^2 = |H_0(1)|^2 + \sum_{s=1}^{m-1} |H_s(1)|^2 = m^2 \tag{21}$$

giving us the result we already knew

$$H_s(1) = 0 \qquad s = 1, 2, \ldots m-1 \tag{22}$$

that the sum of the elements in any wavelet vector is zero.

Comparing (16) and (17) with (20) and (21), we have a basic paraunitary equivalence. Summing the Fourier modulated filters across the bank is analogous to summing down the filter bank. Thus, we can *simulate* the entire filter bank using modulated versions of the scaling vector. Once the scaling vector is computed, then the wavelet vectors can be computed from the scaling vector. See (39) and (40).

DAUBECHIES' REGULARITY CONDITION

Daubechies's [2] original statement of her regularity condition was that the frequency response have an order g zero at $\omega = \pi$. Thus, the scaling vector filter must the form $H_1(z) = [1 + z^{-1}]^g Q_g(z)$. This condition ensures Holder continuity (smoothness) of the scaling function [4], [5]. Strang [40] observed that this was equivalent to perfect interpolation of polynomials having degree less than g. This is also equivalent to the g – 1 order vanishing moments of the wavelets. Again we modulate the bank to exploit the DFT convolution theorem and associated paraunitary property.

Let $A(\omega)$ be the magnitude squared of the Fourier transform of the scaling vector lowpass filter defined by

$$A(\omega) = |H_0(e^{i\omega})H_0^*(e^{-i\omega})|^2$$

Now (16) can be written as

$$\sum_{n=0}^{m-1} A(\omega + 2\pi n/m) = m^2 \tag{23}$$

Perfect interpolation of the monomial $x(t) = t^k$ for a given k means that the response of the bank to x(t) is the monomial t^k. From page 549 of the MATHLIB manual

$$\mathcal{F}[\ t^k\ ](\omega) = 2\pi\ i^k\ \delta^{(k)}(\omega)$$

where $\delta(\omega)$ is the Dirac delta function. From page 547 using the Fourier symmetry property

$$F(e^{i\omega})\delta^{(k)}(\omega) = \sum_{s=0}^{k} \binom{k}{s} F^{(s)}(1)\ \delta^{(k-s)}(\omega)$$

where F(z) denotes the z transform and $F(e^{i\omega})$ the associated Fourier transform. Using the equivalence concept discussed at the end of the last section, we multiply both sides of (23) by $\delta^{(k)}(\omega)$. Then

$$\sum_{s=0}^{k} \binom{k}{s} \sum_{n=0}^{m-1} A^{(s)}(2\pi n/m)\ \delta^{(k-s)}(\omega) = m^2\ \delta^{(k)}(\omega)$$

Now $A^{(s)}(2\pi n/m) = 0$ for $n = 1, \ldots m-1$ and $s = 0, \ldots g-1$ because it contains the factor $\mathcal{H}^g(e^{i\omega})\mathcal{H}^g(e^{-i\omega})$, so this reduces to

$$\sum_{s=0}^{k} \binom{k}{s} A^{(s)}(0)\ \delta^{(k-s)}(\omega) = m^2\ \delta^{(k)}(\omega)$$

Now in order for the two sides to be equal, we must have $A(0) = m^2$ and $A^{(s)}(\omega) = 0$ for $s = 1, \ldots g-1$. Consequently, we require that the scaling vector be maximally flat and satisfy (Strang [42], Heller [18], Zou [52], [54], and Gopinath [12])

$$A^{(s)}(e^{i2\pi n/m}) = 0 \quad n = 0, 1, \ldots m-1, \quad s = 1, 2, \ldots g-1 \tag{24}$$

Herrmann [24] published a closed form expression for rank 2 maximally flat filters in 1971. His formula, given by (9–7) and computed by POFY, is the same formula rediscovered by Daubechies [2] in 1988. We also observe that, as in the case of equation (17), the filter $\mathcal{H}^g(z)$ canonically satisfies the maximally flat constraint.

Observe that since (3) contains the factor $\mathcal{H}^g(z)$, the only constraints on $Q_g(z)$ which remain to be specified to achieve regularity and the paraunitary property are $A(0) = m^2$ and $A^{(s)}(0) = 0$ for $s = 1, \ldots g-1$. We will shortly see that $A^{(s)}(0) = 0$ does not imply that $H_0^{(s)}(0)$ is zero. In fact, regularity implies that it cannot be zero. This is explained on page 106.

Now consider the wavelet moments. In general we can write

$$\begin{aligned}
&\int_{-\infty}^{\infty} x^j w_s(x)\,dx \\
&= \int_{-\infty}^{\infty} \sum_{k=0}^{mg-1} \left(\frac{mx - k + k}{m}\right)^j a_k^s \int_{-\infty}^{\infty} s(mx - k)\,dx \\
&= m^{j-1} \sum_{r=0}^{j} \binom{j}{r} \sum_{k=0}^{mg-1} k^{j-r} a_k^s \int_{-\infty}^{\infty} (mx - k)^r\, s(mx - k)\, m\,dx \\
&= m^{-j-1} \sum_{r=0}^{j} \binom{j}{r} \sum_{k=0}^{mg-1} k^{j-r} a_k^s \int_{-\infty}^{\infty} x^r\, s(x)\,dx
\end{aligned} \tag{25}$$

Define the moments

$$\mu(s, j) = \int_{-\infty}^{\infty} x^j\, w_s(x)\,dx \tag{26}$$

$$M(s, j) = \sum_{k=0}^{mg-1} k^j\, a_k^s \tag{27}$$

where $w_0(x) = s(x)$. Then we can write

$$\mu(s, j) = m^{-j-1} \sum_{r=0}^{j} \binom{j}{r} M(s, j - r)\, \mu(0, r) \tag{28}$$

Now the argument goes like this. Suppose the monomials x^j for $j = 0, 1, \ldots g - 1$ are perfectly represented as a linear combination of the translates of the scaling function

$$x^j = \sum_{k=0}^{mg-1} c_{jk}\, s(x - k) \qquad j = 0, 1, \ldots g - 1 \tag{29}$$

where

$$c_{jk} = \int_{-\infty}^{\infty} x^j\, s(x - k)\,dx \tag{30}$$

Then in general, the moments $\mu(0, j) \neq 0$, but since the wavelets are orthogonal to $s(x)$ and thus to x^j, $\mu(s, j) = 0$ for $s = 1, 2, \ldots m - 1$ and $j = 0, 1, \ldots g - 1$. Thus $M(s, r) = 0$ for $r = 0, 1, \ldots g - 1$. Conversely, suppose that the $M(s, r)$ are all zero for $s = 1, 2, \ldots m - 1$. Then $\mu(s, j) = 0$ for all j and the response of the bandpass filters is zero. Since we have a perfect reconstruction filter bank which perfectly reproduces the monomials x^j for $j = 0, 1, \ldots g - 1$, the only filter left in the bank to produce that output is the lowpass section which is the scaling function.

This is the continuous wavelet argument. See Chapter 11 for examples and a nice demonstration. By forcing $\mathcal{H}^g(z)$ to be a factor in the scaling vector, we obtain $g - 1$ order of regularity. The result is a maximally flat filter. See also page 106.

COMPUTING RANK M SCALING VECTORS

Define the z transform of the scaling vector as

$$H_0(z) = \mathcal{H}^g(z)Q_g(z) \tag{31}$$

where $\mathcal{H}^g(z)$ and $Q_g(z)$ are defined by (2) and (3). Observe that

$$\mathcal{H}(z)\mathcal{H}(z^{-1}) = \frac{2 - z^m - z^{-m}}{2 - z - z^{-1}} = \frac{1 - \cos m\omega}{1 - \cos \omega} = \frac{\sin^2(m\omega/2)}{\sin^2(\omega/2)} \tag{32}$$

for $z = e^{i\omega}$. Noting that

$$\cos m\omega = 2^{m-1}\cos^m \omega + \sum_{k=1}^{IP[m/2]} (-1)^k \frac{m}{k}\binom{m-1-k}{k-1} 2^{m-2k-1}\cos^{m-2k}\omega \tag{33}$$

we can write

$$h(x) = \mathcal{H}(z)\mathcal{H}(z^{-1}) = \sum_{n=0}^{m-1} \alpha_n x^n \tag{34}$$

where $x = [z + z^{-1}]/2 = \cos \omega$ for $z = e^{i\omega}$. Similarly, define

$$r(x) = Q_g(z)Q_g(z^{-1}) \qquad \text{and} \qquad f(x) = h^g(x)r(x)m^{-2} \tag{35}$$

The constraints of the previous section can now be written as $f(1) = 1$, $f^{(s)}(1) = 0$ $s = 1, 2, \ldots g - 1$. This is because we achieve regularity by forcing $\mathcal{H}^g(z)$ to be a factor in the solution. Thus, all we need specify is that we want a paraunitary bank. Given these constraints and the fact that we know h(x), Heller [18] uses the following approach to solve for r(x). Since r(x) is a polynomial, it must satisfy its own Taylor expansion

$$r(x) = \sum_{k=0}^{g-1} \frac{r^{(k)}(1)}{k!} (x - 1)^k$$

about $x = 1$. But $r(x) = f(x)\,[h(x)]^{-g}$, so we have

$$r^{(s)}(1) = \sum_{p=0}^{s} \binom{s}{p} \left[\left(\frac{d}{dx}\right)^p f(x)\right]_{x=1} \left[\left(\frac{d}{dx}\right)^{s-p} [h(x)]^{-g}\right]_{x=1}$$

Applying the constraints reduces this equation to

$$r^{(s)}(1) = \left[\left(\frac{d}{dx}\right)^s [h(x)]^{-g}\right]_{x=1}$$

so r(x) equals

$$r(x) = \sum_{k=0}^{g-1} \left[\frac{1}{k!} \left(\frac{d}{dx} \right)^k [h(x)]^{-g} \right]_{x=1} (x - 1)^k = \sum_{k=0}^{g-1} \beta_k (x - 1)^k \quad (36)$$

Now define x = 1 – 2y which is the same definition as (9–8). Then we have

$$r(x) = \sum_{k=0}^{g-1} \beta_k (-2y)^k = P_g^m(y) \quad (37)$$

where $P_g^m(y)$ is the M–band generalization of $P_g(y)$ defined in (9–7) and is computed by program POMY in the MAN7 WAVE SOLVE directory. Program DMG in the same directory computes the scaling vector by first calling POMY to get the polynomial $P_g^m(y)$. Then it calls QOFZ to compute the minimum phase $Q_g(z)$. Then it calls MOMZ to get the $\mathfrak{H}^g(z)$ polynomial, multiplies, and normalizes.

The POMY computation is a little involved. It uses the advanced derivative techniques discussed on page 40. The required derivatives for up to genus 6 are stored in subdirectory COPT of the MAN7 WAVE SOLVE directory. To compute higher genus solutions, you will need to compute the symbolic equations for D6, D7, . . . as explained in Chapter 7 and store them in the SOLVE subdirectory COPT. The formulas require thousands of bytes and there is no room on the ROM. Programs DERV1 and DERV2 will do these computations for you, but plan ahead. The symbolic calculations require several hours on the HP48, but once completed and stored in COPT, they evaluate in seconds.

POMY begins by computing (33) and (32). Then it tabulates the derivatives of h(x) which it stores in a variable DDATA after entering the COPT subdirectory. The equations D1, D2, . . . are formulas for $[d(x)]^{-N-1}$ derivatives where N = g – 1 and d(x) is the h(x) polynomial represented by its polynomial list. When it finishes, r(x) has been computed, DDATA purged, and the program returns to the SOLVE directory. Finally, it does the x – 1 = –2y substitution.

POMY, QOFZ, and MOMZ are internal programs used by DMG, which is the M–band generalization of D2G. Given the rank m and the genus g, DMG computes the Daubechies scaling vector. Examples are given after we discuss the computation of the wavelet vectors.

In Chapter 9 we showed for the rank 2 case that $H_0(z)$ defined by (31) satisfied the Bezout equation (see page 71). Equation (16) with s′ = s = 0 is the M–band generalization discussed below.

ALGEBRAIC STRUCTURE OF WAVELET MATRICES

On page 86 we introduced a notion of equivalent sums. Let us explore it in more detail. Let $G_r^s(z) = P[A]_r^s(z)$ with definitions (4), (6), and (7). Then (16) can be written as

$$\begin{aligned}\sum_{n=0}^{m-1} H_s(ze^{i2\pi n/m})H_{s'}^*(z^{-1}e^{-i2\pi n/m}) &= m^2\delta_{ss'} \\ &= \sum_{n=0}^{m-1}\sum_{r=0}^{m-1}\sum_{r'=0}^{m-1} G_r^s(z)\bar{G}_{r'}^{s'}(z^{-1})z^{r'-r}e^{i2\pi n(r'-r)/m} \\ &= m\sum_{r=0}^{m-1} G_r^s(z)\bar{G}_r^{s'}(z^{-1})\end{aligned} \tag{38}$$

from which we get the relation

$$\sum_{r=0}^{m-1} G_r^s(z)\bar{G}_r^{s'}(z^{-1}) = m\delta_{ss'} \tag{39}$$

The independence of n in the argument of $G_r^s(z)$ follows from its polyphase definition (6). We similarly get from (20) that

$$\sum_{s=0}^{m-1} G_r^s(z)\bar{G}_{r'}^s(z^{-1}) = m\delta_{rr'} \tag{40}$$

While (39) is the sum across the rows of P[A](z), (40) is the sum down the columns of P[A](z). By the paraunitary property this comes as no surprise, but I wanted to make these relations explicit. From the paraunitary viewpoint, $P[A](z)P[A]^H(z^{-1}) = P[A]^H(z^{-1})P[A](z) = mI$. However, from the linear viewpoint, P[A](z) and $P[A]^H(z)$ are very different since

$$\sum_{r=0}^{m-1} P[A]_r^s(1) = m\delta_{s0} \tag{41}$$

but the same sum on $P[A]^H(z)$ is just m numbers with the sole property that their sum equals m. The above relations imply a significant amount of algebraic structure must exist. Heller, et al, have worked out many of the details and we summarize some of their results [21].

Define:

SV(m, g) = set of all scaling vectors of rank m and genus g.

WM(m, g) = set of all wavelet matrices of rank m and genus g.

H(m) = WM(m, 1) = Haar wavelet matrices.

As quoted on page 569 of the MATHLIB manual, an m × m complex matrix h is a Haar matrix if and only if it can be represented as†

$$h = \begin{bmatrix} 1 & 0 \\ 0 & U \end{bmatrix} \mathbf{h} = \boldsymbol{U}\mathbf{h} \tag{42}$$

where **h** is the canonical Haar wavelet matrix shown on page 569 and computed by program CHAAR in the MAN7 WAVE HAAR directory, and U is a unitary matrix. See the MATHLIB manual for examples.

In general, for any wavelet matrix $a \in WM(m, g)$, there is a well defined mapping from WM(m, g) to H(m) which is the definition of the polyphase matrix P[a](1). P[a](1) is called the *characteristic Haar wavelet matrix*. Observe that the map $a \to P[a](1)$ commutes with left matrix multiplication by $\boldsymbol{U}$ and that $\boldsymbol{U}$ is invertible.

Consequently, for $a \in WM(m, g)$, define the map

$$\chi(a) = P[a](1) = h$$

Then we have

$$WM(m, g) \xrightarrow{\chi} H(m)$$

Define the inverse mapping to be $WM_h(m, g) = \chi^{-1}(h)$. Then

$$WM(m, g) = \bigcup_{h \in H(m)} WM_h(m, g)$$

is a disjoint union. Define the set

$$WG_h(m, g) = \bigcup_{1 \le g} WM_h(m, g)$$

which we call the *wavelet group of rank m at the Haar wavelet matrix* h. The group operation is the *Pollen product* discussed on page 79. Let $a' \in WG_h(m, g')$ and $a'' \in WG_h(m, g'')$ where $h \in H(m)$. Then $a \in WM(m, g' + g'' - 1)$ is the Pollen product of a′ and a″ if and only if

$$P[a](z) = P[a'](z)h^{-1}P[a''](z)$$

From the properties of matrix products, the Pollen product is noncommutative, associative, P[a](z) must be paraunitary, and therefore an inverse must exist. In fact, the unit element of $WG_h(m, g)$ is h.

As an example, consider the Pollen product of

† Gopinath has an equivalent state space approach to characterizing M–band wavelet matrices [12]. Linear condition (9) is the reason **h** is not simply a unitary matrix. See page 126.

$$a' = \begin{bmatrix} a_0' & a_1' & a_2' & a_3' & a_4' & a_5' \\ b_0' & b_1' & b_2' & b_3' & b_4' & b_5' \end{bmatrix}$$

with corresponding Laurent (polyphase) matrix

$$P[a'](z) = \begin{bmatrix} a_0' + a_2' z^{-2} + a_4' z^{-4} & a_1' + a_3' z^{-2} + a_4' z^{-4} \\ b_0' + b_2' z^{-2} + b_4' z^{-4} & b_1' + b_3' z^{-2} + b_5' z^{-4} \end{bmatrix}$$

and

$$a'' = \begin{bmatrix} a_0'' & a_1'' & a_2'' & a_3'' \\ b_0'' & b_1'' & b_2'' & b_3'' \end{bmatrix}$$

PPRD2({AP0 AP1 AP2 AP3 AP4 AP5}, {{APP0 APP1 APP2 APP3} {BPP0 BPP1 BPP2 BPP3}}) will compute this pollen product for you. PPRD2 is in the MAN7 WAVE SYMB directory and is limited to rank 2 Pollen products. The result is a genus 4 coefficient system with 8 coefficients. The first and last equal

$$a_0 = \frac{1}{2}\left[a_0' a_0'' + a_0' b_0'' + a_1' a_0'' - a_1' b_0''\right]$$

$$a_7 = \frac{1}{2}\left[a_4' a_3'' + a_4' b_3'' + a_5' a_3'' - a_5' b_3''\right]$$

The same computation can be repeated with the wavelet coefficients of a′, the second row of a′, to obtain those eight coefficients.

Finally, if $a' \in H(m')$ and $a'' \in H(m'')$, then the tensor product of a′ and a″ is in H(m′m″). Tensor (Kronecker) products can be performed with program KPRD in the HAAR directory. If $a \in H(m)$, then the determinant of a equals m^m.

What makes the algebraic structure of wavelet matrices important is the it provides a way to compute the wavelet vectors from the scaling vector. The very clever part of Heller's solution technique [21] is his exploitation of the above relationships between a wavelet matrix and its associated Haar wavelet matrix. This allows him to only solve linear equations since m of the orthogonality and m of the linear conditions are built into this relationship. You will see this in the proof below.

