Math Pac
Quick Reference Guide

For Use With the HP-75

Contents

Real Scalar Functions	1
Base Conversions	3
Array Input and Output	4
Matrix Algebra	8
Real-Valued Matrix Functions	11
LU Decomposition	13
Solving a System of Equations	13
Complex Variables	14
Complex Functions	16
Complex Matrix Operations	20
Finding Roots of Polynomials	22
Solving $f(x) = 0$	22
Numerical Integration	23
Finite Fourier Transform	24
Error Conditions	24

Real Scalar Functions

Hyperbolic Functions

SINH(*numeric expression*)	**Hyperbolic Sine**
COSH(*numeric expression*)	**Hyperbolic Cosine**
TANH(*numeric expression*)	**Hyperbolic Tangent**
ASINH(*numeric expression*)	**Inverse Hyperbolic Sine**
ACOSH(*numeric expression*)	**Inverse Hyperbolic Cosine**
ATANH(*numeric expression*)	**Inverse Hyperbolic Tangent**

Logarithmic Functions

LOG2 — Base 2 Logarithm

```
LOG2(X)
```

where *X* is a numeric expression ($X > 0$).

LOGA — Variable Base Logarithm

```
LOGA(X,B)
```

where *X* is a numeric expression ($X > 0$) and
B is a numeric expression ($B > 0$ and $B \neq 1$).

Rounding and Truncating Functions

ROUND — Round

```
ROUND(X,N)
```

where *X*, *N* are numeric expressions.

If *N* is positive, rounds *X* to *N* digits to the right of the decimal point. If *N* is negative, rounds *X* to *N* digits to the left of the decimal point.

TRUNCATE — Truncate

```
TRUNCATE(X,N)
```

where *X*, *N* are numeric expressions.

If *N* is positive, truncates *X* to *N* digits to the right of the decimal point. If *N* is negative, truncates *X* to *N* digits to the left of the decimal point.

Factorial/Gamma Function

FACT **Combined Factorial and Gamma Functions**

> `FACT(`*X*`)`
>
> where *X* is a numeric expression not equal to a negative integer.

If X is a positive integer, returns $X!$ Otherwise, returns $\Gamma(X + 1)$.

Base Conversions

BSTR$ **Decimal to Binary, Octal, or Hexadecimal Conversion**

> `BSTR$(`*X*`,`*N*`)`
>
> where *X* is a numeric expression with a positive value less than 999,999,999,999.5 and *N* is a numeric expression with a rounded integer value of 2, 8, or 16.

Converts the rounded integer value of X (decimal) into the equivalent base N string.

BVAL **Binary, Octal, or Hexadecimal to Decimal Conversion**

> `BVAL(`*S$*`,`*N*`)`
>
> where *S$* is a binary, octal, or hexadecimal string expression and *N* is a numeric expression whose rounded integer value is 2, 8, or 16 respectively.

Converts a string expression *S$*, representing a number expressed in base N, into the equivalent decimal number. The value of the decimal equivalent can't exceed 999,999,999,999 (decimal).

Array Input and Output

REDIM

Redimensioning

```
REDIM  A(i)   [B(j)   ]...
       C(k,l) [D(m,n) ]
```

where **A**, **B** are vectors, and **C**, **D** are matrices, and *i*, *j*, *k*, *l*, *m*, *n* are numeric expressions.

Redimensions arrays and reassigns values in row order. A redimensioning subscript can be a numeric expression; its rounded integer value becomes the upper bound of the corresponding subscript.

The total number of elements in the redimensioned array can't exceed the total number of elements the array was given in a dimension statement.

Assignments

CON

Constant Array

MAT **A**=CON[(*redimensioning subscript(s)*)]

where **A** is an array.

Assigns a value of one to every element of **A**. If redimensioning subscripts are present, redimensions **A** just as REDIM would.

IDN

Identity Matrix

MAT **A**=IDN
MAT **B**=IDN(*X*,*Y*)

where **A** is a square matrix;
or **B** is a matrix and *X*, *Y* are numeric expressions with the same rounded integer value.

For a square matrix **A**, assigns a value of one to every element on the diagonal of **A** and assigns a value of zero to every other element.

