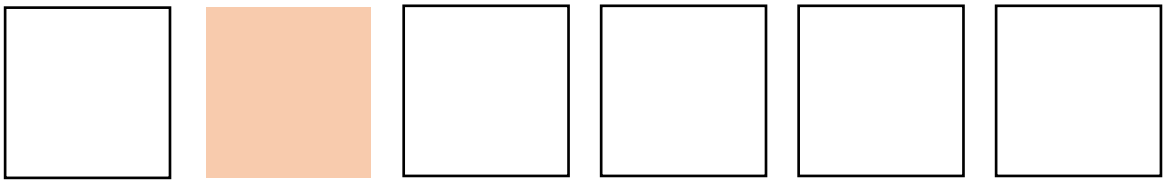
Math Pac 2

Owner's Manual Supplement

For the HP-71



Math Pac 2

Owner's Manual Supplement

For Use with the HP-71

July 2020

NOTICE

The information contained in this document is subject to change without notice.

THE AUTHOR MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. The Author shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

This document and the product described in this document have not been produced by Hewlett-Packard. Information on the origin of the product described in this document is provided in the Web site indicated below.

Web site: http://www.jeffcalc.hp41.eu/emu71/mathrom.html

ACKNOWLEDGEMENTS

Special thanks to Valentin Albillo for proofreading this document and suggesting to create shortcuts for the `INTEGRAL` and `FNROOT` keywords.

Edition History

Preliminary Edition . . . . . March 2020
Edition 1 for the Math Pac 2B . . . . . July 2020

Introducing the Math Pac 2

The Math Pac 2 is an enhanced version of the 1984 Math Pac for the HP-71 Computer. The original Math Pac provided a set of powerful tools for solving a wide range of mathematical, scientific and engineering problems.

The Math Pac 2 adds several complements by integrating the best features from other computers/calculators such as the HP-75 Computer, the Series 80 Desktop Computers, the Series 200/300 HP BASIC, the HP-28C/S Advanced Scientific Calculators and the HP-42S RPN Scientific Calculator, to reach state-of-the-art levels in terms of mathematical tools in a single BASIC environment. You will never have to use an RPN/RPL calculator or a bulky HP desktop computer for special computing purposes [1].

The Math Pac 2 adds the following capabilities to your HP-71:

- Full set of the original Math Pac 1A capabilities.
- Additional real- and complex-valued functions.
- Additional array functions.
- Additional matrix operations.
- Improvements in several functions.
- New keywords for functional compatibility with other computers and calculators.

1. This should not be taken too seriously, RPN/RPL calculators are very good and the HP BASIC (also known as Rocky Mountain BASIC) is still a powerful language for technical and instrumentation applications.

.

Contents

How to Use This Manual

This supplement assumes that you are familiar with the operation of the 1984 Math Pac 1A, which is not described again in this manual. Only the new capabilities and keywords provided in the Math Pac 2 are described here. Please refer to the 1984 Math Pac Owner's Manual for a reference to the original capabilities. However, the keyword index at the end of this manual summarizes all the capabilities of the Math Pac 2 including the original Math Pac capabilities.

Additional Information

Additional information can be found in the manuals for the HP-71, HP-75, Series 80, Series 200/300 BASIC and HP calculators, specifically:

- The HP-71 Math Pac Owner's Manual.
- The HP-75 Math Pac Owner's Manual.
- The Series 80 Matrix ROM Manual.
- The Series 200/300 BASIC Language Reference Manual.

Section 1

Installing the Math 2 LEX File

The Math 2 LEX file is installed from the Math 2 Distribution Pac.

CAUTIONS

- Don't install the Math 2 LEX together with the HP-71 Math ROM. This will definitively cause problems.
- When installing the Math 2 LEX for the first time, and if a Math ROM was previously installed, be sure to turn on the HP-71 AFTER removing the Math ROM, but BEFORE installing the Math Pac 2 (that is, with none of the Math ROM or Math 2 LEX present). This will guarantee that the new Math 2 LEX will be correctly initialized. This step is not needed when later installing new versions of the Math 2 LEX.
- The Math Pac 2 is backward compatible with the Math ROM (old programs will run correctly on the Math Pac 2), but programs using the new Math Pac 2 capabilities should not be run with the previous Math ROM installed. See Appendix A for details.

Please refer to the Math 2 Distribution Pac documentation for instructions on how to install the Math 2 LEX file in your HP-71 or in an HP-71 emulator.

Section 2

New Real- and Complex-Valued Functions

Real-Valued Trigonometric Functions

The three functions described below are for HP-75 and Series 80 compatibility.

SEC **Secant**

`SEC(X)`

Where `X` is a real-valued numeric expression.
The secant is defined by `SEC(X)=1/COS(X)` in the current angular setting.

Can be used in CALC mode.

CSC **Cosecant**

`CSC(X)`

Where `X` is a real-valued numeric expression.
The cosecant is defined by `CSC(X)=1/SIN(X)` in the current angular setting.

Can be used in CALC mode.

COT **Cotangent**

`COT(X)`

Where `X` is a real-valued numeric expression.
The cotangent is defined by `COT(X)=1/TAN(X)` in the current angular setting.

Can be used in CALC mode.

Complex-Valued Inverse Hyperbolic Functions

The Math Pac 2 adds support for complex arguments in the inverse hyperbolic functions.