COMPUTING RANK M WAVELET VECTORS

After Heller [21], define the m × mg wavelet matrix as

$$a = \begin{bmatrix} \alpha \\ \beta^1 \\ \beta^2 \\ \vdots \\ \beta^{m-1} \end{bmatrix} = \begin{bmatrix} \beta^0 \\ \beta^1 \\ \beta^2 \\ \vdots \\ \beta^{m-1} \end{bmatrix} \tag{43}$$

and similarly define the associated Haar wavelet matrix as

$$h = \begin{bmatrix} h^0 \\ \vdots \\ h^{m-1} \end{bmatrix} \tag{44}$$

Now α is the scaling vector

$$\alpha = [a_0 \quad a_1 \quad \ldots \quad a_{mg-1}] = \beta^0 = [b_0^0 \quad b_1^0 \quad \ldots \quad b_{mg-1}^0] \tag{45}$$

which can be broken up into g subblocks

$$\alpha_k = [a_{mk} \quad a_{mk+1} \quad \ldots \quad a_{mk+g-1}] \qquad k = 0, 1, \ldots g - 1 \tag{46}$$

The wavelet vectors can be broken up similarly

$$\beta_k^j = [b_{mk}^j \quad b_{mk+1}^j \quad \ldots \quad b_{mk+g-1}^j] \qquad k = 0, 1, \ldots g - 1 \tag{47}$$

for j = 1, 2, . . . m − 1. Define the right and left shift operators

$$L : \mathbb{C}^{mg} \to \mathbb{C}^{mg} \qquad R : \mathbb{C}^{mg} \to \mathbb{C}^{mg} \tag{48}$$

by

$$L^k\alpha = [a_{mk} \quad a_{mk+1} \quad \ldots \quad a_{mg-1} \quad 0 \quad \ldots \quad 0] \tag{49}$$

and

$$R^k\alpha = [0 \quad \ldots \quad 0 \quad a_0 \quad a_1 \quad \ldots \quad a_{m(g-k)-1}] \tag{50}$$

for k = 0, 1, . . . g − 1. We say that a scaling vector is *nondegenerate* if neither α_0 and α_{g-1} is the zero vector. Using $\langle \bullet, \bullet \rangle$ to denote dot product, the orthogonality conditions (11) can now be written as

$$\begin{aligned} &\langle L^j\alpha, \alpha \rangle = m\delta_{j0} && j = 0, \ldots g - 1 \\ &\langle L^j\alpha, \beta^s \rangle = 0 && j = 0, \ldots g - 1, \quad s = 0, \ldots m - 1 \\ &\langle L^j\beta^s, \alpha \rangle = 0 && j = 0, \ldots g - 1, \quad s = 0, \ldots m - 1 \\ &\langle L^j\beta^s, \beta^{s'} \rangle = m\delta_{ss'} && j = 0, \ldots g - 1, \quad s = 0, \ldots m - 1 \end{aligned} \tag{51}$$

Observe that

$$\alpha_k = R^{g-1}L^k\alpha$$

displays each α_k as the rightmost m elements of an mg length vector while

$$\alpha_k = L^{g-1}R^{g-1-k}\alpha$$

displays each α_k as the leftmost m elements of an mg length vector. We say that the wavelet vectors are nondegenerate if

$$R^{g-1}\beta^j \neq 0 \qquad j = 1, \ldots g - 1 \tag{52}$$

A wavelet matrix is nondegenerate if its scaling vector and wavelet vectors are all nondegenerate.

Let $\alpha \in SV(m, g)$ be nondegenerate. Then the vectors

$$\{\alpha, L\alpha, \ldots, L^{g-1}\alpha\}$$

and the vectors

$$\{\alpha, R\alpha, \ldots, R^{g-1}\alpha\}$$

are linearly independent vectors. Furthermore, if $\beta^j \in WM(m,g)$ is also nondegenerate, then

$$\{\beta^j, R\beta^j, \ldots, R^{g-1}\beta^j\}$$

are linearly independent vectors. Suppose we choose the left shifted α vectors to form the vector space V_α

$$V_\alpha = \left\{ \sum_{k=0}^{g-1} c_k L^k \alpha \right\} \tag{53}$$

and denote the orthogonal complement of V_α by $V_\alpha^\perp$, that is

$$V_\alpha^\perp = \{y \in \mathbf{C}^{mg} : \langle y, x\rangle = 0 \text{ for all } x \in V_\alpha\}$$

Since

$$\langle L^k\alpha,\ R^{k'}\alpha\rangle = \langle L^{k+k'}\alpha,\ \alpha\rangle = 0 \qquad k' > 0 \tag{54}$$

the right shifted vectors $R^k\alpha \in V_\alpha^\perp$ for $k > 0$ and thus can be used as a basis for the wavelet vectors.† Thus, let

$$\beta^j = \sum_{l=0}^{g-2} c_l^j\, R^l\alpha + E^j \tag{55}$$

where $E^j = [0\ \ldots\ 0\ E_0^j\ E_1^j\ \ldots\ E_{m-1}^j]$ is an mg length vector independent of $R^k\alpha$ so that the Haar wavelet matrix constraint

$$\sum_{k=0}^{g-1} \beta_k^j = \sum_{k=0}^{g-1} R^{g-1} L^k \beta^j = h^j \tag{56}$$

may also be satisfied. Substitution of (55) into (56) gives m inhomogeneous linear equations in the variables c_l^j and E_n^j

$$\sum_{l=0}^{g-2} c_l^j \left[\sum_{k=0}^{g-1} R^{g-1} L^k R^l\alpha\right] + E^j = h^j \tag{57}$$

From (51) we also require that

$$\langle L^k \beta^j,\ \alpha\rangle = \langle \beta^j,\ R^k\alpha\rangle = 0 \tag{58}$$

which gives g – 1 additional linear equations

$$\sum_{l=0}^{g-2} c_l^j \langle R^l\ \alpha,\ R^k\alpha\rangle + \langle E^j,\ R^k\ \alpha\rangle = 0 \qquad 0 \le k \le g-1 \tag{59}$$

Equation (59) for $k = g - 1$ actually follows from the previous g – 2 equations and the Haar condition and is not an independent equation. Consequently, we use the m equations (57) and the g – 1 equations (59) for $k = 0, 1, \ldots g - 2$ to form a $(m + g - 1) \times (m + g - 1)$ linear system.

Program CWM in the MAN7 WAVE SOLVE directory uses this technique to compute M-band wavelet matrices. We give examples below. The proof of Theorem 4.8 in [21] gives much insight into the nature of the solution. We repeat it here adding a few details.

† Heller has also done the alternative construction where $R^k\alpha \in V_\alpha$ and $L^k\alpha \in V_\alpha^\perp$ for $k > 0$. It has the form

$$\beta^j = \sum_{l=0}^{g-2} c_l^j\, L^l\alpha + E^j \tag{60}$$

where $E^j = [E_0^j\ \ E_1^j\ \ \ldots\ \ E_{m-1}^j\ \ 0\ \ \ldots\ \ 0]$ is an mg length vector and gives rise to a very different set of wavelets.

THEOREM *Let the genus $g \geq 1$. If $\{c_0^j, \ldots c_{g-2}^j, E_0^j, \ldots E_{m-1}^j\}$ is a solution to the equations (57) and (59), then the wavelet system given by (55) satisfies the orthogonality conditions (51) as well as the Haar condition (56). Moreover, each wavelet vector β^j is nondegenerate if and only if $c_0^j \neq 0$.*

Proof:

(i): The Haar condition (56) is satisfied by construction as is the orthogonality condition (51)

$$\sum_{l=0}^{g-2} c_l^j \langle R^l, R^k \alpha \rangle + \langle E^j, R^k \alpha \rangle = 0 \qquad 0 \leq k \leq g-2 \tag{61}$$

(ii): We need to show that (59) holds for $k = g - 1$. Thus,

$$\langle L^{g-1} \beta^j, \alpha \rangle = \langle \beta^j, R^{g-1} \alpha \rangle = \sum_{l=0}^{g-2} c_l^j \langle R^l \alpha, R^{g-1} \alpha \rangle + \langle E^j, R^{g-1} \alpha \rangle \tag{62}$$

must equal zero. The Haar condition (56) can be rewritten as

$$\begin{aligned} h^j &= \sum_{k=0}^{g-1} L^{g-1} R^k \beta^j \\ &= \sum_{l=0}^{g-2} c_l^j \left[\sum_{k=0}^{g-1} L^{g-1} R^{k+l} \alpha \right] + L^{g-1} E^j \end{aligned} \tag{63}$$

The linear constraint (10) on the scaling vector α gives

$$1 = \sum_{n=0}^{g-1} L^{g-1} R^n \alpha \tag{64}$$

where 1 is the m dimensional unit row vector [1 . . . 1]. The orthogonality of the Haar rows implies $0 = \langle 1, h^j \rangle$ for $j = 1, 2, \ldots m - 1$ so we have

$$\left\langle \sum_{n=0}^{g-1} L^{g-1} R^n \alpha, \sum_{l=0}^{g-2} c_l^j \left[\sum_{k=0}^{g-1} L^{g-1} R^{k+l} \alpha \right] + L^{g-1} E^j \right\rangle = 0 \tag{65}$$

The second term equals

$$\begin{aligned} \left\langle \sum_{n=0}^{g-1} L^{g-1} R^n \alpha, L^{g-1} E^j \right\rangle &= \sum_{n=0}^{g-1} \langle R^n \alpha, E^j \rangle \\ &= \langle R^{g-1} \alpha, E^j \rangle + \sum_{n=0}^{g-2} \langle R^n \alpha, E^j \rangle \end{aligned} \tag{66}$$

By construction, this equals using (51) or (61)

$$\langle R^{g-1}\,\alpha,\, E^{j}\rangle - \sum_{l=0}^{g-2} c_l^{j} \sum_{n=0}^{g-2} \langle R^{n}\,\alpha,\, R^{l}\,\alpha\rangle \tag{67}$$

The first term of (65) can be rewritten as

$$\sum_{l=0}^{g-2} c_l^{j} \left\langle \sum_{n=0}^{g-1} L^{g-1} R^{n}\,\alpha,\, \sum_{k=0}^{g-1} L^{g-1} R^{k+l}\,\alpha \right\rangle \tag{68}$$

For any integers a and b, the following identity holds

$$\begin{aligned} \langle R^{a}\,\alpha,\, R^{b}\,\alpha\rangle &= \sum_{k=0}^{g-1-\mathrm{MAX}(a,b)} \langle \alpha_{k+a},\, \alpha_{k+b}\rangle \\ &= \sum_{k=0}^{g-1} \langle L^{g-1} R^{a+k}\,\alpha,\, L^{g-1} R^{b+k}\,\alpha\rangle \end{aligned} \tag{69}$$

The sums in the dot product of (68) are over a rectangle $0 \le n \le g-1$ and $0 \le k \le g-1-l$. Keeping in mind that $R^{k}\alpha = 0$ for $k > g-1$, we can rewrite this double sum in two parts. If n is the X axis and k is the Y axis of the rectangle, and we sum along diagonals, then the sum

$$\sum_{n=0}^{g-1} \sum_{r=0}^{g-1} \langle L^{g-1} R^{n+r}\,\alpha,\, L^{g-1} R^{l+r}\,\alpha\rangle \tag{70}$$

includes all terms on the diagonal from (0, 0) and also all terms below and to the right of the diagonal. Note that the upper limit of the n summation is g – 1, not g – 2. The terms above the diagonal are

$$\sum_{s=0}^{g-1} \sum_{r=0}^{g-1} \langle L^{g-1} R^{r}\,\alpha,\, L^{g-1} R^{l+r+s}\,\alpha\rangle \tag{71}$$

Applying identity (69) to both (70) and (71), then adding the results, the dot product in (68) is

$$\sum_{n=0}^{g-1} \langle R^{n}\,\alpha,\, R^{l}\,\alpha\rangle + \sum_{s=0}^{g-1} \langle \alpha,\, R^{l+s}\,\alpha\rangle \tag{72}$$

The second term of (72) is zero by (54), so we can write (68) as

$$\begin{aligned} &\sum_{l=0}^{g-2} c_l^{j} \sum_{n=0}^{g-1} \langle R^{n}\,\alpha,\, R^{l}\,\alpha\rangle \\ &\quad = \sum_{l=0}^{g-2} c_l^{j} \langle R^{g-1}\,\alpha,\, R^{l}\,\alpha\rangle + \sum_{l=0}^{g-2} c_l^{j} \sum_{n=0}^{g-2} \langle R^{n}\,\alpha,\, R^{l}\,\alpha\rangle \end{aligned} \tag{73}$$

Adding (73) to (67), gives the desired result that (65) equals

$$\langle R^{g-1}\,\alpha,\, E^{j}\rangle + \sum_{l=0}^{g-2} c_l^{j} \langle R^{g-1}\,\alpha,\, R^{l}\,\alpha\rangle = \langle R^{g-1}\,\alpha,\, \beta^{j}\rangle = 0 \tag{74}$$

which is what we needed to prove.

(iii): Parts (i) and (ii) showed that the right shifted and unshifted scaling vector α is orthogonal to β

$$\langle R^k \alpha, \beta^j \rangle = 0 \qquad k = 0, \ldots g-1, \quad j = 1, \ldots m-1 \tag{75}$$

Now we need to show that the left shifted scaling vector α is also orthogonal to β. We have

$$\langle L^k \alpha, \beta^j \rangle = \langle L^{k-1} \alpha, R \beta^j \rangle \qquad k > 0 \tag{76}$$

By (55)

$$R\beta^j = \sum_{l=0}^{g-2} c_l^j R^{l+1} \alpha \tag{77}$$

since $R E^j = 0$. Consequently, for $k = 1, \ldots g-1$, and $j = 1, \ldots m-1$

$$\begin{aligned} \langle L^{k-1} \alpha, R\beta^j \rangle &= \sum_{l=0}^{g-2} c_l^j \langle L^{k-1} \alpha, R^{l+1} \alpha \rangle \\ &= \sum_{l=0}^{g-2} c_l^j \langle \alpha, R^{k+l} \alpha \rangle = 0 \end{aligned} \tag{78}$$

(iv): Next we want to show for $j = 1, \ldots m-1$ that the right shifted wavelet vector $R^s \beta^j$ is orthogonal to all wavelet vectors $\beta^{j'}$. For $s > 0$ we have

$$\langle R^s \beta^j, \beta^{j'} \rangle = \sum_{l=0}^{g-2} c_l^j \langle R^{l+s} \alpha, \beta^{j'} \rangle = 0 \qquad s > 0 \tag{79}$$

by parts (i) and (ii).

(v): By the techniques of part (ii) and the results of (iv) we have

$$\begin{aligned} \langle h^j, h^{j'} \rangle &= \langle \sum_{k=0}^{g-1} L^{g-1} R^k \beta^j, \sum_{k'=0}^{g-1} L^{g-1} R^{k'} \beta^{j'} \rangle \\ &= \sum_{k=0}^{g-1} \sum_{r=0}^{g-1} \langle L^{g-1} R^{k+r} \beta^j, L^{g-1} R^r \beta^{j'} \rangle \\ &\quad + \sum_{s=1}^{g-1} \sum_{r=0}^{g-1} \langle L^{g-1} R^r \beta^j, L^{g-1} R^{s+r} \beta^{j'} \rangle \\ &= \sum_{k=0}^{g-1} \langle R^k \beta^j, \beta^{j'} \rangle + \sum_{s=1}^{g-1} \langle \beta^j, R^s \beta^{j'} \rangle \\ &= \langle \beta^j, \beta^{j'} \rangle + \sum_{s=1}^{g-1} [\langle R^s \beta^j, \beta^{j'} \rangle + \langle \beta^j, R^s \beta^{j'} \rangle] \\ &= \langle \beta^j, \beta^{j'} \rangle = m \delta_{jj'} \end{aligned} \tag{80}$$

Observe that it is through the exploitation of the relationship between the wavelet matrix and the Haar wavelet matrix, that the quadratic conditions are implied by the linear ones.

(vi): Finally, we want to show that the wavelet vectors β^j are nondegenerate if and only if $c_0^j \neq 0$. From (55) this immediately follows

$$R^{g-1}\,\beta^j = c_0^j\,R^{g-1}\,\alpha \qquad \textbf{(81)}$$

since α is assumed to be nondegenerate. QED

COMPUTING RANK M WAVELETS

Program CWM computes rank M wavelet matrices using the technique presented in the previous section. It has four inputs.

4: SCALING VECTOR FOR m AND g
3: RANK m
2: GENUS g
1: m × m HAAR WAVELET MATRIX

CWM is in the MAN7 WAVE SOLVE directory. For your convenience COSM has been provided in the menu for computing a DCT Haar wavelet matrix. The program simply goes over to the HAAR directory to get the matrix so if you rearrange the WAVE directory, it may no longer work. For other Haar wavelet matrices, simply edit COSM.

Once the wavelet matrix is computed, you can compute the scaling function and M–band wavelets using SOVN, which assumes that the number of rows of the wavelet matrix equals the rank and the number of columns of the wavelet matrix is mg. The following example program computes and plots the scaling and wavelet functions.

```
< → m g n < m g DMG m g m COSM CWM 1E-10 n
SOVN → M < 1 M SIZE 1 GET FOR J M J EROW PLT1
NEXT M > > >
```

The inputs are the rank m, the genus g, and the depth of approximation n. See also the rank 2 examples in Chapter 9.

Some authors believe that it is important to center the wavelet support around zero. I do not agree and my arguments are given on page 46. Since the support of the M–band wavelets (size of the SMAT matrix) is somewhat irregular, program CRN(m, g) is available to compute it.

I define the support to always begin at zero and to equal the interval [0, CRN(m, g) + 1], (theoretically [0, (mg - 1)/(m - 1)], [4]). The number of points computed by SOVN is in general m^{n+1}[CRN(m, g) + 1] + 1. Observe that the scaling vector computed by DMG is always real. Consequently, the Haar matrix used by CWM must also be real since the DMG scaling vector cannot be a basis for complex wavelet vectors.

To test the accuracy of a wavelet matrix, use the program

```
« → M « M M TRNP × DUP D←M μ / DUP SIZE 1 GET →
 N « N ZERO 0 B→M MABS RSUM VΣ N N 1 - × / »»»
```

For m = 4 and g = 4, the accuracy is about 11 digits or 110 dB dynamic range. This is an HP 48 limitation. Now compute the scaling and wavelet functions using n = 2 so that the resulting matrix is 4 × 321. This takes about 32 minutes, the time required to evaluate the **SIN** command nearly 70,000 times on the HP 48. Running the above paraunitary test, now on the function matrix, yields an accuracy of about √8.3E-5 = 9E-3 (41 dB) which is a little over 2 digits. This is better than the result on page 66 due to both increased regularity and increased rank. See Chapter 11 for an alternative computation method.

Now let us examine the moments. The below integrals should be zero.

$$\int_{-\infty}^{\infty} x^k \, w_n(x) \, dx = 0 \qquad n = 1,...m, \quad k = 1, \ldots g - 1$$

The following program will approximate these integrals. Input M is the above computed function matrix and g is the genus.

```
« → M g « M SIZE EVAL { } → m N L « 1 g 1 - FOR
 J 0 N 1 - FOR K K J ^ NEXT N →ARRY DUP N ONE
 DOT / → V « 2 m FOR K 'L' M K EROW V DOT STO+
NEXT » NEXT L V←L g 1 - m 1 - 2 →LIST RDM TRNP
                    »»»
```

The output is a matrix whose rows correspond to the wavelet vectors and whose columns correspond to the moments x, x^2, x^3, ... For the above example with m = 4 and g = 4, the resulting matrix is

$$\begin{bmatrix} -2E{-}12 & -3E{-}12 & -3E{-}12 \\ 5E{-}12 & 6E{-}12 & 7E{-}12 \\ 1E{-}12 & 2E{-}12 & 3E{-}12 \end{bmatrix}$$

Thus, even with poor orthogonality, the wavelet moments are on the order of the HP 48 precision which is impressive. The following program will allow you to check the moments of the wavelet matrix. It is the M–band generalization of program TESTV in the MAN7 WAVE HAAR directory.

```
« → V m « V SIZE EVAL → N «
'Σ(k=0,N-1,k^m×V(k+1))/Σ(k=0,N-1,k^m)' EVAL » » »
```

With the wavelet matrix on the stack, use ENTER 2 EROW to extract the first wavelet vector. The put m = 1 on the stack and evaluate the program. After noting the result, push LAST STACK to do m = 2 . . .. The result is

$$\begin{bmatrix} -3E-12 & -5E-12 & -7E-12 \\ 8E-12 & 1E-11 & 2E-11 \\ 2E-12 & 4E-12 & 7E-12 \end{bmatrix}$$

which is again on the order of the HP 48 precision. Hence, in spite of the fact that the orthogonality of the wavelet approximations is poor, the regularity of the approximations is very good.