For a matrix **B**, redimensions **B** to a square matrix with the upper bound of each subscript equal to the rounded integer value of *X* and *Y*; then assigns a value of one to every element on the diagonal of **B** and assigns a value of zero to every other element.

ZER — Zero Array

```
MAT A=ZER[(redimensioning subscript(s))]
```

where **A** is an array.

Assigns a value of zero to every element of **A**. If redimensioning subscripts are present, redimensions **A** just as `REDIM` would.

= — Simple Assignment

```
MAT A=B
```

where **A**, **B** are both vectors, or **A**, **B** are both matrices.

Redimensions **A** to be the same size as **B** and assigns to **A** the corresponding values from **B**.

=( ) — Numeric Expression Assignment

```
MAT A=(X)
```

where **A** is an array and *X* is a numeric expression.

Assigns the value of *X* to every element of **A**.

Array Input

INPUT — Assign Values From Keyboard Input

```
MAT INPUT A[,B] ...
```

where **A**, **B** are arrays.

Assigns values to the specified array(s) by prompting with the name of an array element and then accepting a number from the keyboard as the value of that element. For each array, prompts are given for the elements in row order; if there is more than one array, they are handled in the order specified.

READ — Assign Values From Data Statements

MAT READ **A**[, **B**] ...

where **A**, **B** are arrays.

Assigns values to the specified array(s) by reading from one or more DATA statements in the same program as the MAT READ statement. Operation is similar to READ keyword in the HP-75. For each array, elements are assigned values in row order; if there is more than one array, they are filled in the order specified.

Array Output

DISP — Display in Standard Format

MAT DISP **A** [, ; **B**]...[, ;]

where **A**, **B** are arrays.

Displays the values of the elements of the specified arrays. The values are displayed in row order. Each row begins on a new line; a blank line is displayed between the last row of an array and the first row of the next array. The choice of terminator—comma or semicolon—determines the spacing between the elements of an array.

Terminator		Spacing Between Elements
	Close:	Elements are separated by two spaces. A minus sign, if present, occupies one of the two spaces.
	Wide:	Elements are placed in 21-column fields.

If the last array specified doesn't have a terminator, the array will be displayed with wide spacing between elements.

PRINT

Print in Standard Format

```
MAT PRINT A [ , B ]...[ , ]
              ;            ;
```

where **A**, **B** are arrays.

Prints the values of the elements of the specified arrays. Operation is identical to `MAT DISP` except that the output is sent to the `PRINTER IS` device. (If no `PRINTER IS` device present, the output is sent to the `DISPLAY IS` device.)

DISP USING

Display Using Custom Format

```
                     format string
MAT DISP USING                        ;A
                     statement number

                                      [ , B ]...[ , ]
                                        ;           ;
```

where **A**, **B**, ..., are arrays.

Displays the values of the elements of the specified arrays in a format determined by the *format string* or by the specified `IMAGE` statement. (Refer to section 16 of the *HP–75 Owner's Manual* for a description of `DISP USING`, format strings, `IMAGE` statements, and their results.)

The values are displayed in row order. Each row begins on a new line; a blank line is displayed between the last row of an array and the first row of the next array.

The terminators between the arrays—commas or semicolons—serve only to separate the arrays and have no effect on the display format.

```
MAT PRINT USING  format string / statement number  ; A [ , ; B ] ... [ , ; ]
```

where **A**, **B** are arrays.

Prints the values of the elements of the specified arrays in a format determined by the *format string* or by the specified `IMAGE` statement. Operation is identical to that of `MAT DISP USING` except that the output is sent to the `PRINTER IS` device. (If no `PRINTER IS` device is present, the output is sent to the `DISPLAY IS` device.)

Matrix Algebra

Arithmetic

=− **Negation**

```
MAT A=-B
```

where **A**, **B** are both vectors or both matrices.

Redimensions **A** to be the same size as **B** and assigns to each element of **A** the negative of the corresponding element of **B**.

+ **Addition**

```
MAT A=B+C
```

where **A**, **B**, **C** are all vectors or all matrices, and **B**, **C** are conformable for addition.

Redimensions **A** to be the same size as **B** and **C**, and assigns to each element of **A** the sum of the values of the corresponding elements of **B** and **C**.