ASINH **Inverse Hyperbolic Sine**

`ASINH(Z)`

Where `Z` is a complex-valued numeric expression.
Returns the complex principal value of the inverse hyperbolic sine of `Z`.
The complex inverse hyperbolic sine is defined by:
`ASINH(Z)=LOG(Z+SQRT(1+Z^2))` [1]

Can be used in CALC mode.

ACOSH **Inverse Hyperbolic Cosine**

`ACOSH(Z)`

Where `Z` is a complex-valued numeric expression.
Returns the complex principal value of the inverse hyperbolic cosine of `Z`.
The complex inverse hyperbolic cosine is defined by:
`ACOSH(Z)= SGN(IMPT(Z))*LOG(Z+(0,1)*SQRT(1-Z^2))` [1]

Can be used in CALC mode.

ATANH **Inverse Hyperbolic Tangent**

`ATANH(Z)`

Where `Z` is a complex-valued numeric expression, `Z≠(±1,0)`
Returns the complex principal value of the inverse hyperbolic tangent of `Z`.
The complex inverse tangent is defined by:
`ATANH(Z)=LOG((1+Z)/(1-Z))/2` [1]

Can be used in CALC mode.

1. The Math Pac 2 internally uses special algorithms to provide accurate results over the whole argument range.

Complex-Valued Inverse Trigonometric Functions

The Math Pac 2 adds support for complex arguments in the inverse trigonometric functions. `ASN/ACS/ATN` are for compatibility with the Series 200/300 HP BASIC.

ASIN/ASN **Inverse Sine**

`ASIN(Z)` or `ASN(Z)`

Where `Z` is a complex-valued numeric expression.
Returns the complex principal value of the inverse sine of `Z`.
The complex inverse sine is defined by:
`ASIN(Z)=(0,-1)*ASINH((0,1)*Z)` [1]

Can be used in CALC mode.

ACOS/ACS **Inverse Cosine**

`ACOS(Z)` or `ACS(Z)`

Where `Z` is a complex-valued numeric expression.
Returns the complex principal value of the inverse cosine of `Z`.
The complex inverse cosine is defined by:
`ACOS(Z)= SGN(IMPT(Z))*(0,-1)*ACOSH(Z)` [1]

Can be used in CALC mode.

ATAN/ATN **Inverse Tangent**

`ATAN(Z)` or `ATN(Z)`

Where `Z` is a complex-valued numeric expression, `Z≠(0,±1)`
Returns the complex principal value of the inverse tangent of `Z`.
The complex inverse tangent is defined by:
`ATAN(Z)=(0,-1)*ATANH((0,1)*Z)` [1,2]

Can be used in CALC mode.

---

1. The Math Pac 2 internally uses special algorithms to provide accurate results over the whole argument range.
2. A warning or error message "ATANH=Inf" is generated for `Z=(0,1)` and `Z=(0,-1)`.

Complex-Valued Base-10 Logarithm Function

The Math Pac 2 adds support for complex arguments in the base-10 logarithm function for compatibility with Series 200/300 HP BASIC `LGT` function and the RPN/RPL calculators `LOG` function.

LGT/LOG10 **Base-10 Logarithm**

`LGT(Z)` or `LOG10(Z)`

Where `Z` is a complex-valued numeric expression, `Z≠(0,0)`
The complex base-10 logarithm is defined by `LGT(Z)=LOG(Z)/LOG(10)`

Can be used in CALC mode.

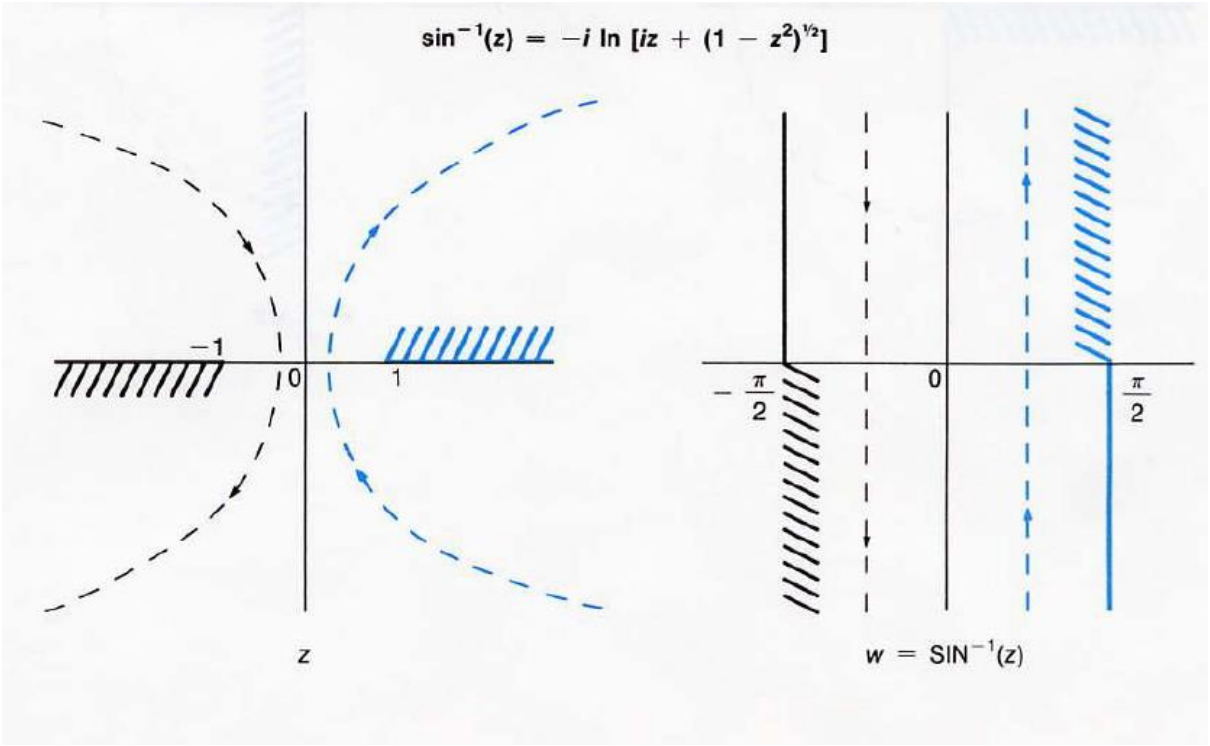
Additional Information [1]