RANK M GENERALIZED WAVELET TRANSFORMS †

The accurate paraunitary property of the wavelet matrices defines a transform independent of their associated continuous wavelets. The wavelet matrix is a filter bank. From (12) we have that

$$\sum_{s=0}^{m-1} \sum_{r=0}^{g-1} \bar{a}^{s}_{k+mr}\, a^{s}_{k'+mr} = m\delta_{kk'} \qquad (82)$$

Let x[n] be a discrete sequence and define c_r^s to be its discrete block transform

$$c_r^s = \sum_{n=-\infty}^{\infty} \bar{a}^{s}_{n+mr}\, x[n] = \sum_{n=0}^{mg-1} \bar{a}^{s}_{n}\, x[n - mr] \qquad (83)$$

where r may be regarded as the block index and s as the subband (bank) index. Then we can derive the inverse transform as follows

† See also pages 143 through 147 for an alternate derivation of the wavelet transform.

$$\begin{aligned} x[n] &= \sum_{n'=-\infty}^{\infty} x[n']\,\delta_{nn'} \\ &= \frac{1}{m}\sum_{n'=-\infty}^{\infty} x[n']\sum_{s=0}^{m-1}\sum_{r=0}^{g-1} \bar{a}^{s}_{n'+mr}\, a^{s}_{n+mr} \\ &= \frac{1}{m}\sum_{s=0}^{m-1}\sum_{r=0}^{g-1} a^{s}_{n+mr}\sum_{n'=-\infty}^{\infty} \bar{a}^{s}_{n'+mr}\, x[n'] \\ &= \frac{1}{m}\sum_{s=0}^{m-1}\sum_{r=0}^{g-1} a^{s}_{n+mr}\, c_r^{s} \end{aligned} \tag{84}$$

Observe that for genus = 1, this is simply a Haar transform. The MAN7 WAVE HAAR directory contains several examples including the DFT as discussed in Appendix G of the MATHLIB manual. In particular, let $a_k^s = e^{i2\pi sk/m}$. Then the above formulas reduce to the standard DFT. For $g > 1$, we have a polyphase filter bank and a lapped transform.

From (83) and (84) we have *Parseval's formula*

$$\begin{aligned} &\sum_{s=0}^{m-1}\sum_{r=0}^{g-1} |c_r^{s}|^2 \\ &\quad= \sum_{s=0}^{m-1}\sum_{r=0}^{g-1}\sum_{k=-\infty}^{\infty}\sum_{k'=-\infty}^{\infty} \bar{a}^{s}_{k+mr}\, a^{s}_{k'+mr}\, x[k]\, x^{*}[k'] \\ &\quad= \sum_{k=-\infty}^{\infty}\sum_{k'=-\infty}^{\infty} m\,\delta_{kk'}\, x[k]\, x^{*}[k'] \\ &\quad= m\sum_{k=-\infty}^{\infty} |x[k]|^2 = m\,||x[k]||_2^2 < \infty \end{aligned} \tag{85}$$

Harmuth [16] and others have been promoting non-Fourier transform solutions to various problems for many years. The above transform definition thus unifies a large number of transform theories in the literature.

We can also define the generalization of the discrete Fourier series as

$$x[n] = \frac{1}{m}\sum_{s=0}^{m-1}\sum_{r=-\infty}^{\infty} a^{s}_{n+mr}\, c_r^{s} \tag{86}$$

using (83) with its corresponding Parseval formula

$$||x[n]||_2^2 = \frac{1}{m}\sum_{s=0}^{m-1}\sum_{r=-\infty}^{\infty} |c_r^{s}|^2 < \infty \tag{87}$$

Let $g = 1$ and suppose that the two sequences $x_1[n]$ and $x_2[n]$ have transform coefficients ξ_1^s and ξ_2^s, respectively. Consider the convolution

of these two sequences.

$$\sum_{k=0}^{m-1} x_1[k]\, x_2[n-k] = \sum_{k=0}^{m-1} \frac{1}{m^2} \sum_{s=0}^{m-1} \sum_{s'=0}^{m-1} a_k^s\, a_{n-k}^{s'}\, \xi_1^s\, \xi_2^{s'} \tag{88}$$

In order for a convolution theorem to exist for this transform, the sum over k must be zero for $s' \neq s$. Suppose that $a_k^s = e^{i2\pi sk/m}$ is the mth root of unity. Then the right side of (88) is

$$\begin{aligned} &\sum_{k=0}^{m-1} \frac{1}{m^2} \sum_{s=0}^{m-1} \sum_{s'=0}^{m-1} e^{i2\pi(s-s')k/m}\, e^{i2\pi ns/m}\, \xi_1^s\, \xi_2^{s'} \\ &\quad = \frac{1}{m} \sum_{s=0}^{m-1} e^{i2\pi ns/m}\, \xi_1^s\, \xi_2^s \end{aligned} \tag{89}$$

Thus, the DFT is unique, because it transforms delays into phase shifts, which gives rise to a convolution theorem. As explained in Appendix G of the MATHLIB manual, this same DFT property is also key to the paraunitary property associated with polyphase matrices.

Now the z transform also has a convolution theorem as discussed on page 561 of the MATHLIB manual.

$$\begin{aligned} &\sum_{n=-\infty}^{\infty} \sum_{k=0}^{m-1} x_1[k]\, x_2[n-k]\, z^{-n} \\ &\quad = \sum_{k=0}^{m-1} x_1[k]\, z^{-k}\, X_2(z) = X_1(z) X_2(z) \end{aligned} \tag{90}$$

for the assumed support [0, m – 1] of $x_1[k]$. From (84) or (86) we have

$$\begin{aligned} X(z) &= \frac{1}{m} \sum_{s=0}^{m-1} \sum_{r=0}^{g-1} c_r^s \sum_{n=-\infty}^{\infty} a_{n+mr}^s\, z^{-n} \\ &= \frac{1}{m} \sum_{s=0}^{m-1} \sum_{r=0}^{g-1} c_r^s\, A_r^s(z) \end{aligned} \tag{91}$$

where $A_r^s(z)$ is the indicated partial z transform of the filter bank coefficients. Thus, there is no basic convolution formula with respect to the coefficients c_r^s.

Let $p[n] = p_0 + p_1 n + \ldots + p_d n^d$ be a polynomial and suppose

$$p[n] = \frac{1}{m} \sum_{r=0}^{g-1} a_{n+mr}^0\, c_r^0 \tag{92}$$

Then

$$c_r^s = \sum_{n=-\infty}^{\infty} \bar{a}_{n+mr}^s \, p[n] = \frac{1}{m} \sum_{r'=0}^{g-1} c_{r'}^0 \sum_{n=-\infty}^{\infty} \bar{a}_{n+mr}^s \, a_{n+mr'}^0 = 0 \quad s > 0$$

$$= \sum_{k=0}^{d} p_k \sum_{n=-\infty}^{\infty} n^k \, \bar{a}_{n+mr}^s$$

$$= \sum_{k=0}^{d} p_k \sum_{n'=0}^{mg-1} [n' - mr]^k \, \bar{a}_{n'}^s \tag{93}$$

by (11). Since the p_k are arbitrary, it follows that

$$\sum_{n=0}^{mg-1} n^k \, \bar{a}_n^s = 0 \quad s = 1, \ldots m-1, \quad k = 0, \ldots d \tag{94}$$

Conversely, if (94) holds, then $c_r^s = 0$ for $s = 1, \ldots m-1$, $r = 0, \ldots, g-1$ and by the perfect reconstruction property

$$p[n] = \sum_{s=0}^{m-1} \sum_{r=0}^{g-1} a_{n+mr}^s \, c_r^s = \sum_{r=0}^{g-1} a_{n+mr}^0 \, c_r^0 \tag{95}$$

A wavelet matrix is said to be *lowpass of degree d* if

$$M(s, j) = \sum_{k=0}^{mg-1} k^j \, a_k^s = 0 \quad j = 0, \ldots d, \quad s > 0 \tag{96}$$

We showed on page 88 that if a wavelet matrix is lowpass of degree d, then so are the wavelet functions, that is, they have regularity d.

From the linear condition (9), every wavelet matrix is lowpass of degree 0. Consequently, as proved on page 85

$$\sum_{r=0}^{g-1} 1 a_{n+mr}^0 = 1 \quad n = 0, \ldots m-1 \tag{97}$$

We also have the relations

$$m \sum_{r=0}^{g-1} r \, a_{mr}^0 - m \sum_{r=0}^{g-1} r \, a_{n+mr}^0 = n \quad n = 0, \ldots m-1 \tag{98}$$

$$m^2 \sum_{r=0}^{g-1} r^2 \, a_{n+mr}^0 - m^2 \sum_{r=0}^{g-1} r^2 \, a_{mr}^0 + 2mn \sum_{r=0}^{g-1} r \, a_{mr}^0 = n^2 \tag{99}$$

Thus, if the wavelet matrix is lowpass of degree d, the coefficients c_r^0, associated with the polynomial p[n] of degree d, are linear combinations of the moments of a_k^0.

EQUIVALENT NOTIONS OF REGULARITY

The above discussion combined with that on pages 86 through 88 lead us to the equivalence of several definitions of regularity. A FIR scaling vector is said to be *regular of order N* if:

1) Perfect interpolation of discrete polynomial sequences of degree N by the scaling vector.
2) Nth order vanishing of the wavelet vector moments.
3) The frequency response of the scaling vector is maximally flat so that $A^{(s)}(e^{i2\pi n/m}) = 0$ for n = 0, . . . m − 1 and s = 1, . . . N.
4) Perfect interpolation of polynomials of degree N by the scaling function.
5) Nth order vanishing of the wavelet moments.

The zeros of the filter $\mathcal{H}^g(z)$ require that

$$H_0^{(s)}(e^{i2\pi n/m}) = 0 \quad n = 1, 2, \ldots m - 1, \quad s = 1, 2, \ldots g - 1 \tag{100}$$

which implies regularity condition 3 for values of $n \neq 0$. But observe that

$$\frac{d^s}{d\omega^s} H_0(e^{i\omega})\Big|_{\omega=0} = H_0^{(s)}(1) = \sum_{n=0}^{mg-1} a_n^0 \, (-in)^s = (-i)^s \, M(0, s) \tag{101}$$

which cannot be zero if we are to satisfy condition 1. Any change in the f(x) design constraints given on page 89 results in a wavelet matrix which is not paraunitary. All of these conditions are exactly right. Nevertheless, given the scaling vector polynomial list L computed by DMG, you can use the following program to verify that $H_0^{(s)}(1) \neq 0$ for s = 1, . . . g − 1.

```
« → L  s « L  SIZE  EVAL  →  N  'Σ(k=0,N−1,L(k+1)×k^s)' » »
```

Observe that from the zeros of $\mathcal{H}^g(z)$ that for n = 1, . . . m − 1

$$\frac{d^s}{d\omega^s} H_0(e^{i\omega} e^{i2\pi n/m})\Big|_{\omega=0} = (-i)^s \sum_{j=0}^{mg-1} j^s \, a_j^0 \, e^{-i2\pi nj/m} = 0 \tag{102}$$

by (100). Hence, for 0 < n < m

$$\sum_{k=0}^{m-1} \sum_{r=0}^{g-1} [k + mr]^s \, a_{k+mr}^0 \, e^{-i2\pi nk/m} = 0 \tag{103}$$

which implies that each of the partial moments

$$\sum_{r=0}^{g-1} [k + mr]^s \, a^0_{k+mr} \tag{104}$$

must be a constant independent of k. This is the generalization of (97). The following program computes these partial moments. V is the scaling vector, m is the rank, and s = 0, 1, . . . g – 1.

```
≪ → V m s ≪ V SIZE EVAL → N ≪ 0 m 1 - FOR k 'Σ(r=0
,N/m-1,(k+m×r)^s×V(k+m×r+1))' →NUM NEXT m →ARRY ≫ ≫ ≫
```

The next program computes the autocorrelation matrix corresponding to the wavelet matrix M.

```
≪ → M ≪ M SIZE EVAL → m N ≪ 1 m FOR J M J
EROW DUP VCORR NEXT m →LIST M←RL ≫ ≫ ≫
```

Observe that every mth value except the center value is zero. Using the command **CSUM**, you can verify (20) by summing down the columns of the autocorrelation matrix. The result is a vector with all zero elements except the center value which equals m^2.

There are yet some other notions of regularity which are more rigorous (see [2] and [4]). In [5], Daubechies defines *sum rules* which are equivalent to (104) in the rank m case. The dilation equation

$$f(x) = \sum_{k=0}^{N} a_k^0 \, f(mx - k) \tag{105}$$

is called a *two-scale difference equation*. If f(x) is an L^1 solution of (105), that is f(x) is absolutely integrable, and

$$\int_0^N f(x)\, dx = 1$$

then f(x) is Holder continuous

$$|f(x) - f(y)| \le C\,|x - y|^{\alpha} \tag{106}$$

with the Holder exponent $\alpha = |\ln \lambda| / \ln 2$.

As the genus increases, there is increased regularity and we can speak of the Lth derivative $f^{(L)}(x)$ being Holder continuous. Specifically she shows that there exists a number λ such that f(x) has $\lfloor \lambda N \rfloor$, the floor of λN, continuous derivatives and $f^{(L)}(x)$ has Holder exponent $\lambda N - L$. She uses the notation C^L to denote the subset of $\mathbf{R}$ of functions f(x) which are continuously differentiable L times everywhere.

In [3], Daubechies makes a distinction between the Lth derivative of a wavelet being continuous and the wavelet moments being zero. While the latter can lead to the former, it does not necessarily follow, in general. In the case of compactly supported wavelets, $w_n(x) \in C^{N-1}$ only if $w_n(x)$ has N vanishing moments [3]. However, the Daubechies rank 2 wavelets are in $C^{\mu N}$ where $\mu \sim .2$. This means that about 80% of the zero moments are wasted relative to regularity in terms of continuous higher order derivatives and a given regularity could be achieved with a shorter coefficient set with only N/5 vanishing moments. The importance of vanishing moments versus continuous derivatives depends on the application. A detailed discussion of all this is beyond the scope of this tutorial, and I refer you to her papers.

Additional notions of regularity given by Strang [40] are:

6) Smooth functions f(x) can be approximated with error $O(h^p)$ by combinations at every scale $h = 2^{-j}$:

$$|| f(x) - \sum_k a_k s(2^j x - k) ||_2 \le C 2^{-jp} || f^{(p)}(x) ||_2$$

7) Wavelet coefficients of a smooth function f(x) decay like

$$\int f(x) w(2^j x) dx \le C 2^{-jp}$$

Both Daubechies and Strang also examine the eigenvalues and eigenvectors of the two-scale transition matrix computed by program SMAT to formulate additional meanings of regularity.

Finally, we point out that a longer filter (higher genus) can be useful, independent of regularity, to obtain other objectives. Longer filters can produce better subband isolation, for example.

RANK M BIORTHOGONAL CASE

On page 84 we derived equation (18)

$$\sum_{n=0}^{m-1} H_s(z e^{i2\pi n/m}) H_{s'}^*(z^{-1} e^{-i2\pi n/m}) = m^2 \delta_{ss'} \qquad (107)$$

from which we have the special case

$$\sum_{n=0}^{m-1} H_0(z e^{i2\pi n/m}) H_0^*(z^{-1} e^{-i2\pi n/m}) = m^2 \qquad (108)$$

where $H_0(z) = \mathcal{H}^g(z) Q_g(z)$ as defined by (31). Equation (37) defined

$$P_g^m(y) = Q_g(z)Q_g(z^{-1}) \tag{109}$$

where y is defined by (9–8). Equation (32) may be generalized as

$$\mathcal{H}(ze^{i2\pi n/m})\,\mathcal{H}(z^{-1}e^{-i2\pi n/m}) = \frac{1 - \cos[m(\omega + 2\pi n/m)]}{1 - \cos[\omega + 2\pi n/m]} \tag{110}$$

where again $z = e^{i\omega}$. Define the generalization of y to be

$$v = v(n) = \tfrac{1}{2}\,[1 - \cos(\omega + 2\pi n/m)] \tag{111}$$

noting that v(0) = y. Observe that (110) is a real function for real ω. Define the generalization of x = cos ω to be

$$u = u(n) = \cos[\omega + 2\pi n/m] \tag{112}$$

where we note that u(0) = x and that (110) is a polynomial in u.

Consider the rank 2 case again. The method of solution is given on pages 61 and 62. This was generalized to the rank m case on page 89. Comparing solutions with (108) through (110), we see that the same steps apply yielding the scaling vector polynomial.

$$H_0(Z) = a_0 + a_1Z^{-1} + a_2Z^{-2} + \ldots + a_{mg-1}Z^{-(mg-1)} \tag{113}$$

where $Z = ze^{i2\pi n/m} = z(-1)\text{^}n$. For n = 0, this is just the scaling vector. To obtain the wavelet vector we let n = 1 and perform the following factorization to isolate the coefficients.

$$H_0(-z) = z^{-(mg-1)}\,[a_0z^{mg-1} + a_1(-1)z^{mg-2} + \ldots + a_{mg-1}(-1)] \tag{114}$$

Observe that the new coefficients are not only modulated, but in the process of introducing it into the coefficients, the coefficients are reversed. If you are having trouble following this, see page 46. These are precisely the same steps which program QMFL uses to compute the wavelet vector. For s′ = s, (107) is a statement that each $H_s(z)$ is a spectral factor of a Mth band filter [27].

One can generalize the rank 2 biorthogonal case on page 72 as follows. Define

$$H^0(z) = \mathcal{H}^{N_1}(z)\,Q^1(z)\,z^{-n_1} \tag{115}$$

$$H^0(z) = \mathcal{H}^{N_2}(z)\,Q^2(z)\,z^{-n_2} \tag{116}$$

where $N_1 + N_2 = 2g$, and $n_2 - n_1 = (N_1 - N_2)/2$. Also define

$$Q^1(z)\ Q^2(z^{-1}) = Q^1(z^{-1})\ Q^2(z) = P_g^M(y) \tag{117}$$

so with these definitions,

$$H^0(z)H^0(z^{-1}) = \mathcal{H}^g(z)\ \mathcal{H}^g(z^{-1})\ P_g^M(y) \tag{118}$$

By adding m as an input to program CLPCF and changing POFY to POMY as well as changing MOZ to MOMZ, program CLPCF can be generalized to the rank m case. However, there is little point to doing it, since there currently is no way to compute the M–band phase linear wavelet vectors. In particular, the generalization of (111) to m > 2 introduces complex coefficients and you will find that the wavelet matrix is not paraunitary. In addition, the results of Soman [38] and Gopinath [13], [14] are more general results.

FILTER BANKS AS GENERALIZED TRANSFORMS

In many practical engineering applications, filter banks can be useful as generalized transforms, even when there is no true convolution theorem. Perfect reconstruction is important in coding applications, but often irrelevant in adaptive filtering applications.

Lapped transforms are often used simply to obtain efficient convolution engines. This generally entails blocking the data. Block transforms such as the DFT provide circular convolution, and special techniques are required to realize linear convolution. The result is normally a filter bank which is a factor of two above critical decimation, as discussed on page 568 of the MATHLIB manual. Consider two examples. Move to the MAN6 PROC directory and run the following program.

```
« 100 ZERO 40 1 PUT 8 FFTLC »
```

Program FFTLC is discussed on page 388 of the MATHLIB manual. Now examine the output time series, which to 10 digit accuracy is all zeros except for the 40th value which is a 1. This is what we expect for the impulse response from a perfect reconstruction filter bank such as the FFT. Now modify the program substituting TMIC3 for FFTLC and run it again. The result is a time series with 60 values since values were lost at each end initializing the polyphase filters associated with the transmux. Observe that the nonzero values are

$$V(5) = V(37) = 5.7915938E-3$$
$$V(13) = V(29) = -5.7915938E-3$$
$$V(21) = 1$$

Thus, the output has the 1 value, but it also has four additional outputs since the polyphase filter design does not correspond to a perfect reconstruction filter bank. Changing 40 to 41 in the above transmux example yields the output

$$V(6) = V(38) = .0009307532$$
$$V(14) = V(30) = -.0009307532$$
$$V(22) = 1$$

where the nonzero values are shifted by 1 and the magnitudes are different.