Subtraction

```
MAT A=B-C
```

where **A**, **B**, **C** are all vectors or all matrices, and **B**, **C** are conformable for addition.

Redimensions **A** to be the same size as **B** and **C**, and assigns to each element of **A** the difference of the values of the corresponding elements of **B** and **C**.

***** **Matrix Multiplication**

```
MAT A=B*C
```

where **B** is a matrix, **A**, **C** are both vectors or both matrices, and **B**, **C** are conformable for multiplication.

Redimensions **A** to have the same number of rows as **B** and the same number of columns as **C**, and assigns to **A** the values corresponding to the matrix **BC**.

( )* **Multiplication by a Scalar**

```
MAT A=(X)*B
```

where **A**, **B** are both vectors or both matrices, and *X* is a numeric expression.

Redimensions **A** to be the same size as **B** and assigns to each element of **A** the product of the value of *X* and the value of the corresponding element of **B**.

Operations

CROSS **Cross Product**

```
MAT A=CROSS(B,C)
```

where **B**, **C** are both vectors having three elements and **A** is a vector.

Redimensions **A** to have exactly three elements and assigns to **A** the values of the cross product **B** × **C**.

CSUM

Column Sum

```
MAT A=CSUM(B)
```

where **A**, **B** are arrays.

If **A** is a vector, redimensions **A** to have as many elements as there are columns in **B**; if **A** is a matrix, redimensions **A** to have as many columns as **B** and exactly one row. Assigns to each element in **A** the sum of the values in the corresponding column of **B**.

INV

Matrix Inverse

```
MAT A=INV(B)
```

where **A** is a matrix and **B** is a square matrix.

Redimensions **A** to be the same size as **B** and assigns to **A** the values of the matrix inverse of **B**.

RSUM

Row Sum

```
MAT A=RSUM(B)
```

where **A**, **B** are arrays.

If **A** is a vector, redimensions **A** to have as many elements as there are rows in **B**; if **A** is a matrix, redimensions **A** to have as many rows as **B** and to have exactly one column. Assigns to each element of **A** the sum of the values in the corresponding row of **B**.

TRN

Matrix Transpose

```
MAT A=TRN(B)
```

where **A**, **B** are matrices.

Redimensions **A** to be the same size as the matrix transpose of **B** and assigns to **A** the values of the matrix transpose of **B**.

Real-Valued Matrix Functions

Determinants

DET — Determinant

`DET(`**A**`)`

where **A** is a square matrix.

Returns the determinant of the matrix **A**.

DETL — Determinant of Last Matrix

`DETL`

Returns the determinant of the last matrix that was:

- Inverted in a `MAT...INV` statement.
- Decomposed in a `MAT...LUFACT` statement.
- Used as the first argument of a `MAT...SYS` statement.

`DETL` retains its value (even if your HP-75 is turned off) until another `MAT...INV`, `MAT...LUFACT`, or `MAT...SYS` statement is executed.

Matrix Norms

`ABSUM(`*array*`)` **Array Element Absolute Value Sum**

Returns the sum of the absolute values of all elements.

`AMAX(`*array*`)` **Array Element Maximum**

Returns the value of the maximum element.

`AMIN(`*array*`)` **Array Element Minimum**

Returns the value of the minimum element.

`CNORM(`*array*`)` **One-Norm (Column Norm)**

Returns the maximum value (over all columns) of the sums of the absolute values of all elements in a column.

`FNORM(`*array*`)` **Frobenius (Euclidean) Norm**

Returns the square root of the sum of the squares of all elements.

`MAXAB(`*array*`)` **Array Element Maximum Absolute Value**

Returns the value of the largest element (in absolute value).

`MINAB(`*array*`)` **Array Element Minimum Absolute Value**

Returns the value of the smallest element (in absolute value).

`RNORM(`*array*`)` **Infinity Norm (Row Norm)**

Returns the maximum value (over all rows) of the sums of the absolute values of all elements in a row.

`SUM(`*array*`)` **Array Element Sum**

Returns the sum of the values of all elements.

Inner Product

DOT

Inner (Dot) Product

> `DOT(`**X**`,`**Y**`)`
>
> where **X**, **Y** are vectors with the same number of elements.