For some inverse functions, the definitions of the principal branches are not universally agreed upon. In this discussion, uppercase letters will denote a single-valued inverse function (like $COS^{-1}\ z$) to distinguish from its multivalued inverse ($\cos^{-1} z$).

The branches used by the Math Pac were chosen such that they are analytic in the regions where their real-valued counterparts are defined; that is, the branch cut occurs where the real-valued inverse is undefined. In addition, most of the important symmetries are preserved. For example:

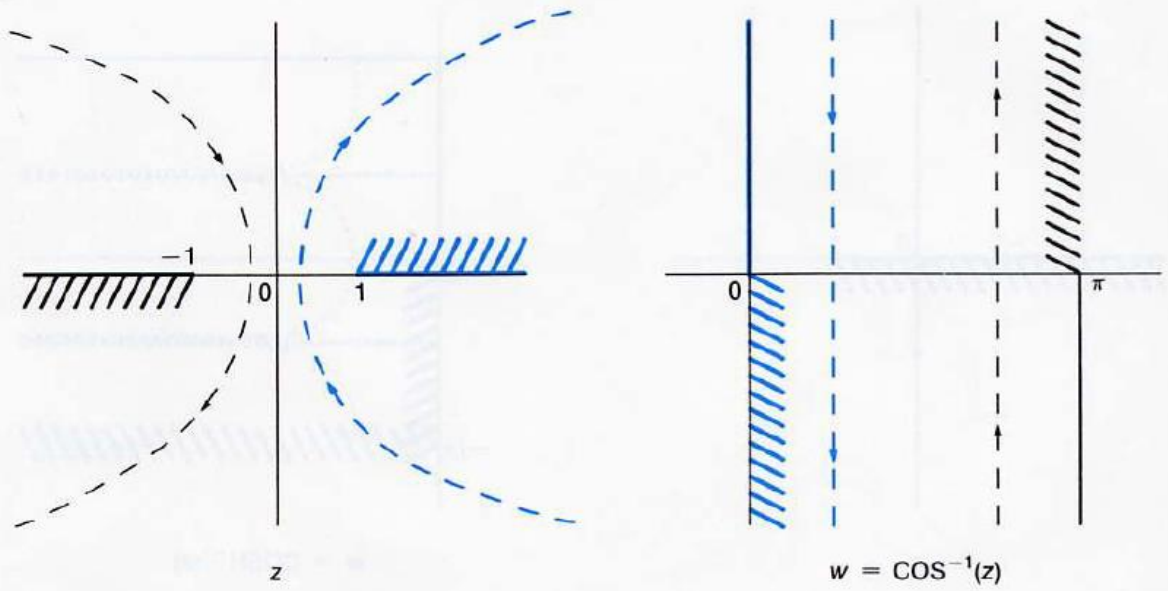
$$\mathrm{SIN}^{-1}(-z) = -\mathrm{SIN}^{-1}(z) \quad \text{for all } z.$$

The illustrations that follow show the principal branches of the new inverse functions that the Math Pac 2 calculates.