Now let us look at the equations to see where we lose perfect reconstruction. The analysis transform may be written as

$$y_q(k) = \sum_{l=0}^{mg-1} h(l)\, x(k - l)\, e^{i2\pi(k-l)q/m} \qquad (119)$$

which is the convolution of the analysis polyphase filter h(n) with the frequency shifted $x(n)e^{i2\pi nq/m}$ input. The k is the time index, the q is the generalized frequency index, g is the genus, and m is the size of the underlying DFT, which is the N input to FFTLC and TMIC3. The corresponding synthesis transform is

$$z(n) = \sum_{q=0}^{m-1} e^{-i2\pi(n+mg-1)q/m} \sum_{j=0}^{mg-1} g(j)\, y_q(n - j) \qquad (120)$$

where g(n) is the synthesis polyphase filter. Setting j = s + mr/2, we can rewrite (120) as

$$z(n) = \sum_{s=0}^{m/2-1} \sum_{r=0}^{2g-1} g(s + mr/2) \sum_{q=0}^{m-1} e^{-i2\pi(n+mg-1)q/m}\, y_q(n - s - mr/2) \qquad (121)$$

Now decimation by a factor m/2 means that $y_q(n - s - mr/2) = 0$ for $n - s \neq jm/2$ for some j which implies that s = n MOD m/2 and the sum over s in (121) vanishes. Substituting (119) into the sum over q right hand part of (121) gives

$$\sum_{l=0}^{mg-1} h(l)x(n - s - mr/2 - l) \sum_{q=0}^{m-1} e^{-i2\pi(n-mg+1-n+s+mr/2+l)q/m} \qquad (122)$$

Now the sum over q is zero unless

$$s + mr/2 + l = mg - 1 \tag{123}$$

so (122) reduces to

$$m\ x(n - mg + 1)\ h(mg - 1 - s - mr/2) \tag{124}$$

and (121) becomes

$$z(n) = m\,x(n - mg + 1) \times \sum_{r=0}^{2g-1} g(n\,\mathrm{MOD}\,m/2 + mr/2)h(mg - 1 - n\,\mathrm{MOD}\,m/2 - mr/2) \tag{125}$$

Thus, perfect reconstruction requires that

$$m \sum_{r=0}^{2g-1} g(n\,\mathrm{MOD}\,m/2 + mr/2)h(mg - 1 - n\,\mathrm{MOD}\,m/2 - mr/2) = 1 \tag{126}$$

and that

$$\sum_{r=0}^{2g-1} g(n\,\mathrm{MOD}\,m/2 + mr/2)h(mk - 1 - n\,\mathrm{MOD}\,m/2 - mr/2) = 0 \tag{127}$$

for $k \neq g$. Now in our transmux example, $g(n) = h(n) = h(3m - 1 - n)$ for $n = 0, 1, \ldots 3m - 1$, so the following program will test these orthogonality conditions.

```
« → N R « N 3 × ZERO N H30N OVER VCMB VCMB N
2 / → H n « 1 n FOR J 'Σ(r=6,11,H(J+r×n)×H(J+r×n+R×N))'
    →NUM NEXT n →ARRY N × n × 10 RND » » »
```

The result for N = 8 is

```
R = 0  [1  1  1  1]
R = 1  [-.0057915938  -.0009307532   .0032909467  -.0009307532]
R = 2  [ .0057915938   .0009307532  -.0032909467   .0009307532]
R = 3  [0  0  0  0]
```

and similarly for R < 0. Comparing these values with the amplitudes of the transmux output values explains their origin. The following program will plot the frequency response of the polyphase filter.

```
« 8 H3ON 128 ZFILN 128 TWIDL FFT HALF1 VABS VSQ
    1E-9 VADD VLOG 10 × PLT1 »
```

Observe that the first lobe is down 33 dB from the main lobe which compares with 13 dB for an FFT with no window. Thus, the polyphase filter improves the subband isolation by 20 dB.

We now show that the computational cost of this improvement is small. Analysis bank (119) may be rewritten as

$$y_q(km/2) = \sum_{s=0}^{m-1} \sum_{r=0}^{g-1} h(s + mr/2)\, x(mk/2 - s - rm/2)\, e^{i2\pi(mk/2-s)q/m} \quad (128)$$

$$= (-1)^{kq} \sum_{s=0}^{m-1} u_s(k)\, e^{-i2\pi sq/m} = (-1)^{kq}\, DFT_s[u_s(k)](q)$$

where k is now the half block index and

$$u_s(k) = \sum_{r=0}^{g-1} h(s + mr/2)\, x(mk/2 - s - rm/2) \quad (129)$$

Since g = 3, the above convolution is over only 3 values. Given $u_s(k)$ and $u_s(k + 1)$ for k even, we mux them up as in TMIC3 and DFT them both. After demuxing the two FFTs, the second one must then be modulated by $(-1)^q$. This is performed by multiplying by the S vector. Now define

$$v_r(n) = \sum_{q=0}^{m-1} e^{-i2\pi nq/m}\, y_q(IP(2n/m)m/2 - mr/2) \quad (130)$$

$$= DFT_q[y_q(IP(2n/m)m/2 - mr/2)](n)$$

Then ignoring delays since they are implemented by proper array indexing, (121) can be written as

$$z(n) = \sum_{r=0}^{2g-1} g(n\,MOD\,m/2 + mr/2)\, v_r(n) \quad (131)$$

The output convolution in our example contains 6 terms. The tricky part of the synthesis bank is extracting the linearly convolved values, which is the reason the analysis and synthesis banks are not symmetric. See the software for details.

Now a different set of filter coefficients could lead to a perfect reconstruction filter bank, but to satisfy both orthogonality and subband isolation (in interference canceler applications, the aliasing does not cancel) requires more coefficients and thus longer convolutions. This is the trade space. While there is no true convolution theorem for the transmux, the interpretation of applying a symmetric and real multiplicative weighting function to the pseudospectrum values at the output of the analysis bank does have meaning provided the filter is computed in terms of those pseudospectrum values. For more discussion on transmultiplexing, see [1] and [32].

PERFECT RECONSTRUCTION FILTER BANKS

Independent of the wavelet movement, Vaidyanathan and his students have been characterizing FIR perfect reconstruction filter banks for a number of years. They have shown that the polyphase matrix P[A](z) defined by (8) can be written in state space form as [12]

$$P[A](z) = C(zI - Q)^{-1}B + D \tag{132}$$

corresponding to the state space system in z transform notation

$$zX(z) = QX(z) + BU(z), \qquad Y(z) = CX(z) + DU(z) \tag{133}$$

The rank of Q is the minimal number of delays to realize the system and equals the degree of DET P[A](z). Every polyphase matrix and every wavelet matrix of McMillan degree g – 1 can be written as [9]

$$P[A](z) = [I - (1 - z^{-1})v_{g-1}v_{g-1}^{H}] \ldots [I - (1 - z^{-1})v_1 v_1^{H}]h \tag{134}$$

where h is the characteristic Haar (unitary) matrix of the wavelet matrix A and $v_1, \ldots v_{g-1}$ are each m dimensional vectors of unit length. A polyphase matrix of degree g – 1 has genus g, but for m > 2, a genus g wavelet matrix need not always have degree g – 1 [22].

Equation (134) is the Vaidyanathan parameterization for filter banks and thus wavelet matrices. In his papers he derives the minimum number of parameters required to characterize these matrices. Observe that given the Vaidyanathan parameterization, the Pollen product discussed on page 92 easily follows. Writing (134) as

$$P[A](z) = \prod_{k=1}^{g-1} L_k[A](z)\ h \tag{135}$$

where each $L_k[A](z)$ is a degree 1 lattice matrix, then

$$P[A](z) = P[A'](z)h^{-1}P[A''](z) = L[A'](z)\ L[A''](z)\ h \tag{136}$$

If H(z) is the corresponding analysis filter, then

$$H(z) = P[A](z^m)\ V_m(z) \tag{137}$$

where $V_N(z) = [1\ \ z^{-1}\ \ \ldots\ z^{-(N-1)}]^T$. Hence, if P[A](z) has degree g – 1, then $P[A](z^m)$ has degree m(g – 1) and the rows of H(z) each have degree mg – 1. A real Haar (unitary) matrix h must satisfy $\binom{m}{2}$ constraints. Each v_k has m – 1 independent parameters. Thus, the total number of angles in the parameter space is [47]

$$N_u = (m - 1)(g - 1) + \binom{m}{2} \tag{138}$$

Since the Haar wavelet matrix must also satisfy linear condition (9), the number of parameters is further reduced by m – 1. Thus, the number of parameters is [54]

$$N_w = (m - 1)(g - 2) + \binom{m}{2} \tag{139}$$

For m = 2, there are (2 – 1)(g – 2) + 1 parameters to determine which equals g – 1. This is the number of parameters programs WD4, WD6, SWD4, and SWD6 use.

We close this section by summarizing some properties of perfect reconstruction filter banks [50]. If $H_p(z)$ is the analysis polyphase matrix and $G_p(z)$ is the synthesis polyphase matrix, a sufficient condition for perfect reconstruction is

$$[G_p(z)]^H H_p(z) = z^{-k} I \tag{140}$$

for non-negative integer k. Other solutions are obtained by pseudocirculant shifting of the identity matrix [46], and are therefore similar within a delay to the solution (139). A necessary and sufficient condition for the existence of $G_p(z)$ is that the determinant of $H_p(z)$ be a monomial in z.

A subset of the set of perfect reconstruction systems are those for which the analysis and synthesis filters have the same length. Vetterli shows that for equal length filters, the orthogonality of overlapping blocks is a sufficient condition for perfect reconstruction. This was the condition (127) which the transmux failed to satisfy.

A subset of the analysis/synthesis systems of equal length are those with identical (within a time reversal) analysis and synthesis filters. A necessary and sufficient condition for this subset is that

$$[H_p(z^{-1})]^H H_p(z) = I \tag{141}$$

$H_p(z)$ be paraunitary and we choose $G_p(z) = z^{-k} H_p(z^{-1})$.

The following six sections summarize some alternative perfect reconstruction filter bank design techniques. They involve brute force nonlinear optimization schemes and are thus not well suited to the HP 48. Koilpillai reports 100 iterations per coefficient is typical [25]. For these design techniques, I recommend that you do the design on a PC and then download the result to the HP 48 for analysis.

COSINE MODULATED WAVELET MATRICES

Koilpillai has developed a design technique for M-band perfect reconstruction filter banks of length $N = 2\alpha m$, $\alpha \in \mathbf{N}$ [25]. The bank is explicitly cosine modulated. The analysis and synthesis filters, $H_k(z)$ and $G_k(z)$ respectively for $k = 0, \ldots m - 1$, of the pseudo-QMF bank are obtained by modulating a real symmetric impulse response $f(n)$

$$h_k(n) = 2f(n) \cos\left[(2k + 1)\frac{\pi}{2m}\left(n - \frac{N - 1}{2}\right) + \theta_k\right] \tag{142}$$

$$g_k(n) = 2f(n) \cos\left[(2k + 1)\frac{\pi}{2m}\left(n - \frac{N - 1}{2}\right) - \theta_k\right] \tag{143}$$

for $n = 0, \ldots N - 1$ and $k = 0, \ldots m - 1$. It is easily verified that

$$g_k(n) = h_k(N - 1 - n) \qquad G_k(z) = z^{-(N-1)} H_k^H(z^{-1}) \tag{144}$$

Define

$$c_{k,n} = 2 \cos\left[(2k + 1)\frac{\pi}{2m}\left(n - \frac{N - 1}{2}\right) + (-1)^k\frac{\pi}{4}\right] \tag{145}$$

where we have picked one possible definition for θ_k. See [25] for others. By the periodicity of the cosine modulation

$$c_{k,(n+2mr)} = (-1)^r c_{k,n} \qquad r = 0, \ldots \alpha - 1 \tag{146}$$

for non-negative integer r. Now the z transform of the lowpass prototype filter $f(n)$ can be written as

$$\begin{aligned} F(z) &= \sum_{n=0}^{N-1} f(n)\, z^{-n} \\ &= \sum_{s=0}^{2m-1} \left[\sum_{r=0}^{\alpha-1} f(s + 2mr)\, z^{-2mr}\right] z^{-s} \\ &= \sum_{s=0}^{2m-1} F_s(z^{2mr})\, z^{-s} \end{aligned} \tag{147}$$

so the analysis filters can be expressed as

$$\begin{aligned} H_k(z) &= \sum_{n=0}^{N-1} h_k(n)\, z^{-n} = \sum_{n=0}^{N-1} c_{k,n}\, f(n)\, z^{-n} \\ &= \sum_{s=0}^{2m-1} c_{k,s}\, z^{-s} \left[\sum_{r=0}^{m-1} (-1)^r f(s + 2mr) z^{-2mr} \right] \\ &= \sum_{s=0}^{2m-1} c_{k,s}\, z^{-s}\, F_s(-z^{2m}) \end{aligned} \tag{148}$$

Koilpillai proves that the necessary and sufficient conditions for the prototype filter f(n) to give rise to a unitary filter bank are that

$$F_s^*(z^{-1})F_s(z) + F_{m+s}^*(z^{-1})F_{m+s}(z) = \frac{1}{2} \qquad s = 0, \ldots m-1 \tag{149}$$

Observe that from the even symmetry of F(z)

$$F_s(z) = z^{-(\alpha-1)}\, F_{2m-1-s}(z^{-1}) \tag{150}$$

and thus if we define $J = \lfloor m/2 \rfloor$ to be the floor of m/2, then (150) can be written as:

1. For m even

$$F_s^*(z^{-1})F_s(z) + F_{m+s}^*(z^{-1})F_{m+s}(z) = \frac{1}{2} \qquad s = 0, \ldots J-1 \tag{151}$$

2. For m odd

$$\begin{aligned} F_s^*(z^{-1})F_s(z) + F_{m+s}^*(z^{-1})F_{m+s}(z) &= \frac{1}{2} \qquad s = 0, \ldots J-1 \\ F_J^*(z^{-1})F_J(z) &= \frac{1}{4} \end{aligned} \tag{152}$$

The derivation of (149) gives good insight into why it all works, so we repeat Koilpillai's derivation. Let **C** and **S** be the Type IV discrete cosine transform (DCT) and discrete sine transform (DST) matrices, respectively [51]. The elements are defined by

$$[\mathbf{C}]_{k,n} = \sqrt{\frac{2}{m}} \cos\left[\frac{\pi}{m}\left(k + \frac{1}{2}\right)\left(n + \frac{1}{2}\right) \right] \qquad 0 \le k, n \le m-1 \tag{153}$$

$$[\mathbf{S}]_{k,n} = \sqrt{\frac{2}{m}} \sin\left[\frac{\pi}{m}\left(k + \frac{1}{2}\right)\left(n + \frac{1}{2}\right) \right] \qquad 0 \le k, n \le m-1 \tag{154}$$

From [51] we have the identities

$$[\mathbf{C}]^{-1} = \mathbf{C} = [\mathbf{C}]^{H} \qquad [\mathbf{S}]^{-1} = \mathbf{S} = [\mathbf{S}]^{H} \tag{155}$$

Let I_m denote the m × m identity matrix and define the m × m *reflection matrix* J_m which is also called an *exchange matrix* or reverse operator as (apply **REV**→ or **REV**↑ to I_m to create it)

$$J_m = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 1 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \end{bmatrix} \tag{156}$$

Also define the cosine modulation matrices

$$[A_0]_{k,n} = 2\cos\left[(2k+1)\frac{\pi}{2m}\left(n - m + \frac{1}{2}\right) + (-1)^k\frac{\pi}{4}\right] \tag{157}$$

$$[A_1]_{k,n} = 2\cos\left[(2k+1)\frac{\pi}{2m}\left(n + \frac{1}{2}\right) + (-1)^k\frac{\pi}{4}\right] \tag{158}$$

for $0 \le k, n \le m-1$. The matrices A_0 and A_1 can be written in terms of **C** and **S** as follows

$$A_0 = \sqrt{m}\ [\mathbf{C} + \Lambda\mathbf{S}] \qquad A_1 = \sqrt{m}\ [\mathbf{C} - \Lambda\mathbf{S}] \tag{159}$$

where Λ is an m × m diagonal matrix whose non-zero elements are given by $[\Lambda]_{k,k} = (-1)^k$ for k = 0, . . . m − 1. With these definitions, we can write

$$A_0^{H}A_0 = m[2I_m + \mathbf{C}\Lambda\mathbf{S} + \mathbf{S}\Lambda\mathbf{C}] \tag{160}$$

$$A_1^{H}A_1 = m[2I_m - \mathbf{C}\Lambda\mathbf{S} - \mathbf{S}\Lambda\mathbf{C}] \tag{161}$$

$$A_0^{H}A_1 = m[-\mathbf{C}\Lambda\mathbf{S} + \mathbf{S}\Lambda\mathbf{C}] = -A_1^{H}A_0 \tag{162}$$

From the definitions, we also have

$$\begin{aligned}[\Lambda\mathbf{S}J_m]_{k,n} &= (-1)^k[\mathbf{S}J_m]_{k,n} = (-1)^k[\mathbf{S}]_{k,(m-1-n)} \\ &= (-1)^k\sqrt{\tfrac{2}{m}}\ \sin\left[\frac{\pi}{2}(2k+1) - \frac{\pi}{m}\left(k+\frac{1}{2}\right)\left(n+\frac{1}{2}\right)\right] \\ &= \sqrt{\tfrac{2}{m}}\ \cos\left[\frac{\pi}{m}\left(k+\frac{1}{2}\right)\left(n+\frac{1}{2}\right)\right] = [\mathbf{C}]_{k,n}\end{aligned} \tag{163}$$

for $0 \le k, n \le m-1$ which yields the identities

$$\Lambda S J_m = C \qquad \Lambda S = C J_m \qquad S\Lambda = J_m C \tag{164}$$

Substitution of (164) into (160) through (162) yields

$$A_0^H A_0 = 2m[I_m + J_m] \tag{165}$$

$$A_1^H A_1 = 2m[I_m - J_m] \tag{166}$$

$$A_1^H A_0 = A_0^H A_1 = 0 \tag{167}$$

Now define the m × m matrices

$$[B_0]_{k,n} = c_{k,n} \qquad [B_1]_{k,n} = c_{k,n+m} \qquad 0 \le k, n \le m - 1 \tag{168}$$

Then it is easily verified that:

1. For α even

$$B_0 = (-1)^{\alpha/2} A_1 \qquad B_1 = (-1)^{\alpha/2-1} A_0 \tag{169}$$

2. For α odd

$$B_0 = (-1)^{(\alpha-1)/2} A_0 \qquad B_1 = (-1)^{(\alpha-1)/2} A_1 \tag{170}$$

Hence, it follows that

$$B_0^H B_0 = 2m[I_m + (-1)^{(\alpha-1)} J_m] \tag{171}$$

$$B_1^H B_1 = 2m[I_m - (-1)^{(\alpha-1)} J_m] \tag{172}$$

$$B_1^H B_0 = B_0^H B_1 = 0 \tag{173}$$

These are the key identities which we require for the derivation.

Equation (148) can be written as

$$H_k(z) = \sum_{s=0}^{m-1} \left\{ c_{k,s} F_s(-z^{2m}) z^{-s} + c_{k,(s+m)} F_{s+m}(-z^{2m}) z^{-(s+m)} \right\} \tag{174}$$

so we define the m × m diagonal matrices

$$[D_0(z)]_{s,s} = F_s(-z) \qquad [D_1(z)]_{s,s} = F_{s+m}(-z) \tag{175}$$

for s = 0, . . . m − 1. Then we can write

$$h(z) = [B_0 D_0(z^{2m}) + z^{-m} B_1 D_1(z^{2m})] V_m = E(z^m) V_m(z) \tag{176}$$

where the analysis filter vector is

$$h(z) = [H_0(z) \quad H_1(z) \ldots H_{m-1}(z)]^T \tag{177}$$

$V_m(z) = [1 \; z^{-1} \ldots z^{-(m-1)}]^T$, and E(z) is the polyphase (Laurent) matrix.

Perfect reconstruction means $E^H(z^{-1})E(z) = m\ I_m$ so we have using identities (171) through (173)

$$\begin{aligned} E^H(z^{-1})E(z) &= 2m\{D_0^H(z^{-2m})[I_m + (-1)^{(m-1)}J_m]D_0(z^{2m}) \\ &\quad + D_1^H(z^{-2m})[I_m - (-1)^{(m-1)}J_m]D_1(z^{2m})\} \\ &= 2m[D_0^H(z^{-2m})D_0(z^{2m}) + D_1(z^{-2m})D_1(z^{2m})] = m\ I_m \end{aligned} \tag{178}$$

Rewriting (178) as m scalar equations gives (149).