Returns **X** • **Y**, the inner product of **X** and **Y**.

Subscript Bounds

LBND

Subscript Lower Bound

> `LBND(`**A**`,`*N*`)`
>
> where **A** is an array and *N* is a numeric expression whose rounded integer value is 1 or 2.

Returns the option base in effect when **A** was dimensioned. If **A** is a vector, `LBND(A,2)` = −1.

UBND

Subscript Upper Bound

```
UBND(A,N)
```

where **A** is an array and *N* is a numeric expression whose rounded integer value is 1 or 2.

Returns the upper bound of the *N*th (first or second) subscript of **A**. If **A** is a vector, `UBND(A,2)` = −1.

LU Decomposition

LUFACT

***LU* Decomposition**

```
MAT A=LUFACT(B)
```

where **A** is a matrix and **B** is a square matrix.

Redimensions **A** to be the same size as **B** and assigns to **A** the values of the *LU* decomposition of **B**:

- The elements in **A** that are above the diagonal have the same value as the corresponding elements in **U**.
- The elements in **A** that are on or below the diagonal have the same value as the corresponding elements in **L**.

Solving a System of Equations

SYS

System Solution

```
MAT X=SYS(A,B)
```

where **A** is a square matrix, **X**, **B** are both vectors or both matrices, and **A**, **B** are conformable for multiplication.

Redimensions **X** to be the same size as **B** and assigns to **X** the values that satisfy the matrix equation **AX**=**B**.

Complex Variables

Polar/Rectangular Conversions

CPTOR — Polar to Rectangular Conversion

```
MAT Z=CPTOR(A)
```

where **A** is an array with two elements and **Z** is an array.

Redimensions **Z** to be a complex scalar; then assigns to the first element of **Z** the real part, and to the second element of **Z** the imaginary part, of the complex number R exp $(i\theta)$, where R is the value of the first element of **A** and θ is the value of the second element of **A**.

θ will be interpreted as degrees or radians, according to the `OPTION ANGLE` in effect.

CRTOP — Rectangular to Polar Conversion

```
MAT A=CRTOP(Z)
```

where **Z** is a complex scalar and **A** is an array.

Redimensions **A** to be a complex scalar; then assigns to the first element of **A** the magnitude, and to the second element of **A** the angle, of the complex number $x + iy$, where x is the value of the first element of **Z** and y is the value of the second element of **Z**.

The angle will be given in degrees ($-180 < \theta \leqslant 180$) or in radians ($-\pi < \theta \leqslant \pi$) according to the `OPTION ANGLE` in effect.

Complex Arithmetic Operations

CADD — Complex Addition

```
MAT Z=CADD(W,U)
```

where **W**, **U** are complex scalars and **Z** is an array.

Redimensions **Z** to be a complex scalar and assigns to **Z** the values corresponding to the complex number **W** + **U**.

CDIV
Complex Division

```
MAT Z=CDIV(W,U)
```

where **W**, **U** are complex scalars (**U** ≠ (0, 0)) and **Z** is an array.

Redimensions **Z** to be a complex scalar and assigns to **Z** the values corresponding to the complex number **W/U**.

CMULT
Complex Multiplication

```
MAT Z=CMULT(W,U)
```

where **W**, **U** are complex scalars and **Z** is an array.

Redimensions **Z** to be a complex scalar and assigns to **Z** the values corresponding to the complex number **W** * **U**.

CONJ
Complex Conjugation

```
MAT Z=CONJ(W)
```

where **W** is a complex scalar and **Z** is an array.

Redimensions **Z** to be a complex scalar and assigns to **Z** the values corresponding to the complex conjugate of **W**.

CRECP
Complex Reciprocal

```
MAT Z=CRECP(W)
```

where **W** is a complex scalar (**W** ≠ (0, 0)) and **Z** is an array.

Redimensions **Z** to be a complex scalar and assigns to **Z** the values corresponding to the complex number 1/**W**.

CSUB
Complex Subtraction

```
MAT Z=CSUB(W,U)
```

where **W**, **U** are complex scalars and **Z** is an array.