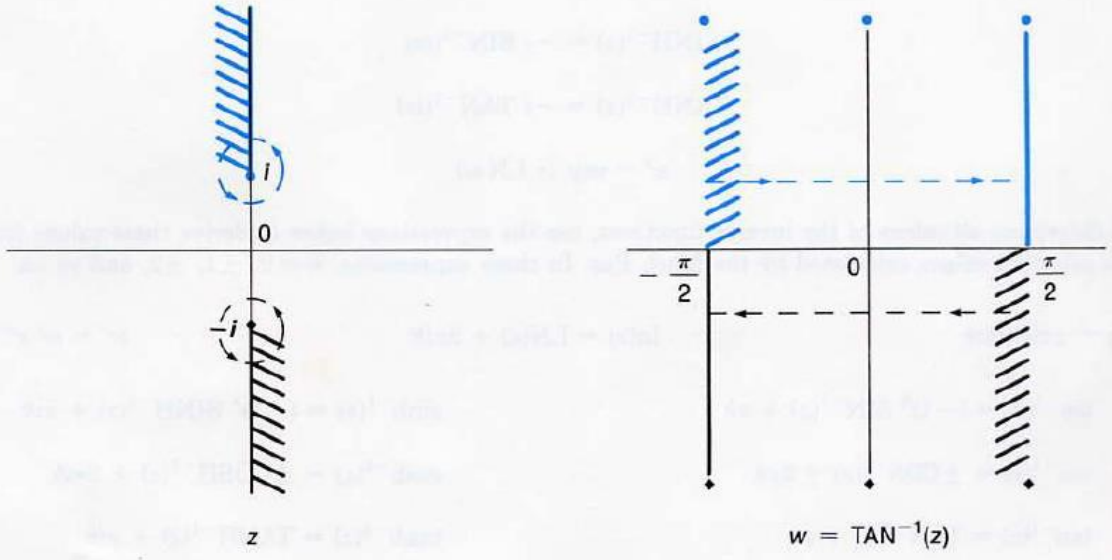


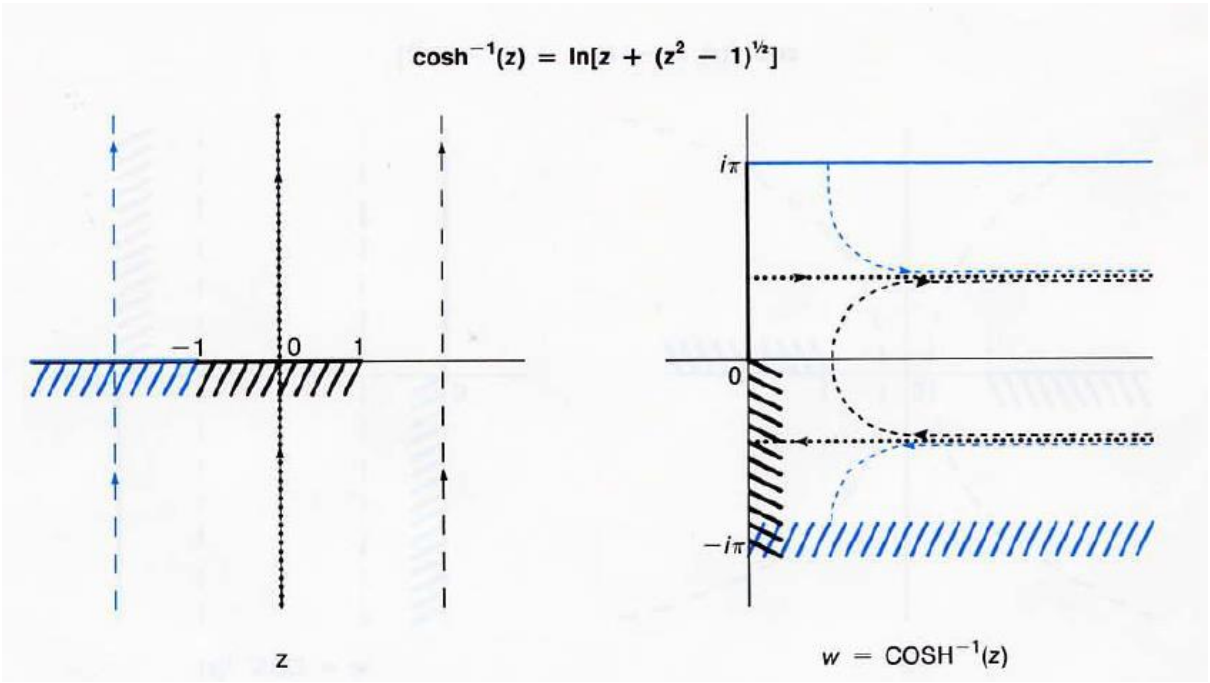
1. The information in this section is adapted from the HP-75 Math Pac Owner's Manual.

$$\cos^{-1}(z) = -i \ln [z + (z^2 - 1)^{1/2}]$$



$$\tan^{-1}(z) = \frac{i}{2} \ln \frac{i + z}{i - z}$$





The principal branches in these four graphs above are obtained from the equations shown, but don't necessarily use the principal branches of ln(z) and sqr(z).

The remaining inverse functions may be determined from the illustrations above and the following equations:

$$\mathrm{SINH}^{-1}(z) = -i\ \mathrm{SIN}^{-1}(iz)$$
$$\mathrm{TANH}^{-1}(z) = -i\ \mathrm{TAN}^{-1}(iz)$$

To determine all values of the inverse functions, use the expressions below to derive these values from the principal values calculated by the Math Pac 2. In these expressions, *k*=0, ±1, ±2, and so on.

$$\sin^{-1}(z) = (-1)^k\ \mathrm{SIN}^{-1}(z) + \pi k$$
$$\cos^{-1}(z) = \pm\ \mathrm{COS}^{-1}(z) + 2\pi k$$
$$\tan^{-1}(z) = \mathrm{TAN}^{-1}(z) + \pi k$$

$$\sinh^{-1}(z) = (-1)^k\ \mathrm{SINH}^{-1}(z) + \pi ik$$
$$\cosh^{-1}(z) = \pm\ \mathrm{COSH}^{-1}(z) + 2\pi ik$$
$$\tanh^{-1}(z) = \mathrm{TANH}^{-1}(z) + \pi ik$$

Section 3

New Scalar-Valued Array Functions

Array Element Sum Functions

The `SUM` function provides compatibility with the HP-75, Series 80 and 200/300 HP BASIC. The `ABSUM` function is for HP-75 and Series 80 compatibility.

SUM **Array Element Sum**

`SUM(A)`

Where `A` is a real- or complex-type array.
Returns the sum of the values of all elements in the array.

To halt operation, press [ATTN] twice.
Not usable in CALC mode.