For m = 2, a pair of filters {P(z), Q(z)} satisfying (149)

$$P^H(z^{-1})P(z) + Q^H(z^{-1})Q(z) = I \tag{179}$$

are called *power complementary pairs* which have been explicitly parameterized by Vaidyanathan [43] as

$$\begin{bmatrix} P(z) \\ Q(z) \end{bmatrix} = \prod_{k=1}^{K-1} \left\{ \begin{bmatrix} \cos\theta_k & \sin\theta_k \\ \sin\theta_k & -\cos\theta_k \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & z^{-1} \end{bmatrix} \right\} \begin{bmatrix} \cos\theta_0 \\ \sin\theta_0 \end{bmatrix} \tag{180}$$

Koilpillai's design approach is to implement the M-band case with length $N = 2\alpha m$ coefficient vectors (genus 2α) as m power complementary pairs $\{F_s, F_{s+m}\}$ for $s = 0, \ldots m-1$. By (150) this can be reduced to J power complementary pairs and there are thus $\alpha \times J$ parameters $\theta_{s,k}$, for $s = 0, \ldots J-1$ and $k = 0, \ldots \alpha - 1$. The result is

$$\begin{bmatrix} F_s(z) \\ F_{s+m}(z) \end{bmatrix} = \frac{1}{\sqrt{2}} \prod_{k=1}^{\alpha-1} \left\{ \begin{bmatrix} \cos\theta_{s,k} & \sin\theta_{s,k} \\ \sin\theta_{s,k} & -\cos\theta_{s,k} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & z^{-1} \end{bmatrix} \right\} \begin{bmatrix} \cos\theta_{s,0} \\ \sin\theta_{s,0} \end{bmatrix} \tag{181}$$

The Koilpillai design approach does not generally lead to a wavelet matrix because the linear condition

$$\sum_{n=0}^{N-1} h_0(n) = \sum_{s=0}^{2m-1} c_{0,s}\ F_s(-1) = m \tag{182}$$

is not satisfied for most prototype filters f(n). Gopinath [8] has worked out a parameterization for the wavelet case which we now will describe. Define the angle sum

$$\Theta_s = \sum_{k=0}^{\alpha-1} \theta_{s,k} \tag{183}$$

and consider first the m even case. From (181) we have

$$\begin{bmatrix} F_s(-1) \\ F_{s+m}(-1) \end{bmatrix} = \frac{1}{\sqrt{2}} \prod_{k=1}^{\alpha-1} \left\{ \begin{bmatrix} \cos\theta_{s,k} & -\sin\theta_{s,k} \\ \sin\theta_{s,k} & \cos\theta_{s,k} \end{bmatrix} \right\} \begin{bmatrix} \cos\theta_{s,0} \\ \sin\theta_{s,0} \end{bmatrix} \tag{184}$$

$$= \frac{1}{\sqrt{2}} \begin{bmatrix} \cos\left(\sum_{k=0}^{\alpha-1} \theta_{s,k}\right) \\ \sin\left(\sum_{k=0}^{\alpha-1} \theta_{s,k}\right) \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} \cos\Theta_s \\ \sin\Theta_s \end{bmatrix}$$

From (150) we also have

$$\begin{bmatrix} F_{2m-1-s}(-1) \\ F_{m-1-s}(-1) \end{bmatrix} = \begin{bmatrix} (-1)^{(\alpha-1)} F_s(-1) \\ (-1)^{(\alpha-1)} F_{s+m}(-1) \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} (-1)^{(\alpha-1)} \cos\Theta_s \\ (-1)^{(\alpha-1)} \sin\Theta_s \end{bmatrix} \tag{185}$$

For m odd, the above equations are true for s = 0, . . . J − 1 and the Jth power complementary pair is a pure delay. See (149) and (199).

$$F_J(-1) = \frac{1}{2}\cos\left(\frac{\pi}{2}(\alpha - 1)\right) \tag{186}$$

$$F_{J+m}(-1) = \frac{1}{2}\sin\left(\frac{\pi}{2}(\alpha - 1)\right) \tag{187}$$

The cosine modulation can be written as

$$c_{k,n} = 2\cos\left[(2k+1)\frac{\pi}{2m}\left(n - \frac{N-1}{2}\right) + (-1)^k\frac{\pi}{4}\right] = 2\cos\beta_{k,n} \tag{188}$$

for n = 0, . . . J − 1 which defines $\beta_{k,n}$. Similarly,

$$c_{k,n+m} = 2\cos\left[\beta_{k,n} + \frac{\pi}{2}(2k+1)\right] = -2(-1)^k \sin\beta_{k,n} \tag{189}$$

$$c_{k,2m-1-n} = 2(-1)^k(-1)^{(1-\alpha)}\sin\beta_{k,n} \tag{190}$$

$$c_{k,m-1-n} = 2(-1)^{(1-\alpha)}\cos\beta_{k,n} \tag{191}$$

for n = 0, . . . J − 1. From (182) we then have for m even

$$\sum_{n=0}^{N-1} h_k(n) = \sum_{s=0}^{J-1} [c_{k,s}F_s(-1) + c_{k,s+m}F_{s+m}(-1) + c_{k,m-1-s}F_{m-1-s}(-1) + c_{k,2m-1-s}F_{2m-1-s}(-1)] \tag{192}$$

Hence, we can write

$$\begin{aligned}\sum_{n=0}^{N-1} h_0(n) &= \frac{2}{\sqrt{2}}\sum_{s=0}^{J-1} [\cos\beta_{0,s}\cos\Theta_s - \sin\beta_{0,s}\sin\Theta_s \\ &+ \cos\beta_{0,s}\sin\Theta_s + \sin\beta_{0,s}\cos\Theta_s] \\ &= \frac{2}{\sqrt{2}}\sum_{s=0}^{J-1} [\cos(\beta_{0,s} + \Theta_s) + \sin(\beta_{0,s} + \Theta_s)] \\ &= 2\sum_{s=0}^{J-1} \sin\left(\frac{\pi}{4} + \beta_{0,s} + \Theta_s\right) = m\end{aligned} \tag{193}$$

Now (193) requires that

$$\sum_{s=0}^{J-1} \sin\left(\frac{\pi}{4} + \beta_{0,s} + \Theta_s\right) = J \tag{194}$$

which can only happen if each of the J terms of the sum equals 1, the maximum value of sin x, so we have that

$$\sin\left(\frac{\pi}{4} + \beta_{0,s} + \Theta_s\right) = 1 \tag{195}$$

for s = 0, . . . J − 1 which implies

$$\Theta_s = \frac{\pi}{4} - \beta_{0,s} = \frac{\pi}{2}\alpha - \frac{\pi}{4m}(2s + 1) \tag{196}$$

For odd m with J = (m − 1)/2 we have from (145)

$$c_{0,J} = 2\cos\left(\frac{\pi}{2}(\alpha - 1)\right) \tag{197}$$

$$c_{0,J+m} = 2\sin\left(\frac{\pi}{2}(\alpha - 1)\right) \tag{198}$$

so $c_{0,J} F_J(-1) + c_{0,J+m} F_{J+m}(-1) = 1$ and (193) for odd m is

$$\sum_{n=0}^{N-1} h_0(n) = 2\sum_{s=0}^{J-1} \sin\left(\frac{\pi}{4} + \beta_{0,s} + \Theta_s\right) + 1 = m \tag{199}$$

so we again obtain (194) through (196). Condition (149) with (182) cause the m × N $h_k(n)$ coefficient matrix to be a rank m, genus 2α wavelet matrix like the ones discussed at the beginning of this chapter.

THEOREM [Gopinath 8]: *For every integer m, there exists cosine modulated wavelet tight frames of length 2αm, that can be explicitly parameterized by J(α − 1) angles. In particular, for N = 2m, there exists a cosine modulated wavelet tight frame.*

Observe that the wavelet matrix has J less parameters than the Koilpillai parameterization on page 120 due to the linear constraint (182). Also note that while f(n) is symmetric, from (188) through (191), $h_k(n)$ is not a phase linear filter.

We now consider two examples from Gopinath's paper [8]. Let m = 2 and N = 4. Then $\alpha = 1$ and from (196)

$$\theta_{0,0} = \frac{\pi}{2} - \frac{\pi}{8} = \frac{3\pi}{8} \tag{200}$$

Therefore, the prototype filter f(n) is given by

$$\sqrt{2}\ f(n) = \begin{bmatrix} \cos\theta_{0,0} \\ \sin\theta_{0,0} \\ \sin\theta_{0,0} \\ \cos\theta_{0,0} \end{bmatrix} = \begin{bmatrix} \cos(3\pi/8) \\ \sin(3\pi/8) \\ \sin(3\pi/8) \\ \cos(3\pi/8) \end{bmatrix} = \begin{bmatrix} \sin(\pi/8) \\ \cos(\pi/8) \\ \cos(\pi/8) \\ \sin(\pi/8) \end{bmatrix} \tag{201}$$

The cosine modulation vector for the $c_{0,n}$ coefficients is

$$C_0 = 2\begin{bmatrix} \cos(-\pi/8) \\ \cos(\pi/8) \\ \cos(3\pi/8) \\ \cos(5\pi/8) \end{bmatrix} = 2\begin{bmatrix} \cos(\pi/8) \\ \cos(\pi/8) \\ \sin(\pi/8) \\ -\sin(\pi/8) \end{bmatrix} \tag{202}$$

and the corresponding scaling vector is

$$h_0 = \sqrt{2}\begin{bmatrix} \cos(\pi/8)\sin(\pi/8) \\ \cos^2(\pi/8) \\ \cos(\pi/8)\sin(\pi/8) \\ -\sin^2(\pi/8) \end{bmatrix} \tag{203}$$

The cosine modulation vector for the $c_{1,n}$ coefficients is

$$C_1 = 2\begin{bmatrix} -\cos(3\pi/8) \\ -\cos(3\pi/8) \\ \cos(\pi/8) \\ -\cos(\pi/8) \end{bmatrix} = 2\begin{bmatrix} -\sin(\pi/8) \\ -\sin(\pi/8) \\ \cos(\pi/8) \\ -\cos(\pi/8) \end{bmatrix} \tag{204}$$

and the corresponding wavelet vector is

$$h_1 = \sqrt{2}\begin{bmatrix} -\sin^2(\pi/8) \\ -\cos(\pi/8)\sin(\pi/8) \\ \cos^2(\pi/8) \\ -\cos(\pi/8)\sin(\pi/8) \end{bmatrix} \tag{205}$$

Observe that the wavelet vector is the QMF of the scaling vector.

$$h_1(n) = (-1)^{3-n} h_0(3 - n) \qquad n = 0, \ldots 3 \tag{206}$$

The scaling vector can be easily typed in so that you can plot and analyze the scaling function and wavelet. Observe that {1 1} perfectly divides the scaling vector once and the resulting wavelet has 0 regularity. This may be compared with the Daubechies minimal length scaling vector D_4 which can be divided by {1 1} twice and thus has regularity 1 (see page 59). The autocorrelation is

$$\left\{ \frac{1 - \sqrt{2}}{4} \quad 0 \quad \frac{3 + \sqrt{2}}{4} \quad 2 \quad \frac{3 + \sqrt{2}}{4} \quad 0 \quad \frac{1 - \sqrt{2}}{4} \right\} \tag{207}$$

Also observe that the cosine modulated scaling function and wavelet, while different from the D_4 ones, do have some similarities. For a 193 value approximation of the scaling and wavelet functions (n = 5), the normalized crosscorrelation is 3.4E–2 (29 dB) which, like the Daubechies ones, is not very orthogonal. This can be compared with the results on pages 66 and 101.

As we increase α, we obtain more design flexibility relative to the prototype f(n). This can be used to increase regularity or for a more desirable filter shape. Gopinath's second example illustrates increasing the regularity of the cosine modulated bank to 1 with an 8 coefficient system. For $\alpha = 2$ we have from (196) that

$$\Theta_0 = \theta_{0,0} + \theta_{0,1} = \pi - \frac{\pi}{8} = \frac{7\pi}{8} \tag{208}$$

and there is $J(\alpha - 1) = 1$ angle which may be chosen. For regularity, this can be chosen so that

$$\sum_{k=0}^{7} (-1)^k \, k \, h_0(k) = 0 \tag{209}$$

is satisfied or so that {1 2 1} divides the 8 coefficient polynomial as on page 60. The vectors are

$$\sqrt{2}\ f(n) = \begin{bmatrix} \cos\theta_{0,1}\ \cos\theta_{0,0} \\ -\cos\theta_{0,1}\ \sin\theta_{0,0} \\ \sin\theta_{0,1}\ \cos\theta_{0,0} \\ \sin\theta_{0,1}\ \sin\theta_{0,0} \\ \sin\theta_{0,1}\ \sin\theta_{0,0} \\ \sin\theta_{0,1}\ \cos\theta_{0,0} \\ -\cos\theta_{0,1}\ \sin\theta_{0,0} \\ \cos\theta_{0,1}\ \cos\theta_{0,0} \end{bmatrix} \qquad C = 2 \begin{bmatrix} -\sin(\pi/8) \\ \sin(\pi/8) \\ \cos(\pi/8) \\ \cos(\pi/8) \\ \sin(\pi/8) \\ -\sin(\pi/8) \\ -\cos(\pi/8) \\ -\cos(\pi/8) \end{bmatrix} \tag{210}$$

and the scaling vector is

$$h_0 = \sqrt{2} \begin{bmatrix} -\sin(\pi/8)\ \cos\theta_{0,1}\ \cos\theta_{0,0} \\ -\sin(\pi/8)\ \cos\theta_{0,1}\ \sin\theta_{0,0} \\ \cos(\pi/8)\ \sin\theta_{0,1}\ \cos\theta_{0,0} \\ \cos(\pi/8)\ \sin\theta_{0,1}\ \sin\theta_{0,0} \\ \sin(\pi/8)\ \sin\theta_{0,1}\ \sin\theta_{0,0} \\ -\sin(\pi/8)\ \sin\theta_{0,1}\ \cos\theta_{0,0} \\ \cos(\pi/8)\ \cos\theta_{0,1}\ \sin\theta_{0,0} \\ -\cos(\pi/8)\ \cos\theta_{0,1}\ \cos\theta_{0,0} \end{bmatrix} \tag{211}$$

Solving for the angle $\theta_{0,1}$, we obtain

$$\theta_{0,1} = \frac{1}{2}\left[\frac{\pi}{4} + \arcsin\left(\frac{\sqrt{2}+1}{4}\right)\right] \tag{212}$$

or

$$\theta_{0,1} = \frac{1}{2}\left[\frac{\pi}{4} + \pi - \arcsin\left(\frac{\sqrt{2}+1}{4}\right)\right] \tag{213}$$

where arcsin is assumed to return a value in $[-\pi/2, \pi/2]$. This combined with (208) completely determines the two angles. The resulting two angles $\theta_{0,1}$ are given by 0.716674224145 and 1.63952026605 and $\theta_{0,0} = 7\pi/8 - \theta_{0,1}$. The corresponding scaling vectors for these two cases are:

CASE 1	CASE 2
0.181677109432	0.0165461952688
-0.365384036868	0.033277255681
-0.382115097273	0.580338401984
0.768499439618	1.16716073216
0.318322890567	0.48345380473
0.158277255678	-0.240384036867
0.88211509728	-0.0803384019836
0.438607341563	0.0399460490234

Use QMF to compute the wavelet matrix and SOVN to compute the scaling function and wavelet. Case 1 results in a fractal like scaling function and wavelet, each having Fourier transforms which are not at all smooth, while case 2 produces a smooth scaling function and wavelet having also nice smooth spectrums. Case 2 resembles the Daubechies D_6 scaling function and wavelet which has regularity 2. Hence, regularity does not always imply smoothness. The above results are generalized in [14] unifying the [25] and [26] approaches.

COMMENTS ON THE LINEAR CONSTRAINT

All of the wavelets considered have had a linear constraint on the scaling vector coefficients. See (9) and (182), for example. Computing the Fourier transform of the two-scale difference equation

$$s(x) = \sum_{n=0}^{N-1} a_n \, s(mx - n) \tag{214}$$

gives

$$S(\omega) = \prod_{n=1}^{\infty} \left[\frac{H(e^{i\omega/m^n})}{m} \right] S(0) \tag{215}$$

where $S(\omega) = \mathcal{F}[\, s(x) \,](\omega)$ is the Fourier transform of the scaling function and

$$H(e^{i\omega}) = \sum_{n=0}^{N-1} a_n e^{-i\omega n} \tag{216}$$

This may be derived as follows.

$$\begin{aligned} S(\omega) &= \sum_{n=0}^{N-1} a_n \int_{-\infty}^{\infty} s(mx - n)\, e^{-i\omega x}\, dx \\ &= \sum_{n=0}^{N-1} a_n\, m^{-1} \int_{-\infty}^{\infty} s(y)\, e^{-i(\omega/m)(y+n)}\, dy \\ &= m^{-1} \sum_{n=0}^{N-1} a_n\, e^{-in\omega/m}\, S(\omega/m) \\ &= \frac{H(e^{i\omega/m})}{m} S(\omega/m) = \frac{H(e^{i\omega/m})}{m} \frac{H(e^{i\omega/m^2})}{m} S(\omega/m^2) \end{aligned} \tag{217}$$

Now suppose that $H(1) > m$. Then clearly, the above product diverges. Next suppose the $H(1) < m$. Then clearly, the above product is trivially equal to zero. Thus, the linear condition is associated with a meaningful Fourier transform existing as well as tight frames [9].

Observe that the alternative normalization $E^H(z^{-1})E(z) = I$ with the corresponding linear constraint

$$H(1) = \sum_{n=0}^{N-1} a_n = \sqrt{m} \tag{218}$$

also works since the denominator m in (215) is replaced by $\sqrt{m}$.

The linear condition is often ignored in the engineering literature in favor of only considering the perfect reconstruction properties of the filter bank. This gives more degrees of freedom for better filter design, αJ versus $(\alpha - 1)J$, for the cosine modulated filter bank. However, there is no wavelet tight frame unless the linear condition (9) and the orthogonality conditions (11) and (12) hold.

UNITARY FIR FILTER BANKS WITH SYMMETRY

Filters in a modulated filter bank cannot be phase linear [14]. However, if one takes an unmodulated approach for $m > 2$, then there is a solution. Soman [38] has parameterized paraunitary filter banks with linear phase. Gopinath has extended that characterization to several additional types of symmetry [13]. These include pairwise-shift symmetry

$$\begin{aligned} h_{m-1-k}(n) &= (-1)^n h_k(n) \\ H_{m-1-k}(z) &= H_k(-z) \end{aligned} \qquad \begin{aligned} k &= 0, \ldots, m-1 \\ n &= 0, \ldots, N-1 \end{aligned} \tag{219}$$

and pairwise-conjugated-shift symmetry

$$\begin{aligned} h_{m-1-k}(n) &= (-1)^n h_k(N-1-n) \\ H_{m-1-k}(z) &= z^{-(N-1)} H_k(-z^{-1}) \end{aligned} \qquad \begin{aligned} k &= 0, \ldots, m-1 \\ n &= 0, \ldots, N-1 \end{aligned} \tag{220}$$

These are considered with and without linear phase symmetry

$$\begin{aligned} h_k(n) &= \pm h_k(N-1-n) \\ H_k(z) &= \pm z^{-(N-1)} H_k(z^{-1}) \end{aligned} \qquad \begin{aligned} k &= 0, \ldots, m-1 \\ n &= 0, \ldots, N-1 \end{aligned} \tag{221}$$

Gopinath obtains parameterizations for all these cases. In addition, he shows how the linear condition (9) can be introduced so that one obtains wavelet tight frames. This theory is currently limited to the m even case, but it does allow one to construct M–band phase linear wavelets.