Redimensions **Z** to be a complex scalar and assigns to **Z** the values corresponding to the complex number **W** − **U**.

Complex Functions

Simple Transcendental Functions

CCOS

Complex Cosine

```
MAT Z=CCOS(W)
```

where **W** is a complex scalar and **Z** is an array.

Redimensions **Z** to be a complex scalar and assigns to **Z** the values corresponding to the complex cosine of **W**.

CCOSH

Complex Hyperbolic Cosine

```
MAT Z=CCOSH(W)
```

where **W** is a complex scalar and **Z** is an array.

Redimensions **Z** to be a complex scalar and assigns to **Z** the values corresponding to the complex hyperbolic cosine of **W**.

CEXP

Complex Exponential

```
MAT Z=CEXP(W)
```

where **W** is a complex scalar and **Z** is an array.

Redimensions **Z** to be a complex scalar and assigns to **Z** the values corresponding to the complex exponential of **W**.

CSIN

Complex Sine

```
MAT Z=CSIN(W)
```

where **W** is a complex scalar and **Z** is an array.

Redimensions **Z** to be a complex scalar and assigns to **Z** the values corresponding to the complex sine of **W**.

CSINH

Complex Hyperbolic Sine

```
MAT Z=CSINH(W)
```

where **W** is a complex scalar and **Z** is an array.

Redimensions **Z** to be a complex scalar and assigns to **Z** the values corresponding to the complex hyperbolic sine of **W**.

CTAN

Complex Tangent

```
MAT Z=CTAN(W)
```

where **W** is a complex scalar and **Z** is an array.

Redimensions **Z** to be a complex scalar and assigns to **Z** the values corresponding to the complex tangent of **W**.

CTANH

Complex Hyperbolic Tangent

```
MAT Z=CTANH(W)
```

where **W** is a complex scalar and **Z** is an array.

Redimensions **Z** to be a complex scalar and assigns to **Z** the values corresponding to the complex hyperbolic tangent of **W**.

Inverse Functions

CACOS

Complex Inverse Cosine

```
MAT Z=CACOS(W)
```

where **W** is a complex scalar and **Z** is an array.

Redimensions **Z** to be a complex scalar and assigns to **Z** the complex principal value of the inverse cosine of **W**.

CACOSH — Complex Inverse Hyperbolic Cosine

```
MAT Z=CACOSH(W)
```

where **W** is a complex scalar and **Z** is an array.

Redimensions **Z** to be a complex scalar and assigns to **Z** the complex principal value of the inverse hyperbolic cosine of **W**.

CASIN — Complex Inverse Sine

```
MAT Z=CASIN(W)
```

where **W** is a complex scalar and **Z** is an array.

Redimensions **Z** to be a complex scalar and assigns to **Z** the complex principal value of the inverse sine of **W**.

CASINH — Complex Inverse Hyperbolic Sine

```
MAT Z=CASINH(W)
```

where **W** is a complex scalar and **Z** is an array.

Redimensions **Z** to be a complex scalar and assigns to **Z** the complex principal value of the inverse hyperbolic sine of **W**.

CATN — Complex Inverse Tangent

```
MAT Z=CATN(W)
```

where **W** is a complex scalar (**W** $\neq$ (0, 1) or (0, −1)) and **Z** is an array.

Redimensions **Z** to be a complex scalar and assigns to **Z** the complex principal value of the inverse tangent of **W**.

CATANH **Complex Inverse Hyperbolic Tangent**

```
MAT Z=CATANH(W)
```

where **W** is a complex scalar (**W** $\neq$ (1, 0) or (-1, 0)) and **Z** is an array.

Redimensions **Z** to be a complex scalar and assigns to **Z** the complex principal value of the inverse hyperbolic tangent of **W**.

CLOG **Complex Logarithm**

```
MAT Z=CLOG(W)
```

where **W** is a complex scalar (**W** $\neq$ (0, 0)) and **Z** is an array.

Redimensions **Z** to be a complex scalar and assigns to **Z** the complex principal value of the logarithm of **W**.

CPOWER **Complex Power**

```
MAT V=CPOWER(Z,W)
```

where **Z**, **W** are complex scalars (**Z** $\neq$ (0, 0) if Re (**W**) $\leqslant$ 0) and **V** is an array.