ABSUM **Array Element Absolute Value Sum**

`ABSUM(A)`

Where `A` is a real-type array.
Returns the sum of the absolute values of all elements in the array.

To halt operation, press [ATTN] twice.
Not usable in CALC mode.

Array Minimum/Maximum Element Functions

The four functions described below are for HP-75 and Series 80 compatibility.

AMIN **Array Element Minimum**

`AMIN(A)`

Where `A` is a real-type array.
Returns the value of the minimum element in the array.
Store the numbers of the row and column where the element is found for `AROW` and `ACOL`.

To halt operation, press [ATTN] twice.
Not usable in CALC mode.

AMAX **Array Element Maximum**

`AMAX(A)`

Where `A` is a real-type array.
Returns the value of the maximum element in the array.
Store the numbers of the row and column where the element is found for `AROW` and `ACOL`.

To halt operation, press [ATTN] twice.
Not usable in CALC mode.

MINAB **Array Element Minimum Absolute Value**

`MINAB(A)`

Where `A` is a real-type array.
Returns the value of the smallest element (in absolute value) in the array.
Store the numbers of the row and column where the element is found for `AROW` and `ACOL`.

To halt operation, press [ATTN] twice.
Not usable in CALC mode.

MAXAB **Array Element Maximum Absolute Value**

`MAXAB(A)`

Where `A` is a real-type array.
Returns the value of the largest element (in absolute value) in the array.
Store the numbers of the row and column where the element is found for `AROW` and `ACOL`.

To halt operation, press [ATTN] twice.
Not usable in CALC mode.

Array Row/Column Functions

The two functions described below are for Series 80 compatibility.

AROW **Array Row**

AROW

Returns the value of the row set by the last execution of one of the functions RNORM, AMIN, AMAX, MINAB or MAXAB.

Usable in CALC mode.

ACOL **Array Column**

ACOL

Returns the value of the column set by the last execution of one of the functions CNORM, AMIN, AMAX, MINAB or MAXAB.

Usable in CALC mode.

Additional Information

The AROW function combines the Series 80 RNORMROW/AMINROW/AMAXROW/MAXABROW functions in a single keyword. The ACOL function combines the CNORMCOL/AMINCOL/AMAXCOL/MAXABCOL functions in a single keyword.

If more than one element is found in the AMIN/AMAX/MINAB/MAXAB functions, the element in the lowest-numbered row is chosen, and this number is returned as the value of the function AROW. If both such elements are in the same row, the element in the lowest-numbered column is chosen, and this number is returned as the value of the function ACOL.

The error "No Data" is generated if the AROW or ACOL function is executed before resp. the RNORM or CNORM function, and before any of the AMIN, AMAX, MINAB or MAXAB functions, since the last installation of the Math Pac 2.

Example [1]

```
5 ! Example adapted from the HP-85 Matrix ROM Manual
10 OPTION BASE 1
20 DIM A(3,3),B(3,3),V1(3),V2(3)
30 DATA -3,2,3,5,-3,5,2,5,-1
40 DATA 2,1,3,1,4,2
70 READ A(,)
80 READ V1(),V2()
90 MAT PRINT A;V1,V2
100 PRINT ABSUM(A)
110 PRINT AMAX(A)
120 PRINT ACOL
130 PRINT AROW
140 PRINT AMIN(A)
150 PRINT ACOL
160 PRINT AROW
170 PRINT CNORM(A)
180 PRINT ACOL
190 PRINT DET(A)
200 MAT B=INV(A)
210 PRINT DETL
220 PRINT DOT(V1,V2)
230 PRINT USING "2D.2D";FNORM(A)
240 PRINT LBND(A,1)
250 PRINT MAXAB(A)
260 PRINT ACOL
270 PRINT AROW
280 PRINT RNORM(A)
290 PRINT AROW
300 PRINT SUM(A)
310 N=2
320 PRINT UBND(A,N)
330 END
```

```
>RUN

-3   2   3   Array A
 5  -3   5
 2   5  -1

 2           Vector V1
 1
 3
```

1. Except from the lines with the `AROW/ACOL` keywords, this program can also be run on the HP-75 with the Math Pac.

1 Vector V2
4
2

29 ABSUM(A): the sum of the absolute values of the elements in A.

5 AMAX(A): the value of the largest element in A.

1 ACOL: the lowest-numbered column containing AMAX(A).

2 AROW: the lowest-numbered row containing AMAX(A).

-3 AMIN(A): the value of the smallest element in A.

1 ACOL: the lowest-numbered column containing AMIN(A).

1 AROW: the lowest-numbered row containing AMIN(A).

10 CNORM(A): the largest sum of the absolute values of the elements in each column of A.

1 ACOL: the lowest-numbered column with the largest sum of absolute values.

189 DET(A): Determinant of A.

189 DETL: Determinant of matrix A inverted in preceding MAT..INV statement.

12 DOT(V1,V2): the sum of the products of corresponding elements of V1 and V2.

10.54 FNORM(A): the square root of the sum of the squares of the elements in A.

1 LBND(A,1): the lower bound of the first subscript of A.

5 MAXAB(A): the largest absolute value of any element in A.

1 ACOL: the lowest-numbered column containing the element with largest absolute value.

2 AROW: the lowest-numbered row containing the element with largest absolute value.

13 RNORM(A): the largest sum of the absolute values of the elements in each row of A.

2 AROW: the lowest-numbered row with the largest sum of absolute values.

15 SUM(A): the sum of the elements in A.

3 UBND(A,2): the upper bound of the second subscript of A.

Complex Matrix Determinant

DET/CDET **Complex Determinant**

`DET(A)` or `CDET(A)`

Where `A` is a square real- or complex-type matrix.
Returns the determinant of the matrix `A`.
If `A` is real, then the result is real else the result is complex.