OTHER FILTER BANK DESIGN TECHNIQUES

In [12], Gopinath gives a computational procedure for solving for scaling vectors which can be viewed as the M–band generalization of the Vetterli complementary filter concept discussed in Chapter 9. Gopinath also develops a state space computational approach for solving for the M–band wavelet vectors.

In the remainder of this chapter, we will not be concerned with wavelet tight frames. We simply regard the impulse response of each filter in the bank (each row of the coefficient matrix) as discrete-time-wavelets. These wavelets may or may not be orthogonal and in essence give a new name to classical perfect reconstruction filter bank theory.

Nayebi [28] has developed an M–band procedure for computing the synthesis filter coefficients, given the analysis filter coefficients, which is optimum in the least squares sense. A conjugate gradient algorithm is used to iteratively improve the analysis filter design. Shenoy [33] gives an improved design procedure for multirate filters which is then compared with the more common Parks–McClellan design. Herley [23] has some new results on the 2–band design of IIR perfect reconstruction filter banks.

TIME-VARYING FILTER BANKS AND WAVELETS

Sodagar [35] has some new results on maintaining the perfect reconstruction property while the analysis bank is changed. Because of the filter bank decimation, a sequence of synthesis filters are required.

WAVELET PACKETS

Wavelet packets or nonuniform filter banks have been studied by many including Soman [36] and [37] who shows how tree structures can be related to parallel M-band bank structures.

[1] Crochiere, R. and Rabiner, L., *Multirate Digital Signal Processing*, Englewood Cliffs, NJ, Prentice-Hall, 1983.

[2] Daubechies, I., "Orthonormal Bases of Compactly Supported Wavelets," *Comm. Pure Appl. Math*, Vol. 41, 1988, 909-996.

[3] Daubechies, I., *Ten Lectures on Wavelets*, Philadelphia, PA, SIAM, 1992.

[4] Daubechies, I., and Lagaris, J., "Two-Scale Difference Equations I: Existence and Global Regularity of Solutions," *SIAM J. Math. Anal.*, Vol. 22, 1991, 1388-1410.

[5] Daubechies, I., and Lagaris, J., "Two-Scale Difference Equations II: Local Regularity, Infinite Products of Matrices and Fractals," *AT&T Bell Laboratories*, preprint, 1989.

[6] Doganata, Z., Vaidyanathan, P., and Nguyen, T., "General Synthesis Procedures for Perfect-Reconstruction Multirate Filter Bank Applications," *IEEE Trans. Acoustics, Speech, and Signal Proc.*, Vol. 36, No. 10, October, 1988, 1561-1573.

[7] Gopinath, R., *Wavelet Transforms and Time-Scale Analysis of Signals*, Masters Thesis, Rice University, 1990.

[8] Gopinath, R., and Burrus, C., "On Cosine-Modulated Wavelet Orthonormal Bases," Technical Report, Computational Mathematics Laboratory, Rice University, CML TR92-06.

[9] Gopinath, R., and Burrus, C., "Wavelet Transforms and Filter Banks," *Wavelets: Theory and Applications*, Chui, C., ed, Cambridge, Academic Press, 1991.

[10] Gopinath, R., and Burrus, C., "On Upsampling, Downsampling and Rational Sampling Rate Filter Banks," Technical Report, Computational Mathematics Laboratory, Rice University, CML TR91-25, November 25, 1991.

[11] Gopinath, R., and Burrus, C., "Oversampling Invariance of Wavelet Frames," Technical Report, Computational Mathematics Laboratory, Rice University, CML TR92-08.

[12] Gopinath, R., and Burrus, C., "State-Space Approach to Multiplicity M Orthonormal Wavelet Bases," Technical Report, Computational Mathematics Laboratory, Rice University, CML TR91-22, November 25, 1991.

[13] Gopinath, R., and Burrus, C., "Unitary FIR Filter Banks and Symmetry," Technical Report, Computational Mathematics Laboratory, Rice University, CML TR92-17, June 20, 1992.

[14] Gopinath, R., and Burrus, C., "Theory of Modulated Filter Banks and Modulated Wavelet Tight Frames," Technical Report, Computational Mathematics Laboratory, Rice University, CML TR92-18, June 22, 1992.

[15] Gopinath, R., and Burrus, C., "Biorthogonal Haar Wavelets and a Class of Multiplicity M Wavelet Frames,"Technical Report, Computational Mathematics Laboratory, Rice University, CML TR91-23, 1991.

[16] Harmuth, H., *Transmission of Information by Orthogonal Functions*, New York, Spring-Verlag, 1972.

[17] Heller, P., "Higher Rank Daubechies Wavelets - Preliminary Report," Aware Technical Report AD911204, Cambridge, MA, 1991.

[18] Heller, P., "Regular M-Band Wavelets," Aware Technical Report AD920608, Cambridge, MA, 1992.

[19] Heller, P., "The Perfect Reconstruction Condition for Higher Multiplier Wavelets," Aware Technical Report AD910408, Cambridge, MA, 1991.

[20] Heller, P., "A Construction of Higher Multiplier Wavelet Matrices," Aware Technical Report AD910728, Cambridge, MA, 1991.

[21] 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.

[22] Heller, P., private communication.

[23] Herley, C. and Vetterli, M., "Wavelets and Recursive Filter Banks," Submitted to *IEEE Trans. on Signal Proc.*, March 1992.

[24] Herrmann, O., "On the Approximation Problem in Nonrecursive Digital Filter Design," *IEEE Trans. Circuit Theory*, Vol. CT-18, 1971, 411-413.

[25] Koilpillai, R., and Vaidyanathan, P., "New Results on Cosine-Modulated FIR Filter Banks Satisfying Perfect Reconstruction," California Institute of Technology, Pasadena, CA, November, 1990.

[26] Malvar, H., "Modulated QMF Filter Banks with Perfect Reconstruction," *Electronics Letters*, Vol. 26, June, 1990, 906–907.

[27] Mintzer, F., "On Half-Band, Third Band and Nth Band FIR Filters and Their Design," *IEEE Trans. Acoustics, Speech, and Signal Proc.*, Vol. ASSP-30, October, 1982, 734–738.

[28] Nayebi, K., Barnwell, T., and Smith, M., "Time-Domain Filter Bank Analysis: A New Design Theory," *IEEE Trans. on Signal Proc.*, Vol 40, No. 6, June, 1992, 1412–1429.

[29] Pollen, D., "Linear One Dimensional Scaling Functions," Aware Technical Report AD900101.1.1, Cambridge, MA, 1990.

[30] Pollen, D. and Linden, D., "Real Quadratic One-Dimensional Scaling Coefficients," Aware Technical Report AD900212.1.1, Cambridge, MA, 1990.

[31] Resnikoff, H., "Linear Low-Pass Scaling Coefficients," Aware Technical Report AD910409, Cambridge, MA, 1991.

[32] Scheuermann, H. and Gockler, H., "A Comprehensive Survey of Digital Transmultiplexing Methods," *Proc. IEEE*, Vol. 69, No. 11, November, 1981, 1419–1450.

[33] Shenoy, R., Burnside, D., and Parks, T., "Linear Periodic Systems and Multirate Filter Design," Research Note GEO-002 92 16b, Schlumberger-Doll Research, September 1, 1992. Submitted to *IEEE Trans. ASSP* Special Issue on Wavelets, Time-Frequency Analysis, and Multirate Filterbanks.

[34] Smith, M., and Barnwell, T., "A New Filter Bank Theory for Time-Frequency Representation," *IEEE Trans. ASSP*, 35, No. 3, March 1987, 314–327.

[35] Sodagar, I., Nayebi, K., and Barnwell, T., "Time-Varying Filter Banks and Wavelets," School of Electrical Engineering, Georgia Institute of Technology, Atlanta, GA.

[36] Soman, A. and Vaidyanathan, P., "On Orthonormal Wavelets and Paraunitary Filter Banks," Dept. of Electrical Engineering, California Institute of Technology, Pasadena, CA, June 5, 1991.

[37] Soman, A. and Vaidyanathan, P., "Paraunitary Filter Banks and Wavelet Packets," *Proc. ICASSP 1992*, Vol. 4, 397–400.

[38] Soman, A. and Vaidyanathan, P., "Linear Phase Paraunitary Filter Banks: Theory, Factorizations and Applications," Dept. of Electrical Engineering, California Institute of Technology, Pasadena, CA, May 1992.

[39] Steffen, P., "Closed Form Derivation of Orthogonal Wavelet Bases for Arbitrary Integer Scales," University of Erlangen–Nurnberg, Institute for Communication Theory, to be published.

[40] Strang, G., "Wavelets and Dilation Equations: A Brief Introduction," *SIAM Review*, 31, 1989, 614–627.

[41] Strang, G., "The Optimal Coefficients in Daubechies Wavelets," preprint MIT Math. Dept., 1991.

[42] Strang, G. and Fix, G., "A Fourier Analysis of the Finite Element Variational Method," CIME *Constructive Aspects of Functional Analysis*, Rome, Edizioni Cremonese, 1973, 793–840.

[43] Vaidyanathan, P., "Passive Cascaded-Lattice Structures for Low-Sensitivity FIR Filter Design, with Applications to Filter Banks," *IEEE Trans. in Circuits and Systems*, CAS Vol. 33, November, 1986, 1045–1064.

[44] Vaidyanathan, P., "Quadrature Mirror Filter Banks, M–Band Extensions and Perfect-Reconstruction Techniques," *IEEE ASSP Mag.*, Vol. 4, July, 1987, 4–20.

[45] Vaidyanathan, P., "Multirate Digital Filters, Filter Banks, Polyphase Networks, and Applications: a Tutorial," *Proc. IEEE*, Vol. 78, January, 1990, 56–93.

[46] Vaidyanathan, P. and Mitra, S., "Polyphase Networks, Block Digital Filtering, LPTV Systems, and Alias-Free QMF Banks: A Unified Approach Based on Pseudocirculants," *IEEE Trans. ASSP*, Vol. 36, March, 1988, 381–391.

[47] Vaidyanathan, P., Nguyen, T., Doganata, Z., and Saramaki, T., "Improved Technique for Design of Perfect Reconstruction FIR QMF Banks with Lossless Polyphase Matrices," *IEEE Trans. Acoustics, Speech, and Signal Proc.*, Vol. 37, No. 7, July, 1989, 1042–1056.

[48] Vaidyanathan, P. and Hoang, P., "Lattice Structures for Optimal Design and Robust Implementation of Two–Channel Perfect Reconstruction QMF Banks," *IEEE Trans. ASSP*, Vol. 36, No. 1, January, 1988, 81–94.

[49] Vetterli, M., "A Theory of Multirate Filter Banks," *IEEE Trans. Acoustics, Speech, and Signal Proc.*, Vol. ASSP–35, No. 3, March, 1987, 356–371.

[50] Vetterli, M. and Le Gall, D., "Perfect Reconstruction FIR Filter Banks: Some Properties and Factorizations," *IEEE Trans. Acoustics, Speech, and Signal Proc.*, Vol. 37, No. 7, July 1989, 1057–1071.

[51] Yip, P. and Rao, K., "Fast Discrete Transforms," *Handbook of Digital Signal Processing*, D. Elliot, ed., Academic Press, 1987, Chapter 6.

[52] Zou, H. and Tewfik, A., "Discrete Orthogonal M–Band Wavelet Decompositions," *Proc. of ICASSP*, 1992, Vol IV, 605–608.

[53] Zou, H. and Tewfik, A., "Parameterization of Compactly Supported Orthonormal Wavelets," to appear in *IEEE Trans. on Signal Processing*, April, 1993.

[54] Zou, H. and Tewfik, A., "A Theory of M–Band Orthogonal Wavelets," Submitted to *IEEE Trans. on Circuits and Systems.*

11

WAVELETS AND APPROXIMATIONS

INTRODUCTION

The past two chapters examined the computation of wavelets and their properties. Now we study the transform aspect of wavelets and how it compares with the Fourier transform. Programs are given for computing the discrete wavelet transform.

At the time of writing this chapter, several texts and books have become available. References [2], [3], and [5] present wavelets from the mathematics point of view. References [1] and [11] present wavelets from an engineering point of view. A nice overview of time-frequency signal representation is given in [8].

We begin by reviewing the short time Fourier transform and windows. Then the wavelet transform is defined. A program for computing wavelet moments is given as well as one for computing wavelets by Daubechies method which is closely related to the actual wavelet transform computation. Finally, programs for computing wavelet transforms are presented.

SHORT TIME FOURIER TRANSFORM

The short time Fourier transform (STFT) is a name given to a generic set of windowed Fourier transforms. In the continuous case, the STFT of f(t) may be defined as

$$F(\Omega, \tau) = \int_{-\infty}^{\infty} f(t)\, g_1^*(t - \tau)\, e^{-i\Omega t}\, dt \tag{1}$$

with inverse transform

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(\Omega\, \tau)\, g_2(\tau - t)\, e^{i\Omega t}\, d\Omega d\tau \tag{2}$$

We derive the conditions on $g_1(t)$ and $g_2(t)$ as follows

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(\alpha)\, e^{-i\Omega\alpha}\, g_1^*(\alpha - \tau)\, g_2(\tau - t)\, e^{i\Omega t}\, d\alpha d\Omega d\tau \tag{3}$$

Since

$$\frac{1}{2\pi}\int_{-\infty}^{\infty} e^{i\Omega(t - \alpha)}\, d\Omega = \delta(t - \alpha) \tag{4}$$

equation (3) reduces to

$$\begin{aligned} f(t) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(\alpha)\, g_1^*(\alpha - \tau)\, g_2(\tau - t)\, \delta(t - \alpha)\, d\alpha d\tau \\ &= f(t) \int_{-\infty}^{\infty} g_1^*(t - \tau)\, g_2(\tau - t)\, d\tau \end{aligned} \tag{5}$$

so the last integral in (5) must equal 1.

GABOR TRANSFORM

Consider an example where $g_1(t)$ and $g_2(t)$ are defined by

$$g_1(t) = C_1 e^{-\beta_1 t^2} \qquad g_2(t) = C_2 e^{-\beta_2 t^2} \tag{6}$$

A STFT with a Gaussian window is commonly called a *Gabor transform*. Command **GAUS** provides this window. From pages 548 and 550 of the MATHLIB manual, we have the transform relations

$$\mathcal{F}[\, e^{-\beta_1(t - \tau)^2}\,](\Omega) = \sqrt{\frac{\pi}{\beta_1}}\; e^{-\Omega^2/(4\beta_1)}\, e^{i\Omega t} \tag{7}$$

$$\mathcal{F}[\, e^{-\beta_2(\tau - t)^2}\,](\Omega) = \sqrt{\frac{\pi}{\beta_2}}\; e^{-\Omega^2/(4\beta_2)}\, e^{-i\Omega t} \tag{8}$$

so that the last integral in (5) equals

$$C_1 C_2 \frac{\pi}{\sqrt{\beta_1 \beta_2}} = 1 \tag{9}$$

and we may choose the coefficients as

$$C_1 = \sqrt{\frac{\beta_1}{\pi}} \qquad C_2 = \sqrt{\frac{\beta_2}{\pi}} \tag{10}$$

UNCERTAINTY PRINCIPLE

The Fourier transform uncertainty principle [9] states that if a function f(t) vanishes at infinity faster than $t^{-1/2}$

$$\lim_{t \to \pm\infty} \sqrt{t}\ f(t) = 0 \tag{11}$$

then $D_t D_\omega \geq \sqrt{(\pi/2)}$ and that equality only holds for Gaussian functions where

$$D_t^2 = E^{-1} \int_{-\infty}^{\infty} t^2\ |f(t)|^2\ dt \qquad \int_{-\infty}^{\infty} |f(t)|^2\ dt = E \tag{12}$$

$$D_\omega^2 = E^{-1} \int_{-\infty}^{\infty} \omega^2\ |F(\omega)|^2\ d\omega \qquad \int_{-\infty}^{\infty} |F(\omega)|^2\ d\omega = 2\pi E \tag{13}$$

and $F(\omega)$ is the Fourier transform of f(t). Thus, the better we isolate frequency (small D_ω), the poorer the isolation of time (large D_t). The Gabor Gaussian window attempts to make the best trade.

COMMENTS ON WINDOWS

The literature is filled with applications of windows. They can be very useful in system identification, spectral estimation, speech pitch and formant analysis, estimation of group delay and instantaneous frequency, and parameter estimation problems. They are also useful in filter design applications. MATHLIB provides 10 windows and others are easily programmed. See the KAISER example in MAN6 WIND.

However, a common misapplication is in optimum minimum probability of error demodulation theory in the presence of colored noise. While a window may improve one's ability to estimate the interference environment for adaptive interference cancellation, the same window introduces a time varying amplitude modulation onto the signal of interest which ultimately will increase the symbol error rate. In over 20 years of simulating and building optimum demodulators, I have never found windowing to improve the symbol error rate.

DISCRETE STFT

A special case of the above STFT is where we define

$$\Omega = k\Omega_0 \qquad \tau = n\tau_0 \tag{14}$$

so that the transform becomes

$$F[k, n] = \int_{-\infty}^{\infty} f(t)\, g_1^*(t - n\tau_0)\, e^{ik\Omega_0 t}\, dt \tag{15}$$

with corresponding inverse transform

$$f(t) = \frac{\Omega_0\tau_0}{2\pi} \sum_{k=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} F[k, n]\, e^{ik\Omega_0 t}\, g_2(t - n\tau_0) \tag{16}$$

DISCRETE TIME STFT

If in addition, f(t) has been Nyquist sampled (without loss in generality we assume at a sampling rate of 1 Hz), then as explained on page 559 of the MATHLIB manual, (12) can be rewritten as

$$F[k, n] = \sum_{l=-\infty}^{\infty} f[l]\, g_1[l - n]\, e^{-i2\pi kl/M} \qquad k = 0, \ldots, M - 1 \tag{17}$$

where we now assume that the window $g_1[k]$ is real with compact support.

CONTINUOUS WAVELET TRANSFORM

Let $\psi(t)$ denote the *mother wavelet* function and define

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\, \psi\left(\frac{t - b}{a}\right) \tag{18}$$

whose Fourier transform is given by

$$\Psi_{a,b}(\Omega) = \sqrt{a}\, \Psi(a\Omega)\, e^{-ib\Omega} \tag{19}$$

The continuous wavelet transform (CWT) is defined as

$$W_f(a, b) = \int_{-\infty}^{\infty} \psi_{a,b}(t)\, f(t)\, dt = \langle \psi_{a,b}, f(t) \rangle \tag{20}$$

where the $\langle \bullet, \bullet \rangle$ denotes inner or dot product. The inverse transform is given by

$$f(t) = \frac{1}{C_\psi} \int_{-\infty}^{\infty} \int_0^{\infty} \frac{da\, db}{a^2} W_f(a, b)\, \psi_{a,b}(t) \tag{21}$$

where

$$C_\psi = \int_0^\infty \frac{|\Psi(\Omega)|^2}{\Omega}\, d\Omega \tag{22}$$

Non-negative **a** is the scaling parameter and **b** is the translation parameter. A wavelet is said to be *admissible* if $C_\psi < \infty$ which implies

$$\Psi(0) = \int_{-\infty}^{\infty} \psi(t)\, dt = 0 \tag{23}$$

Comparing (1) with (20), we observe that the STFT is a function of a *modulation* parameter Ω and a *translation* parameter τ, while the CWT is a function of a *dilation* or *scaling* parameter **a** and a translation parameter **b**. While the values of D_t and D_ω are constant throughout the (Ω, τ) plane, they are not for the CWT in the (a, b) plane. To see this, suppose $\psi(t)$ is centered at (t_0, Ω_0) and define [1]

$$\sigma_t^2 = E^{-1} \int_{-\infty}^{\infty} (t - t_0)^2\, |\psi(t)|^2\, dt \tag{24}$$

$$\sigma_\Omega^2 = E^{-1} \int_{-\infty}^{\infty} (\Omega - \Omega_0)^2\, |\Psi(\Omega)|^2\, d\Omega \tag{25}$$

Then $\psi_{a,b}(t)$ is centered at $(b + at_0, \Omega_0/a)$ and

$$D_t^2 = E^{-1} \int_{-\infty}^{\infty} (t - b - at_0)^2\, |\psi_{a,b}(t)|^2\, dt = a^2\, \sigma_t^2 \tag{26}$$

$$D_\Omega^2 = E^{-1} \int_{-\infty}^{\infty} \left(\Omega - \frac{\Omega_0}{a}\right)^2 |\Psi_{a,b}(\Omega)|^2\, d\Omega = \frac{\sigma_\Omega^2}{a^2} \tag{27}$$

which corrects (5.28) in [1].