Redimensions **V** to be a complex scalar and assigns to **V** the complex principal value of $\mathbf{Z}^{\mathbf{W}}$.

CSQR **Complex Square Root**

```
MAT Z=CSQR(W)
```

where **W** is a complex scalar and **Z** is an array.

Redimensions **Z** to be a complex scalar and assigns to **Z** the complex principal value of the square root of **W**.

Roots of a Complex Number

CROOT — Roots of a Complex Number

```
MAT R=CROOT(Z,N)
```

where **R** is a matrix, **Z** is a complex scalar, and *N* is a numeric expression whose rounded integer value is positive.

Redimensions **R** to be an $P \times 2$ array (where P is the rounded integer value of N) and assigns to **R** all the values of $\mathbf{Z}^{(1/P)}$.

Complex Matrix Operations

CDET — Complex Determinant

```
MAT Z=CDET(A)
```

where **A** is a *square* complex matrix (twice as many columns as rows) and **Z** is an array.

Redimensions **Z** to be a complex scalar and assigns to **Z** the complex value of the determinant of the complex matrix represented by **A**.

CIDN — Complex Identity Matrix

```
MAT A=CIDN
```

where **A** is a *square* complex matrix (twice as many columns as rows).

Assigns to **A** the values of the complex identity matrix. **A** is not redimensioned.

CINV

Complex Matrix Inverse

```
MAT A=CINV(B)
```

where **B** is a *square* complex matrix (twice as many columns as rows) and **A** is a matrix.

Redimensions **A** to be exactly the same size as **B** and assigns to **A** the values of the matrix inverse of the complex matrix represented by **B**.

CMMULT

Complex Matrix Multiplication

```
MAT A=CMMULT(B,C)
```

where **B**, **C** are complex matrices such that there are twice as many columns in **B** as there are rows in **C**, and **A** is a matrix.

Redimensions **A** to have the same number of rows as **B** and the same number of columns as **C**, and assigns to **A** the values of the complex matrix product **BC**.

CSYS

Complex System Solution

```
MAT Z=CSYS(A,B)
```

where **A** is a *square* complex matrix (twice as many columns as rows), **B** is a complex matrix with the same number of rows as **A**, and **Z** is a matrix.

Redimensions **Z** to be exactly the same size as **B** and assigns to **Z** the complex values that solve the complex matrix equation **AZ** = **B**.

CTRN

Complex Conjugate Transpose

```
MAT A=CTRN(B)
```

where **B** is a complex matrix and **A** is a matrix.

Redimensions **A** to have half as many rows as **B** has columns and twice as many columns as **B** has rows—if **B** is an $N \times 2P$ matrix, **A** will be a $P \times 2N$ matrix. **A** will be assigned the values of the complex conjugate transpose of the complex matrix represented by **B**.

Finding Roots of Polynomials

PROOT — Roots of a Polynomial

```
MAT R=PROOT(P)
```

where **P** is an array with at least two elements and **R** is a matrix.

Redimensions **R** to be an $N \times 2$ array (where **P** has a total of $N + 1$ elements) and assigns to **R** the (complex) values of the solutions of the equation $P(x) = 0$ (where P is the polynomial of degree N whose coefficients are the values of the elements of **P**). The first column of **R** will contain the real parts of the roots and the second column will contain the imaginary parts.

Solving *f*(x) = 0

FNROOT — Function Root

FNROOT(*A*, *B*, *user-defined function*(*X*))

where *A*, *B* are numeric expressions and *X* is a numeric variable.

Returns the first value found (starting with guesses *A* and *B*) that is a root of the user-defined function or is the best approximation available. This keyword can be used only in a program.

FNGUESS — Previous Estimate of Function Root

```
FNGUESS
```

Returns the next-to-last value tried as a solution in the most recent FNROOT statement. FNGUESS retains its value (even if your HP-75 is turned off) until FNROOT is again executed.

Numerical Integration

INTEGRAL

Definite Integral

INTEGRAL(*A*, *B*, *E*, *user-defined function*(*X*))

where *A*, *B*, *E* are numeric expressions and *X* is a numeric variable.