To halt operation, press [ATTN] twice.
Not usable in CALC mode.

Additional Information

The `DET` function is now extended to handle complex matrices.

`CDET` is equivalent to `DET` but doesn't accept the syntax `CDET` without operand.

Note that the `DET` without operand and the `DETL` functions don't provide the determinant of the last complex matrix used in the `MAT..SYS`, `MAT..INV` or `MAT..INV*` statements. The reason is that the determinant is not computed in the process of inverting a complex matrix. The `DET` or `CDET` function has to be used to get the determinant of a complex matrix.

Alternate Inner Product

DOT2 is an alternate form of the inner (dot) product function, for compatibility with RPN/RPL calculators and Series 200/300 BASIC.

DOT2 **Alternate Inner (Dot) Product**

DOT2(A,B)

Where A and B are real- or complex-type arrays with the same number of elements. Returns A.B the inner product of A and B. If both A and B are real, then the result is real. If either A or B is complex, then the result is complex.

To halt operation, press [ATTN] twice.
Not usable in CALC mode.

Additional Information

For real vectors, the DOT2 function is equivalent to DOT.

For complex vectors, DOT2 differs from DOT in that the DOT2 inner product uses the ordinary complex product of corresponding vector elements, instead of using the complex conjugate of the elements of the first vector. This ensures compatibility with the HP-28S and HP-42S calculators and the Series 200/300 BASIC.

DOT2 also accepts matrices as arguments. The matrices A and B must have the same numbers of rows and columns. The inner product is computed on all corresponding elements of the matrices. This feature comes from the HP-28S and HP-42S calculators.

Section 4

New Matrix Operations

Cross Product Operation

The MAT..CROSS operation provides compatibility with the HP-75 and Series 80.

CROSS **Cross Product**

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

Where A, B and C are vectors (1-dimension arrays).
Both B and C must be real-type vectors and have 2 or 3 elements.

Implicitly redimensions A to have exactly three elements and assigns to A the values of the cross product B x C.

Not usable in CALC mode.

Additional Information

The operands of the CROSS statement can each have two or three elements. If missing, the third element is replaced by zero. If both operands have two elements only, then the elements 1 and 2 of the result are zero. The MAT A=CROSS(B,C) is then a quick way to compute the quantity B(1)*C(2)-B(2)*C(1) .

This feature comes from the HP-28S and HP-42S calculators.

Example

```
> OPTION BASE 1
> DIM A(3),B(2),C(2)
> MAT INPUT B,C
B(1)? 2,7
C(1)? 5,8
> MAT A=CROSS(B,C)
> DISP A(3)
-19          This is 2*8-7*5
```

Row and Column Sum Operations

The MAT..RSUM and MAT..CSUM operations provide compatibility with the HP-75, the Series 80 and the Series 200/300 BASIC.

RSUM **Row Sum**

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

Where A is a vector and B is a matrix.
Matrix B may be either real or complex type.

Implicitly redimensions A to have as many elements as there are rows in B and assigns the sum of the values in each row of B to the corresponding element of A.

To halt operation, press [ATTN] twice.
Not usable in CALC mode.

CSUM **Column Sum**

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

Where A is a vector and B is a matrix.
Matrix B may be either real or complex type.

Implicitly redimensions A to have as many elements as there are columns in B and assigns the sum of the values in each column of B to the corresponding element of A.

To halt operation, press [ATTN] twice.
Not usable in CALC mode.

Additional Information

If the operand B is complex type, then A must be complex.
If B is real, then A may be real or complex; if complex, all imaginary parts of all elements in A are set to zero.

MAT..CSUM can't be used to get the sum of the elements of a vector, use the array function SUM instead.

Matrix Division Operation

The MAT..INV()* operation provides compatibility with the Series 80.

INV * **Matrix Division**

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

Where A is a square matrix, X and B are both vectors or both matrices, and A and B are conformable for multiplication.
Arrays A and B may be either real or complex type. If either A or B is complex, then X must be complex.
If both A and B are real, then X may be real or complex; if complex, all imaginary parts of all elements in x are set to zero.

Implicitly redimensions X to be the same size as B and assigns to X the computed solution to the matrix equation A X=B.

To halt operation, press [ATTN] twice.
Not usable in CALC mode.

Additional Information

MAT X=INV(A)*B directly computes the solution of the matrix equation A X=B instead of inverting A then multiplying by B. It is however not exactly equivalent to MAT X=SYS(A,B) in that it doesn't compute the residual correction that MAT..SYS does.

MAT X=INV(A)*B uses less memory than MAT X=SYS(A,B) and is faster and generally more accurate than doing MAT A=INV(A) then MAT X=A*B.

The MAT X=INV(A)*B operation is the exact equivalent of the matrix division operation on the HP-28S and HP-42S calculators, hence the name "matrix division" for MAT..INV()*.

The table below summarizes the three methods available to solve the matrix equation A X=B, together with their benefits. Each method can produce a slightly different result.