An example mother wavelet is the second derivative of the Gaussian sometimes called the Mexican hat wavelet [5]

$$\psi(t) = (1 - t^2)\, e^{-t^2/2} \qquad \Psi(\Omega) = \sqrt{2\pi}\ \Omega^2\, e^{-\Omega^2/2} \tag{28}$$

which corrects (5.25) in [1]. For the Gabor transform defined above

$$E = \int_{-\infty}^{\infty} |g(t)|^2\, dt = \frac{2\beta}{\pi} \int_{-\infty}^{\infty} e^{-2\beta t^2}\, dt = \sqrt{\frac{\beta}{2\pi}} = \frac{1}{2\pi} \int_{-\infty}^{\infty} |F(\Omega)|^2\, d\Omega \tag{29}$$

so

$$D_t^2 = \sqrt{\frac{2\pi}{\beta}} \int_{-\infty}^{\infty} t^2\, |f(t)|^2\, dt = \frac{1}{4\beta} \tag{30}$$

$$D_\Omega^2 = \sqrt{\frac{2\pi}{\beta}} \int_{-\infty}^{\infty} \Omega^2\, |F(\Omega)|^2\, d\Omega = 2\pi\beta \tag{31}$$

using the relationships in Chapter 5 of the MATHLIB manual and $D_t^2 D_\Omega^2 = \pi/2$. For the Mexican hat wavelet centered at (0, 0)

$$E = \int_{-\infty}^{\infty} |\psi(t)|^2 \, dt = \frac{1}{2\pi} \int_{-}^{\infty} |\Psi(\Omega)|^2 \, d\Omega = \frac{3\sqrt{\pi}}{4} \tag{32}$$

so

$$\sigma_t^2 = \frac{4}{3\sqrt{\pi}} \int_{-\infty}^{\infty} t^2 \, |\psi(t)|^2 \, dt = \frac{7}{6} \tag{33}$$

$$\sigma_\Omega^2 = \frac{4}{3\sqrt{\pi}} \int_{-\infty}^{\infty} \Omega^2 \, |\Psi(\Omega)|^2 \, d\Omega = 5\pi \tag{34}$$

and $D_t^2 D_\Omega^2 = 35\pi/6$ which is almost a factor of 12 larger than for the Gabor transform. Thus, after giving up 11 dB of performance, we can take what is left and steer it around using (26) and (27).

Most of the operations associated with the STFT are related to modulations and translations, which form the *Weil–Heisenberg* group. Similarly, the group of dilations and translations associated with wavelet transforms are called the *Affine* group [6].

DISCRETE WAVELET TRANSFORM

A special case of the above CWT is where we define

$$a = a_0^k = m^k \qquad b = n b_0 a_0^k = n m^k \tag{35}$$

and generalize to the M–band case

$$\psi_{j,k,n}(t) = m^{k/2} \, \psi_j(m^k t - n) \tag{36}$$

for $j = 0, \ldots, m - 1$ so that

$$f(t) = \sum_{j=0}^{m-1} \sum_{k=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} c_{j,k,n} \, \psi_{j,k,n}(t) \qquad c_{j,k,n} = \langle f(t), \, \psi_{j,k,n}(t) \rangle \tag{37}$$

which can also be written as [7]

$$f(t) = \sum_{n=-\infty}^{\infty} c_{0,0,n} \, \psi_{0,0,n}(t) + \sum_{j=1}^{m-1} \sum_{k=1}^{\infty} \sum_{n=-\infty}^{\infty} c_{j,k,n} \, \psi_{j,k,n}(t) \tag{38}$$

where

$$\psi_{0,k,n}(t) = \phi_{k,n}(t) = m^{k/2} \, \phi(m^k t - n) \tag{39}$$

is the scaling function and

$$\psi_j(t) = \sqrt{m} \sum_{k=0}^{mg-1} h_j(k)\ \psi_0(mt - k) \tag{40}$$

where h_j is the jth row of the associated wavelet matrix.

DISCRETE TIME WAVELET TRANSFORM

In a similar fashion to the discrete time STFT, by letting the variable $t = Tl$ for $l \in \mathbf{I}$, equations (36) through (40) can be converted to a discrete time form.

COMPUTATION OF WAVELET MOMENTS

From the above discussion, the wavelet coefficients are computed from the formula

$$c_{j,k,n} = \int_{-\infty}^{\infty} f(t)\ \psi_{j,k,n}(t)\ dt \tag{41}$$

In the previous two chapters we pointed out that while the Strang method of computation gives accurate values, for any reasonable investment in computation equivalent to computing the sine function 70,000 times, we only have two digits of orthogonality between the wavelets. This is because we did not compute enough values.

One way of circumventing this problem is to assume f(t) can be locally approximated by its Maclaurin series expansion so that (41) can be computed in terms of moments. This works because the moments can be exactly computed. From (10–26) through (10–28) we have

$$\mu(s, j) = \int_{-\infty}^{\infty} x^j\ \psi_s(x)\ dx \tag{42}$$

$$M(s, j) = \sum_{k=0}^{mg-1} k^j\ h_s(k) \tag{43}$$

$$\mu(s, j) = m^{-j-1} \sum_{r=0}^{j} \binom{j}{r} M(s, j - r)\ \mu(0, r) \tag{44}$$

Thus, for s = 0

$$\mu(0, j) = m^{-j-1} \sum_{r=0}^{j} \binom{j}{r} M(0, j - r)\ \mu(0, r) \tag{45}$$

which can be recursively solved for the moments of the scaling function. Given these, (44) will compute $\mu(s, j)$ for $j = 1, \ldots, m - 1$.

Given the wavelet matrix M and the depth J, WMOM in MAN7 WAVE FFT2D computes all the moments M(s, j) and $\mu(s, j)$ up to k^J and t^J.

```
« → M J « M SIZE EVAL M 1 EROW { 1 } { } → m N V
C D « 0 J FOR K 'D' 'Σ(n=0,N-1,n^K×V(n+1))' →NUM STO+
NEXT 1 J FOR K 'C'
'Σ(n=0,K-1,COMB(K,n)×D(K-n+1)×C(n+1))/(m^(K+1)-m)' →NUM
STO+ NEXT C →ROW 2 m FOR I M I EROW { } → W E
« 0 J FOR K 'E' 'Σ(n=0,N-1,n^K×W(n+1))' →NUM STO+
NEXT 0 J FOR K
'Σ(n=0,K,COMB(K,n)×E(K-n+1)×C(n+1))/m^(K+1)' →NUM NEXT
{ 1 } J 1 + + →ARRY RCMB 'D' E STO+ » NEXT D OBJ→
DROP m J 1 + 2 →LIST →ARRY » » »
```

Two matrices are returned. The moments t^n for n = 0, . . ., J are returned in column n + 1 of the matrix on Level 2 of the stack where each row of the matrix corresponds to the associated scaling function or wavelet. The moments k^n for n = 0, . . ., J are returned in column n + 1 of the matrix on Level 1 of the stack where each row of the matrix corresponds to the associated scaling or wavelet vector. The rank of the input wavelet matrix must be at least 2. For the Daubechies D_4 wavelet matrix with J = 3, the output matrices are:

$$\begin{bmatrix} 2 & 1.26795 & .80385 & -1.22243 \\ 0 & 0 & 1.73205 & 9.29423 \end{bmatrix} \tag{46}$$

$$\begin{bmatrix} 1 & .63397 & .40192 & .13109 \\ 0 & 0 & .21651 & .78678 \end{bmatrix} \tag{47}$$

As shown below, these moments can be propagated through the filter bank to compute the wavelet transform.

DAUBECHIES' WAVELET COMPUTATION METHOD

The Strang method of wavelet computation, used in the previous two chapters, gives exact values within numerical error. Daubechies uses a different approach to computing the scaling function and wavelets [4]. For a reasonable investment in computation, her technique yields good enough values for plotting, but relative to 10-digit precision, all of the values are wrong. The two advantages of her technique are that we need not solve for an eigenvector and secondly and most important, the values are consistently wrong in such a way that the scaling function and wavelet approximations are always mutually orthogonal.

Noting that from (10–9), $H_0(1) = m$, rewrite (10–215) as

$$\Phi(m\omega) = \prod_{n=0}^{\infty} \left[\frac{H(e^{i\omega/m^n})}{m} \right] \tag{48}$$

where we also assume that $\Phi(0) = 1$. Define the partial product

$$\Phi_{(K)}(m\omega) = \prod_{n=0}^{K} \left[\frac{H(e^{i\omega/m^n})}{m} \right] \tag{49}$$

Then

$$\Phi_{(1)}(m\omega) = \frac{H(e^{i\omega/m})}{m} \frac{H(e^{i\omega})}{m} \tag{50}$$

We now derive a practical implementation of (49) as a means of approximating the scaling function and wavelets. By the time scaling property of the Fourier transform discussed on page 548 of the MATHLIB manual, we have

$$\mathcal{F}\left[\frac{H_0(e^{i\omega/m})}{m} \right](t) = h_0(mt) \tag{51}$$

Define the filter impulse response

$$h_0(t) = \sum_k a_k^0 \, \delta(t - k) \tag{52}$$

Then it follows that

$$h_0(mt) = \sum_k a_k^0 \, \delta(mt - k) \tag{53}$$

Now $m \, \Phi_{(1)}(m\omega)$ is the transform of the convolution

$$\begin{aligned} [h_0(t) \otimes h_0(mt)](t) &= \sum_k \sum_{k'} a_k^0 \, a_{k'}^0 \int_{-\infty}^{\infty} \delta(\tau - k) \, \delta(m[t - \tau] - k') \, d\tau \\ &= \sum_k \sum_{k'} a_k^0 \, a_{k'}^0 \, \delta(mt - mk - k') \end{aligned} \tag{54}$$

For each k′ MOD m, there are mg + g – 1 values of t for which there is a discrete nonzero output value.

$$t = k + \frac{k'}{m} = k + \lfloor k'/m \rfloor + k' \text{ MOD } m \tag{55}$$

There are m sets of these values which are all interleaved making a total of

$$m[mg + g - 1] = m^2g + m(g - 1) \tag{56}$$

After two iterations, there are

$$m[m^2g + m(g - 1) + g - 1] = m^3g + (m^2 + m)(g - 1) \tag{57}$$

values and after J iterations

$$m^{J+1}g + \left\{\sum_{k=1}^{J} m^k\right\}(g - 1) \tag{58}$$

values. The following two programs demonstrate the practical implementation for m = 2.

```
SOVS: « → V J « V 1 2 VDEC V 2 2 VDEC → H0 H1 «
1 J START V H0 VMPY →ROW V H1 VMPY →ROW RCMB
   TRN OBJ→ EVAL × →ARRY 'V' STO NEXT V » » »
```

```
SOVW: « → V J « V 1 2 VDEC V 2 2 VDEC → H0 H1 «
      V QMFL 'V' STO 1 J START V H0 VMPY
   →ROW V H1 VMPY →ROW RCMB TRN OBJ→ EVAL ×
      →ARRY 'V' STO NEXT V » » »
```

where the wavelet expansion is based on the formula

$$\Psi_j(m\omega) = \frac{H_j(e^{i\omega})}{m}\Phi(\omega) = \prod_{n=1}^{\infty}\left[\frac{H(e^{i\omega/m^n})}{m}\right]\frac{H_j(e^{i\omega})}{m} \tag{59}$$

for j = 1, . . ., m - 1. V is the scaling vector and J is the depth of approximation. SOVW must be run in the MAN7 WAVE directory so that QMFL is available. The M-band version of this program is SOVJ given below and is in the MAN7 WAVE FFT2D directory.

```
« → M J « M SIZE EVAL OVER / → m g « M 1 EROW
      g m 2 →LIST RDM M→CL → C « 1 J FOR n
"CONVOLUTION " n + 3 DISP 1 m FOR K "WAVELET " K
   1 - + 4 DISP M K EROW → V « V C 1 GET VMPY
→ROW 2 m FOR L V C L GET VMPY →ROW RCMB NEXT
TRN OBJ→ EVAL × { 1 } SWAP + →ARRY IF K 1 > THEN
      RCMB END » NEXT 'M' STO NEXT M » » » »
```

where M is the wavelet matrix and J is the depth of approximation. The output matrix rows correspond to the wavelet matrix rows and the number of columns is given by (58).

The Strang method of solution implemented by SOVN required that we set up a matrix whose eigenvector determined the scaling function on the integers. Program CRN computes the dimension of that matrix. As a result, the SOVN output matrix may have extra zeros on the right. If you wish to overlay the plots of the output rows from SOVN and SOVJ, you can use the command **LZDEL** to (approximately) delete the trailing zeros of the SOVN output vectors (rows) before plotting.

Since the Daubechies method of computation only involves convolving the filter bank with itself, the orthogonality of the bank results in the orthogonality of the scaling function and wavelet approximations for all J. Given the output matrix from SOVJ for one of the Daubechies wavelet matrices computed by CWM, the sequence **< DUP TRN × >** will compute the correlation matrix which is, within numerical precision, a constant times the identity matrix. The fact that the exact Strang solution exhibits such poor orthogonality says that regardless of rank and regularity, wavelets are not smooth functions. However, by the Daubechies algorithm, we can construct smooth approximations to them. In the next section we will see that we compute the wavelet transform in a similar fashion to the SOVJ algorithm by iteratively convolving the wavelet moments computed in the previous section with the filter bank.

COMPUTATION OF THE WAVELET TRANSFORM

We now look at the computation of the wavelet transform. From (40) we have

$$m^{(k-1)/2}\,\psi_j(m^{k-1}t - n) = \sum_l h_j(l)\, m^{k/2}\,\psi_0(m[m^{k-1}t - n] - l) \quad \textbf{(60)}$$

which gives the identity

$$\psi_{j,k-1,n}(t) = \sum_l h_j(l)\,\psi_{0,k,mn+l}(t) \quad \textbf{(61)}$$

Now multiply both sides by some function f(t) and integrate. We have

$$c_{j,k-1,n} = \sum_l h_j(l)\, c_{0,k,mn+l} = \sum_l h_j(l - mn)\, c_{0,k,l} \quad \textbf{(62)}$$

Thus, given the scaling function coefficients $c_{0,k,n}$ defined by (41)

$$c_{0,k,n} = \int_{-\infty}^{\infty} f(t)\,\phi_{k,n}(t)\,dt = m^{k/2}\int_{-\infty}^{\infty} f(t)\,\phi(m^k t - n)\,dt \quad \textbf{(63)}$$

we can use repeated convolution with the filter bank (reversed in time) to compute the other scaling function and wavelet coefficients.

For $\phi(t)$ has compact support, then for sufficiently large k, $\phi(m^k t - n)$ approaches a Dirac delta function and we have the approximation

$$c_{0,k,n} = m^{-k/2} \int_{-\infty}^{\infty} f(t)\, \phi(m^k t - n)\, d(m^k t) \sim m^{-k/2} f(m^{-k} n) \tag{64}$$

so it common in the discrete case where $f(t) = f(nT)$ to take $T = m^{-k}$ and use the samples of f(t) as the starting coefficients [7].

Alternatively, suppose that f(t) can be locally approximated by its Maclaurin series $\Sigma\, d_l\, t^l$. Then we have

$$\begin{aligned} c_{0,k,n} &= m^{-k/2} \sum_l d_l \int_{-\infty}^{\infty} (m^{-k}[x + n])^l\, \phi(x)\, dx \\ &= m^{-k/2} \sum_l d_l\, m^{-kl} \sum_{q=0}^{l} \binom{l}{q} n^{l-q} \int_{-\infty}^{\infty} x^q\, \phi(x)\, dx \end{aligned} \tag{65}$$

where the last integral is $\mu(0, q)$.

Observe that if we choose the discrete approximation (64) and define $x[n] = m^{k/2} f(m^{-k}n)$, except for the sign convention for n and r, (62) is the same as (10–83) discussed on page 102 which we called the rank M generalized wavelet transform. Index k in (62) keeps track of how many times we run the data through the bank which determines the multiresolution decomposition of x[n] as explained below.

From (40) we have the identity for $j = 0, \ldots, m - 1$

$$\begin{aligned} &\int_{-\infty}^{\infty} \psi_j(x)\, \psi_0(mx - n)dx \\ &\quad = \sqrt{m} \sum_l h_j(l) \int_{-\infty}^{\infty} \psi_0(mx - l)\, \psi_0(mx - n)\, dx \\ &\quad = \sqrt{m}\; h_j(n) \end{aligned} \tag{66}$$

so we have from (41) and (37)

$$\begin{aligned} c_{0,k,s} &= \int_{-\infty}^{\infty} f(t)\, m^{k/2}\, \psi_0(m^k t - s)\, dt \\ &= \sum_{j=0}^{m-1} \sum_r \sum_n c_{j,r,n} \int_{-\infty}^{\infty} m^{r/2}\, \psi_j(m^r t - n)\, m^{k/2}\, \psi_0(m^k t - s)\, dt \end{aligned} \tag{67}$$

From (40)

$$\psi_j(m^r t - n) = \sqrt{m} \sum_q h_j(q)\, \psi_0(m^{r+1} t - mn - q) \tag{68}$$

so (67) is zero unless $k = r + 1$ and $s = mn + q$. Hence,

$$c_{0,k,s} = \sum_{j=0}^{m-1} \sum_{n=-\infty}^{\infty} h_j(s - mn)\, c_{j,k-1,n} \tag{69}$$

Alternatively, from (68) and (10–12)

$$\begin{aligned}
&\sum_{j=0}^{m-1} \sum_{n=-\infty}^{\infty} h_j(s - mn)\, \psi_{j,k-1,n}(t) \\
&\quad = \sum_{j=0}^{m-1} \sum_{n=-\infty}^{\infty} h_j(s - mn) \sum_r h_j(r)\, \psi_{0,k,mn+r}(t) \\
&\quad = \sum_{j=0}^{m-1} \sum_{n=-\infty}^{\infty} h_j(s - mn) \sum_r h_j(r - mn)\, \psi_{0,k,r}(t) \\
&\quad = \sum_r \psi_{0,k,r}(t) \left[\sum_{j=0}^{m-1} \sum_{n=-\infty}^{\infty} h_j(s - mn)\, h_j(r - mn) \right] \\
&\quad = \sum_r \psi_{0,k,r}(t)\, \delta_{sr} = \psi_{0,k,s}(t)
\end{aligned} \tag{70}$$

so multiplying by f(t) and integrating again gives (69). Gopinath gives the more general biorthogonal derivation [7].

Equation (62) defines the basic analysis recursion and (69) defines the basic synthesis recursion. These are implemented in the programs below. Consider the 2–band case. Let operator L represent the lowpass analysis operation and H be the corresponding highpass or wavelet operation as defined by (62). Assuming (64) we write

$$F_{0,1} = L\, F_{0,0} \qquad F_{1,1} = H\, F_{0,0} \tag{71}$$

to describe the analysis operation where $F_{j,r}$ denotes the jth output after r iterations. The corresponding synthesis operation defined by (69) may be written as

$$F_{0,0} = L^*\, F_{0,1} + H^*\, F_{1,1} \tag{72}$$

Thus,

$$F_{0,0} = [L^*\, L + H^*\, H] F_{0,0} \tag{73}$$

which is a statement of the Bezout equation discussed in Chapter 9. Now the wavelet transform iteratively splits the lowpass output into highpass and lowpass pairs. Thus,

$$F_{0,2} = L\, F_{0,1} = L\, L\, F_{0,0} \qquad F_{1,2} = H\, F_{0,1} = H\, L\, F_{0,0} \tag{74}$$

The next stage is

$$F_{0,3} = L\,F_{0,2} = L\,L\,L\,F_{0,0} \qquad F_{1,3} = H\,F_{0,2} = H\,L\,L\,F_{0,0} \tag{75}$$

Hence, the recomposition or synthesis is

$$F_{0,0} = L^*\{\,L^*[\,L^*\,F_{0,3} + H^*\,F_{1,3}\,] + H^*\,F_{1,2}\,\} + H^*\,F_{1,1} \tag{76}$$

which is the so called tree or pyramid decomposition and recomposition. It is implemented by the below demonstration programs.