Returns an approximation to the integral from A to B of the specified user-defined function. The relative error E (rounded to the range $1E-12 \leqslant E \leqslant 1$) indicates the accuracy of the user-defined function and is used to calculate the acceptable error in the approximation to the integral. This keyword can be used only in a program.

IVALUE

Last Result of INTEGRAL

IVALUE

Returns the last approximation computed by the INTEGRAL keyword. If the [ATTN] key was pressed or the operation of INTEGRAL was otherwise interrupted, then IVALUE returns the value of the current approximation to the integral. Otherwise, IVALUE returns the same value that INTEGRAL last returned. IVALUE retains its value (even if your HP-75 is turned off) until another INTEGRAL is computed.

IBOUND

Error Approximation for INTEGRAL

IBOUND

Returns the final error estimate for the definite integral most recently computed by INTEGRAL.

- A positive value for IBOUND means that the approximations converged.
- A negative value for IBOUND means that the approximations didn't completely converge, so that the value returned by INTEGRAL may not be within the acceptable error of the actual value.

Like IVALUE, IBOUND retains its value (even if the HP-75 is turned off) until another INTEGRAL is computed. Unlike IVALUE, the value of IBOUND has no relation to the current approximation to the integral if the operation of INTEGRAL is interrupted.

Finite Fourier Transform

FOUR **Finite Fourier Transform**

MAT **W**=FOUR(**Z**)

where **Z** is a $N \times 2$ matrix (N a non-negative integer power of 2) and **W** is a matrix.

Redimensions **W** to be exactly the same size as **Z** and assigns to **W** the complex values of the finite Fourier transform of the data points represented by **Z**.

Error Conditions

Number **Error Message and Condition**

1 num too small

- $|Result| < 1E-499$.

2 num too large

- $|Result| > 9.99999999999E499$.
- MAT U=INV(V), MAT U=CINV(V), MAT U=LUFACT(V), MAT U=SYS(V,W), MAT U=CSYS(V,W), DET(V), MAT U=CDET V.

  The matrix **V** is singular (that is, its determinant is zero) and the *LU* decomposition of **V** requires division of a non-zero number by zero. This does not always indicate that the results of the operation are invalid. In particular, the results of DET and CDET will be valid. The results of the other operations should be checked when this error occurs.

11 `arg out of range`

- `ACOSH(X)`: $X < 1$.
- `ATANH(X)`: $|X| > 1$.
- `LOGA(X,B)`: $B = 1$.
- `MAT Z=CDIV(W,V)`, `MAT Z=CRECP(V)`: $\mathbf{V} = (0, 0)$.
- `MAT Z=CPOWER(W,V)`: $\mathbf{W} = (0, 0)$ and $\mathrm{Re}(\mathbf{V}) \leqslant 0$.
- `BSTR$(M,N)`: $M \geqslant 999{,}999{,}999{,}999.5$.
- `BVAL(B$,N)`: $Result > 999{,}999{,}999{,}999$.

12 `LOG(0)`

- `LOG2(X)`: $X = 0$.
- `LOGA(X,B)`: $X = 0$ or $B = 0$.
- `MAT Z=CLOG(W)`: $\mathbf{W} = (0, 0)$.

13 `LOG(neg number)`

- `LOG2(X)`: $X < 0$.
- `LOGA(X,B)`: $X < 0$ or $B < 0$.

89 `bad parameter`

- `BVAL(B$,N)`, `BSTR$(M,N)`: rounded integer value of N not equal to 2, 8, or 16.
- `BVAL(B$,N)`: *B$* not a valid number in base N.
- `BSTR$(M,N)`: $M < 0$.
- `MAT A=IDN(`*redimensioning subscript(s)*`)`,
  `MAT A=CON(`*redimensioning subscript(s)*`)`,
  `MAT A=ZER(`*redimensioning subscript(s)*`)`,
  `REDIM A(`*redimensioning subscript(s)*`)`:

  rounded integer value of one or both subscripts is less than the option base in effect.
- `UBND(A,N)`, `LBND(A,N)`: rounded integer value of N not equal to 1 or 2.
- `MAT R=CROOT(P,N)`: rounded integer value of N not positive.