System solving comparison

Operations	Features
MAT X=SYS(A,B)	Best accuracy, more memory used.
MAT X=INV(A)*B	Less memory used, no residual correction.
MAT A=INV(A) @ MAT X=A*B	Matrix inversion done only once for solving several systems.

Example

Solving the system $\begin{bmatrix} 69 & 58 & 95 \\ 51 & 43 & 71 \\ 32 & 55 & 54 \end{bmatrix} \begin{bmatrix} \\ X \\ \\ \end{bmatrix} = \begin{bmatrix} 692 \\ 515 \\ 445 \end{bmatrix}$.

```
> OPTION BASE 1
> DIM A(3,3),B(3),X(3)
> MAT INPUT A,B
A(1,1)? 69,58,95
A(2,1)? 51,43,71
A(3,1)? 32,55,54
B(1)? 692,515,445
```

Using `MAT..SYS` :

```
> MAT X=SYS(A,B)
> MAT DISP X
 2
 3
 4
```

Using `MAT..INV()*` :

```
> MAT X=INV(A)*B
> MAT DISP X
 1.99999999732
 2.99999999918
 4.00000000242
```

Using `MAT..INV` and `MAT..*` :

```
> MAT A=INV(A)
> MAT X=A*B
> MAT DISP X
 2.00000000360
 3.00000000053
 3.99999999890
```

LU Decomposition

The `MAT..LUFACT` operations provides compatibility with the HP-75.

LUFACT ***LU* Decomposition**

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

Where `B` is a square real matrix, and `A` is a matrix.

Implicitly 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` above the diagonal are assigned the value of the corresponding elements in `U` and the elements in `A` on or below the diagonal are assigned the value of the corresponding elements in `L`.

To halt operation, press [ATTN] twice.
Not usable in CALC mode.

Example

```
> OPTION BASE 1
> DIM A(3,3)
> MAT INPUT A
A(1,1)? 1,1,1
A(2,1)? 1,0,2
A(3,1)? 2,1,2
> MAT A=LUFACT(A)
> MAT DISP A;
 2   .5   1
 1  -.5  -2
 1   .5   1
```

The `L` and `U` parts of the *LU* decomposition of `A` are respectively:

$$\begin{bmatrix} 2 & & \\ 1 & -.5 & \\ 1 & .5 & 1 \end{bmatrix} \text{ and } \begin{bmatrix} & .5 & 1 \\ & & -2 \\ & & \end{bmatrix},$$

so $L = \begin{bmatrix} 2 & 0 & 0 \\ 1 & -.5 & 0 \\ 1 & .5 & 1 \end{bmatrix}$ and $U = \begin{bmatrix} 1 & .5 & 1 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}$

Additional Information [1]

The Math Pac *LU* decomposition factors a square matrix A into the matrix product LU, where

- L is a lower-triangular matrix - it has values of zero for all elements above the diagonal.
- U is a upper-triangular matrix - it has values of zero for all elements below the diagonal - with values of one for all elements on the diagonal.

For example, $A = \begin{bmatrix} 2 & 1 \\ 3 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 3 & -.5 \end{bmatrix} \begin{bmatrix} 1 & .5 \\ 0 & 1 \end{bmatrix} = LU$.

Some matrices can't be factored into LU form. For example, $A = \begin{bmatrix} 0 & 1 \\ 1 & 2 \end{bmatrix} \neq LU$ for *any* pair of lower- and upper-triangular matrices L and U. However, if rows are interchanged in the matrix to be decomposed, then *any non-singular* matrix can be so decomposed. Row interchanges in the matrix A can be represented by the matrix product PA for some permutation matrix P. Allowing for row interchanges, the *LU* decomposition can be represented by the equation PA = LU. So, for the above example

$$PA = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix} = LU.$$

Row interchanges can also reduce rounding errors that can occur during the calculation of the decomposition.

The *LU* decomposition is returned in the form $\begin{bmatrix} & U \\ L & \end{bmatrix}$.

For example, if the result of the `MAT A=LUFACT(B)` statement is

$A = \begin{bmatrix} 2 & 3 & 4 \\ 5 & 6 & 7 \\ 8 & 9 & 2 \end{bmatrix}$, then $L = \begin{bmatrix} 2 & 0 & 0 \\ 5 & 6 & 0 \\ 8 & 9 & 2 \end{bmatrix}$ and $U = \begin{bmatrix} 1 & 3 & 4 \\ 0 & 1 & 7 \\ 0 & 0 & 1 \end{bmatrix}$

and PB = LU for some row interchange matrix P.

It is not necessary to store the diagonal elements of U in the result matrix since the value of each of these is equal to one. The row interchanges are also recoverable in many cases because, aside from row interchanges, the first column of L is the same as the first column of the matrix being decomposed.

For example, if $B = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}$ and `MAT A=LUFACT(B)` is executed, then $A = \begin{bmatrix} 2 & .5 \\ 1 & -.5 \end{bmatrix}$.

The fact that the first column of A is reversed from the first column of B indicates that the rows have been interchanged, so that

$$PB = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 1 & -.5 \end{bmatrix} \begin{bmatrix} 1 & .5 \\ 0 & 1 \end{bmatrix} = LU.$$

1. The information in this section is adapted from the HP-75 Math Pac Owner's Manual.

In many cases, the *LU* decomposition will be correct *even if the matrix is singular*. This can be checked by remultiplying the `L` and `U` matrices and comparing the result to the original matrix. This feature gives you the ability to find the *LU* decomposition of matrices that are not square.