The below program DEMO computes three rank 2 forward wavelet transforms. The corresponding partial inverse transform is then computed by IDWT3. IDWT3 is like the inverse wavelet transform program IDWT2, but the final summation is left out so you can see the result. Program CLEN gets rid of the leading and trailing zeros.

```
« 3 { SOLVE } EVAL D2G { FFT2D } EVAL QMF 0 1 1 16
RNDN → M V « V M DWT2 M DWT2 M DWT2 M IDWT3
DUP2 PLT1 PLT3 + CLEN M IDWT3 OVER PLT3 + CLEN
M IDWT3 OVER PLT3 + CLEN V - » »
```

The above program computes a genus 3, rank 2 wavelet matrix and a length 16 random vector. It then calls DWT2 three times to compute the wavelet transform defined by (75). Then it calls IDWT3 to perform the recomposition defined by (76). The following program computes the forward transform.

```
DWT2: « → V M « "FORWARD DWT" 3 DISP V M 2
EROW VREV VMPY 2 2 VDEC V M 1 EROW VREV VMPY
2 2 VDEC » »
```

DWT2 convolves the input V with each of the reversed rows of the wavelet matrix and decimates the result by a factor of 2. At each stage of the recomposition, the Level 1 IDWT3 output contains the lowpass smoothed approximation (the *blurry* representation of V) and Level 2 contains the highpass approximation error generally called the *detail.* After plotting, DEMO performs the required addition.

```
IDWT2: « → V2 V1 M « "INVERSE DWT" 3 DISP V2 2
VINTP M 2 EROW VMPY V1 2 VINTP M 1 EROW VMPY +
2 / » »
```

```
IDWT3: « → V2 V1 M « "INVERSE DWT" 3 DISP V2 2
VINTP M 2 EROW VMPY 2 / V1 2 VINTP M 1 EROW
VMPY 2 / » »

CLEN: « → V « V 10 RND LZDEL VREV LZDEL VREV » »
```

After completing the recomposition, DEMO subtracts the original vector V to show the difference within roundoff error is zero. All of these programs are available in the MAN7 WAVE FFT2D directory. DEMO does directory switching so if you change WAVE, it may no longer work. For larger filters and data, use the FFT convolver as discussed in [10].

DWT2 is not an efficient implementation because half of the computed values are decimated. A better implementation is as follows [10].

```
« → V M « "FORWARD DWT" 3 DISP V 1 2 VDEC V 2 2
VDEC M REV→ CDLV DUP SIZE 2 GET 2 / CSPLT → V1 V2
M2 M1 « V1 M1 2 EROW VMPY V2 M2 2 EROW VMPY +
V1 M1 1 EROW VMPY V2 M2 1 EROW VMPY + » » »
```

Similarly, IDWT2 is not efficient since we interpolate and hence, we waste multiplies on zero values. A more efficient implementation is

```
« → V2 V1 M « "INVERSE DWT" 3 DISP M CDLV DUP
SIZE 2 GET 2 / CSPLT → M2 M1 « V2 M2 2 EROW
VMPY V2 M1 2 EROW VMPY VCMB →ROW CNLV V1 M2
1 EROW VMPY V1 M1 1 EROW VMPY VCMB →ROW CNLV
+ →VTR 2 / » » »
```

These techniques generalize to the M-band case such as in the SOVJ program.

MULTIRESOLUTION AND WAVELET PACKETS

As equations (71) through (76) demonstrate, the basic wavelet decomposition sequentially decomposes the "DC" output from the previous stage. A multiresolution analysis consists of a sequence of closed subspaces $\{V_m | m \in \mathbb{I}\}$ of $L^2(\mathbb{R})$. In our above example, we used the operators L and H to orthogonally decompose the given space. Each subsequent space was a subset of the previous space. In practice, one applies the operators L and H until further decomposition of the "DC" band is no longer fruitful.

The M-band generalization, creates a collection of bandpass outputs plus the "DC" output. At each stage, it is only the "DC" output which continues to be decomposed. The wavelet packet concept says that given the four outputs $F_{0,3}$, $F_{1,3}$, $F_{1,2}$, and $F_{1,1}$ as defined in (76), one can proceed to define new multiresolution decompositions for each of these four outputs and that the decompositions need not be the same. Thus, the nonuniform filter bank concept.

The quality of the wavelet decomposition and representation obviously depends on which wavelet one chooses. This presentation and software has mostly focused on the Daubechies wavelets. However, many researchers are involved with searching the infinite space of wavelets attempting to find the optimum one for specific applications. Others simply view wavelets as alternative filter bank design techniques for subband encoders.

[1] Akansu, A. and Haddad, R., *Multiresolution Signal Decomposition*, Boston, Academic Press, 1992.

[2] Chui, C., *An introduction to Wavelets*, Boston MA, Academic Press, 1992.

[3] Chui, C., (ed.), *Wavelets: A Tutorial in Theory and Applications*, Boston MA, Academic Press, 1992.

[4] Daubechies, I., "Orthonormal Bases of Compactly Supported Wavelets," *Comm. Pure Appl. Math*, Vol. 41, 1988, 909-996.

[5] Daubechies, I., *Ten Lectures on Wavelets*, Philadelphia, PA, SIAM, 1992.

[6] Gopinath, R., *Wavelet Transforms and Time-Scale Analysis of Signals*, Masters Thesis, Rice University, 1990.

[7] Gopinath, R., and Burrus, C., "Wavelet Transforms and Filter Banks," *Wavelets: Theory and Applications*, Chui, C., ed, Cambridge, Academic Press, 1991.

[8] 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.

[9] Papoulis, A., *The Fourier Integral and its Applications*, New York, McGraw-Hill, 1962.

[10] Rioul, O. and Duhamel, P., "Fast Algorithms for Discrete and Continuous Wavelet Transforms," *IEEE PGIT*, Vol. 38, No. 2, March 1992, 569-586.

[11] Vaidyanathan, P., *Multirate Systems and Filter Banks*, Englewood Cliffs, NJ, Prentice Hall, 1993.

12

OTHER INCLUDED SOFTWARE

INTRODUCTION

There are a number of programs which have been donated to the HP calculator bulletin board. Many have been collected by Joseph Horn and are distributed by EduCALC on *Goodies Disks*. The programs are free and provided as is and without any support from anyone. You use them at your own risk. EduCALC may be reached at 1 (800) 535–9650.

None of this software is under copyright and it may be copied and distributed without limitation.

Ken Cooke has taken the time to write a very fast machine language FFT program which I have included on the ROM as a convenience to those without PC download capability. If you bought this application package on disk, then the other programs discussed in this chapter are also on your disk as are the D6 and D7 programs discussed on page 90.

Again, *I did not write this software, I do not support it, and you use it at your own risk.* If you purchased this applications package on disk, the author's own description is also included. For questions about these programs, contact the authors.

FAST MACHINE LANGUAGE FFT AND CONVOLVER

Ken Cooke at kcooke@milton.u.washington.edu has written a very fast machine language FFT. It uses the same definition of the FFT as MATHLIB, is about a factor of 28 faster than my commands, and is done in 15-digit precision. The HP 48 library compiler has many limitations and mixing various program types is not included. If all the MATHLIB programs were the average size of these machine language FFT programs, MATHLIB would only contain 143 commands. There are five programs included in the MAN7 FFTML directory:

- FFTM forward FFT
- IFFTM inverse FFT
- ABSV array absolute value
- ANGL array angle (phase)
- CONVM linear convolution and correlation

FFTM and IFFTM do the twiddle computation internally and like **FFT** and **IFFT**, the input vector must have a size which is an integer power of 2. The ABSV program is equivalent of **VABS** and **MABS** and the ANGL program is equivalent to the **VARG** and **MARG** commands. ANGL returns the argument according to the current angle mode.

Program CONVM, which I wrote, is the equivalent of the **CONV** and **CCORR** commands. The first two inputs to CONVM are the same as these two commands and the outputs are the same as these commands. The third input to CONVM is a flag, 0 if you want the **CONV** output and 1 if you want the **CCORR** output. The FFT source code is available on Goodies disk 7. See also FFT.DOC on the disk.

By moving these programs to the HOME directory, you can use these programs by editing the application programs to call these five programs instead of the equivalent MATHLIB commands. This will improve the speed of your FFT applications.

LINEAR PROGRAMMING

George Chow has donated a set of linear programming programs which use the simplex method of solution. Since notation varies between texts, I will walk you through his example filling in a few points.

Maximize $3x_1 + 2x_2$ subject to the constraints $x_1 + 2x_2 \leq 3$, $2x_1 - x_2 \leq 2$,

$-x_1 + 2x_2 = 1$, $x_1 \geq 0$, and $x_2 \geq 0$. We need two slack variables x_3 and x_4 to turn the first two inequalities into equalities. However, this does not give us an identity submatrix so we do not have a starting feasible solution. Consequently, we add the artificial variable x_5 to get a starting feasible solution. The table with the right hand side (RHS) as the left most column according to his convention is

		x_1	x_2	x_3	x_4	x_5
x_3	3	1	2	1	0	0
x_4	2	2	-1	0	1	0
x_5	1	-1	2	0	0	1
	0	-3	-2	0	0	0

where the objective row, usually on the top, is the bottom row. The author's phase 0 is usually called phase 1. This step drives the artificial variable x_5 out of the basis. We do this by pivoting on either the -1 or the 2 in the third row where x_5 enters the basis. Following the author's example, we pivot on the -1 bringing x_1 into the basis and removing x_5 from the basis. The result is

		x_1	x_2	x_3	x_4	x_5
x_3	4	0	4	1	0	1
x_4	4	0	3	0	1	2
x_1	-1	1	-2	0	0	-1
	-3	0	-8	0	0	-3

The next step is to make the -1 in the left column non-negative. We do this by pivoting on the -2 in the third row which replaces x_1 with x_2 as a basis variable. The table is now

		x_1	x_2	x_3	x_4	x_5
x_3	2	2	0	1	0	-1
x_4	2.5	1.5	0	0	1	.5
x_2	.5	-.5	1	0	0	.5
	1	-4	0	0	0	1

Now we are ready for phase 2. We need to bring x_1 back into the basis. Following the example, we pivot on the 2 in the first row and second column. The result is

		x_1	x_2	x_3	x_4	x_5
x_1	1	1	0	.5	0	-.5
x_4	1	0	0	-.75	1	1.25
x_2	1	0	1	.25	0	.25
	5	0	0	2	0	-1

so the solution is $x_1 = x_2 = 1$ with maximum 5. See the SIMPLEX and SIMPLEX.DOC files.

3D PLOTS

Dave Jansen has written a 3D surface plotter. See the PLT3D.DOC and PLT3D files.

CALENDAR PROGRAM

CALDR is a modified version of a calendar program written by Eric B. Davis. The inputs are the month and year. For example,

```
< 12 1996 CALDR >
```

displays December, 1996.

APPENDIX A

WARRANTY AND USER SUPPORT

USER SUPPORT

You can get answers to your questions concerning the operation of the MATHLIB software from the author. If you do not find the information in this manual, the MATHLIB 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 and programs. 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.

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.

LIMITED WARRANTY AND DISCLAIMER

The author cannot and does 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. 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. The author 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 the author'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 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 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. 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.

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 the author.

APPENDIX B

IF YOUR SOFTWARE CAME ON A DISK

HP 48SX FILES

If you purchased your software on disk, you may download any of the HP 48SX program files to the calculator. Once the files have been loaded into your VAR directory, you may follow the instructions given in Chapter 1. The following files without a suffix can be downloaded.

MAN1	MAN2	LGDR
MAN3	MAN4	MAN5
MAN6	MAN7	SD1
SD2	SD3	MVER
D6	D7	SIMPLEX
PLT3D	CONS	CALDR

Follow the instructions in your HP 48SX owner's manual and those which came with your download hardware and Kermit software.

DOCUMENTATION FILES

The documentation files FFT.DOC, SIMPLEX.DOC, and PLT3D.DOC can be viewed on your computer.

APPENDIX C

CONSTANTS AND UNITS

If your software came on disk, the following are in the CONS directory.

NAME	VALUE	VAR*
Atomic mass unit (u)	$1.6605655 \times 10^{-27}$ kg	u
Avogadro's number (N)	6.0221367×10^{23} $(mol)^{-1}$	NA
Bohr magnetron (μ_B)	$9.2740154 \times 10^{-24}$ J•m²/Wb	μB
Boltzmann's constant ($k = R_0/N$)	$1.380658\ 10^{-23}$ J/K	k
Compton wavelength of electron ($h/m_e c$)	$2.42631058 \times 10^{-12}$ m	λc
Dirac's constant ($h/2\pi = \hbar$)	$1.05457266 \times 10^{-34}$ J•s	hbar
Electronic charge (e)	$1.60217733\ 10^{-19}$ C	q
Electron rest mass (m_e)	$9.1093897 \times 10^{-31}$ kg	me
Faraday constant ($\mathcal{F}$)	9.6485309×10^{4} C/mol	F
Fine structure constant ($\alpha = [\mu_0 c^2/(4\pi)][e^2/(\hbar c)]$)	$7.29735308 \times 10^{-3}$	α
First Bohr electron-orbit radius (a_0)	$5.29177249 \times 10^{-11}$ m	a0
Gravitation constant (G)	6.67259×10^{-11} N•m²/kg²	G
Ice–point (standard temperature)	273.1500 °K	StdT
Magnetic flux quantum ($\Phi = h/[2e]$)	$2.06783461 \times 10^{-15}$ Wb	φ
Neutron rest mass (m_n)	$1.6749543 \times 10^{-27}$ kg	mn
Nuclear magneton ($\mu_N = e\hbar/[2m_p]$)	$5.0507866 \times 10^{-27}$ J/T	μN
Permeability in free space (μ_0)	$4\pi \times 10^{-7}$ H/m	μ0
Permittivity in free space (ε_0)	$8.8541878176 \times 10^{-12}$ F/m	ε0
Planck constant (h)	$6.6260755 \times 10^{-34}$ J•s	h
Proton rest mass (m_p)	$1.6726231 \times 10^{-27}$ kg	mp
Radiation constant ($c_1 = 2\pi hc^2$)	$3.741832\ 10^{-16}$ W•m²	c1
Radiation constant ($c_2 = hc/k$)	1.438786×10^{-2} m•K	c2
Rydberg wave number for infinite mass (R_∞)	$1.0973731534 \times 10^{7}$ m^{-1}	R∞
Speed of light in vacuum ($c = [\mu_0 \varepsilon_0]^{-1/2}$)	$2.997924580 \times 10^{8}$ m/s	c
Speed of sound in dry air at 0°C	331.36 m/s	ss
Standard atmosphere (atm)	101,325 N/m²	StdP
Standard gravitational acceleration (g_s)	9.80665 m/s²	g
Standard volume of ideal gas (V_0)	22,4141 l•atm/mol	Vm
Stefan–Boltzmann constant ($\sigma = \pi^2 k^4/[60\hbar^3 c^2]$)	5.67051×10^{-8} W/m²/K⁴	σσ
Universal gas constant (R_0)	8.31451 J/(mol•K)	R
Wien displacement-law constant ($\lambda_{max}T$)	2.897756×10^{-3} m•K	c3

* Variable name in the constants directory CONS. $R_\infty = [\mu_0 c^2/(4\pi)]^2[m_e e^4/(4\pi\hbar^3 c)]$.
NOTE: mol = gmol = gram-mole ≠ g•mol and lbmol = pound-mole = 453.59237 mol.

INTERNATIONAL SYSTEM OF UNITS (SI UNITS)			
SYMBOL	NAME	QUANTITY	RELATIONS
A	ampere	electric current	basic unit
C	coulomb	electric charge	$A \cdot s$
cd	candela	luminous intensity	basic unit
F	farad	electric capacitance	$A \cdot s \cdot V^{-1}$
H	henry	electric inductance	$V \cdot s \cdot A^{-1}$
Hz	hertz	frequency	s^{-1}
J	joule	work, energy	$N \cdot m$
K	kelvin	temperature degree	basic unit
kg	kilogram	mass	basic unit
lm	lumen	luminous flux	$cd \cdot sr$
lx	lux	illumination	$lm \cdot m^{-2}$
m	meter	length	basic unit
mol	mole	amount of substance	basic unit
N	newton	force	$kg \cdot m \cdot s^{-2}$
Pa	pascal	pressure, stress	$N \cdot m^{-2}$
rad	radian	plane angle	supplementary unit
s	second	time	basic unit
S	siemens	electric conductance	Ω^{-1}
sr	steradian	solid angle	supplementary unit
T	tesla	magnetic flux density	$Wb \cdot m^{-2}$
V	volt	voltage, electromotive force	$W \cdot A^{-1}$
W	watt	power	$J \cdot s^{-1}$
Wb	weber	magnetic flux	$V \cdot s$
Ω	ohm	electric resistance	$V \cdot A^{-1}$

DECIMAL MULTIPLES AND FRACTIONS OF UNITS

FACTOR	PREFIX	SYMBOL	FACTOR	PREFIX	SYMBOL
10^{1}	deca	D or da	10^{-1}	deci	d
10^{2}	hecto	h	10^{-2}	centi	c
10^{3}	kilo	k	10^{-3}	milli	m
10^{6}	mega	M	10^{-6}	micro	μ
10^{9}	giga	G	10^{-9}	nano	n
10^{12}	tera	T	10^{-12}	pico	p
10^{15}	peta	P	10^{-15}	femto	f
10^{18}	exa	E	10^{-18}	atto	a

U.S. CUSTOMARY SYSTEM OF UNITS (FPS SYSTEM)			
SYMBOL	NAME	QUANTITY	RELATIONS
Btu	British thermal unit	work, energy	778.128 ft • lbf
deg	degree	plane angle	supplementary unit
°F	degree Fahrenheit	temperature	basic unit
ft	foot	length	basic unit
g	standard gravity	acceleration	32.174 ft • sec^{-2}
hp	horsepower	power	550 lbf • ft • sec^{-1}
lb	pound-mass	mass	lbf • g^{-1}
lbf	pound-force	force	basic unit
pd	poundal	force	lb • ft • sec^{-2}
sec	second	time	basic unit
sl	slug	mass	lbf • ft^{-1} • sec^2

METRIC SYSTEM OF UNITS (MKS SYSTEM)			
SYMBOL	NAME	QUANTITY	RELATIONS
°C	degree Celsius	temperature	basic unit
cal	calorie	work, energy	0.42665 m • kgf
deg	degree	plane angle	supplementary unit
dyn	dyne	force	10^{-6} kgf
erg	erg	work, energy	10^{-8} m • kgf
g	standard gravity	acceleration	9.80665 m • sec^{-2}
hp	horsepower	power	75 kgf • m • sec^{-1}
kg	killogram-mass	mass	kgf • g^{-1}
kgf	killogram-force	force	basic unit
m	meter	length	basic unit
sec	second	time	basic unit

See Chapter 13 of the *HP 48SX Owner's Manual* and pages 581 and 582 of the MATHLIB manual for units and conversions.

Cohen, R. and Taylor, B., "The 1986 CODATA Recommended Values of the Fundamental Physical Constants," In *Journal of Physical and Chemical Reference Data*, Vol. 17, No. 4, National Bureau of Standards, 1988.

HP Solve Equation Library Application Card, Hewlett Packard, 1990.

Tuma, J., *Handbook of Numerical Calculations in Engineering*, New York, McGraw-Hill, 1989.

INDEX