Number	Error Message and Condition
201	result dimension

- MAT A=CON(i,j), MAT A=ZER(i,j), MAT A=IDN(i,i), REDIM A(i,j): **A** singly subscripted.
- MAT A=CON(i), MAT A=ZER(i), REDIM A(i): **A** doubly subscripted.
- MAT A=*operation* (*operand array(s)*): number of subscripts of **A** not the same as the number of subscripts required for the result of the operation.

202 result size

- REDIM A(*redimensioning subscript(s)*), MAT A=CON(*redimensioning subscript(s)*), MAT A=ZER(*redimensioning subscript(s)*), MAT A=IDN(*redimensioning subscript(s)*):

  number of elements in the redimensioned array greater than the total number of elements given to it in a dimensioning statement.
- MAT A=*operation*(*operand array(s)*):

  total number of elements in **A** (as given in its original dimensioning statement) less than the number of elements needed to store the results of the operation.

203 conformability

- MAT A=B+C, MAT A=B-C: **B** and **C** not *conformable for addition* (the number of rows are unequal or the number of columns are unequal).
- MAT A=B*C, MAT X=SYS(B,C): **B** and **C** not *conformable for multiplication* (the number of columns of **B** is not equal to the number of rows of **C**).
- DOT(A,B): number of elements of **A** not equal to the number of elements of **B**.

- `MAT R=CMMULT(A,B)`, `MAT X=CSYS(A,B)`: number of columns of **A** not equal to twice the number of rows of **B**.

204 `not square`

- `DET(A)`, `MAT X=SYS(A,B)`, `MAT B=INV(A)`, `MAT B=LUFACT(A)`, `MAT A=IDN`:

  number of rows of **A** not equal to the number of columns.

- `MAT A=IDN(i,j)`: $i \neq j$.
- `MAT R=CINV(A)`, `MAT R=CSYS(A,B)`, `MAT A=CIDN`, `MAT B=CDET(A)`:

  number of columns of **A** not equal to twice the number of rows.

205 `not vector`

- `MAT X=CROSS(A,B)`, `DOT(A,B)`: **A** or **B** not singly subscripted.

206 `not 3-vector`

- `MAT X=CROSS(A,B)`: **A** or **B** not three dimensional.

207 `operand dimension`

- `MAT A=IDN(i)`: only one redimensioning subscript specified.
- `DET(B)`, `MAT A=CDET(B)`, `MAT X=SYS(B,C)`, `MAT A=LUFACT(B)`, `MAT A=TRN(B)`, `MAT A=CTRN(B)`, `MAT A=CINV(B)`, `MAT A=INV(B)`, `MAT A=FOUR(B)`:

  B not doubly subscripted.

- `MAT R=CMMULT(A,B)`, `MAT R=CSYS(A,B)`:

  A or **B** not doubly subscripted.

208 `operand size`

- `MAT R=`*complex function*`(Z)`: **Z** not a complex scalar.
- `MAT R=`*complex function*`(Z,W)`: **Z** or **W** not a complex scalar.
- `MAT R=CROOT(Z,N)`: **Z** not a complex scalar.
- `MAT R=PROOT(P)`: **P** contains exactly one element (and so represents a polynomial of degree zero).
- `MAT R=CDET(A)`, `MAT R=CINV(A)`, `MAT R=CTRN(A)`: **A** doesn't have an even number of columns.
- `MAT R=CMMULT(A,B)`, `MAT X=CSYS(A,B)`: **A** or **B** doesn't have an even number of columns.
- `MAT A=FOUR(B)`: **B** is not an $N \times 2$ array with N a non-negative integer power of 2.

209 `PROOT failure`

- `PROOT` cannot find a root of the specified polynomial.

210 `nesting error`

- `FNROOT(A,B,`*user-defined function*`(X))`: user-defined function uses the `FNROOT` keyword in its definition.
- `INTEGRAL(A,B,E,`*user-defined function*`(X))`: user-defined function uses the `INTEGRAL` keyword in its definition.

HEWLETT PACKARD

Portable Computer Division

1000 N.E. Circle Blvd., Corvallis, OR 97330, U.S.A.

00075-90107 English Printed in U.S.A. 5/83