For example, to find the *LU* decomposition of the 3 x 4 matrix $\begin{bmatrix}1 & 2 & 3 & 4\\ 1 & 4 & 9 & 16\\ 1 & 8 & 27 & 64\end{bmatrix}$,

we will find the decomposition of $\begin{bmatrix}1 & 2 & 3 & 4\\ 1 & 4 & 9 & 16\\ 1 & 8 & 27 & 64\\ 0 & 0 & 0 & 0\end{bmatrix}$ instead.

From this decomposition, the *LU* decomposition of the original matrix is easily found.
The keystroke sequence below illustrates the process (assuming `OPTION BASE 1`).

```
> DIM S(4,4)
> MAT INPUT S
S(1,1)? 1,2,3,4
S(2,1)? 1,4,9,16
S(3,1)? 1,8,27,64
S(4,1)? 0,0,0,0
> MAT S=LUFACT(S)
> MAT DISP S;
 1   2   3   4
 1   6   4   10
 1   2  -2   4
 0   0   0   0
```

Therefore `L` = $\begin{bmatrix}1 & 0 & 0 & 0\\ 1 & 6 & 0 & 0\\ 1 & 2 & -2 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}$ and `U` = $\begin{bmatrix}1 & 2 & 3 & 4\\ 0 & 1 & 4 & 10\\ 0 & 0 & 1 & 4\\ 0 & 0 & 0 & 1\end{bmatrix}$.

Their matrix product is $\begin{bmatrix}1 & 2 & 3 & 4\\ 1 & 8 & 27 & 64\\ 1 & 4 & 9 & 16\\ 0 & 0 & 0 & 0\end{bmatrix}$,

so that $\begin{bmatrix}1 & 0 & 0\\ 0 & 0 & 1\\ 0 & 1 & 0\end{bmatrix}\begin{bmatrix}1 & 2 & 3 & 4\\ 1 & 4 & 9 & 16\\ 1 & 8 & 27 & 64\end{bmatrix} = \begin{bmatrix}1 & 0 & 0\\ 1 & 6 & 0\\ 1 & 2 & -2\end{bmatrix}\begin{bmatrix}1 & 2 & 3 & 4\\ 0 & 1 & 4 & 10\\ 0 & 0 & 1 & 4\end{bmatrix}$

This technique works best when the matrix has fewer rows than columns. If your matrix has fewer columns than rows, find the *LU* decomposition of its *transpose* by the above technique, and take the transpose of the result.

Section 5

Other Math Pac 2 Improvements

Complex Square Operation

With the Math Pac 1A, a square operation on a complex number such as Z^2 is internally computed as exp(2*ln(Z)). With complex numbers, this can limit the accuracy in some cases, such as:

```
> (1E-9,1)^2
(-1,2.00000769645E-9)
```

With the Math Pac 2, the complex square operation Z^2 is now internally computed as Z*Z to provide the same performance re speed and accuracy for complex numbers as the [x^2] key on RPN/RPL calculators (15C, 28C/S, 42S and others).

```
> (1E-9,1)^2
(-1,2E-9)
```

It is still possible to use the general exponentiation formula by forcing a complex power:

```
> (1E-9,1)^(2,0)
(-1,2.00000769645E-9)
```

Moreover, the special case is not based on a pattern detected during the expression parsing but at execution, so an expression like:

`X=2 @ Z=(1E-9,1)^X` is handled as a multiplication rather than an exponentiation.

Note that the new square functionality operates only on complex numbers, a real square operation like X^2 is still computed by the HP-71 as exp(2*ln(X)). This generally doesn't affect the accuracy of the result.

A side improvement of the new complex square functionality is that the signed 0 (a specific HP-71 feature) is now conserved by the complex square operation:

```
> (-0,1)^(2,0)
(-1,0)
```

the sign of 0 is not preserved with the Math 1A square function [1]

```
> (-0,1)^2
(-1,-0)
```

the sign of 0 is now preserved

1. This limitation was mentioned by Prof. W. Kahan in his article: "Branch Cuts for Complex Elementary Functions", 1986

RNORM Row and CNORM Column

The RNORM and CNORM functions now store the number of the row (for RNORM) or column (for CNORM) that has the largest sum of absolute values (the norm), these numbers are then available with the new AROW and ACOL functions, until the execution of another function that changes the stored row/column numbers (see section 3).

This functionality is similar to the RNORMROW and CNORMCOL functions of the Series 80.

If more than one row/column (resp. for RNORM/CNORM) has the largest sum of absolute values, the lowest-numbered row/column is chosen and this number is returned as the value of the function AROW/ACOL.

Revised Matrix Algorithms

The original HP-71 Math Pac was sometimes giving results that were slightly different from RPN/RPL calculators for some matrix operations. Parts of the matrix algorithms in the Math Pac 2 have been revised and aligned with the HP-28S and HP-42S calculators.

The main change is related to the internal summation operation that is used in several matrix algorithms.

For instance, computing the determinant of the matrix $\begin{bmatrix} 69 & 58 & 96 \\ 51 & 43 & 71 \\ 32 & 55 & 54 \end{bmatrix}$:

```
> OPTION BASE 1
> DIM A(3,3)
> MAT INPUT A
A(1,1)? 69,58,96
A(2,1)? 51,43,71
A(3,1)? 32,55,54
> DET(A)
```

produced the result 1.00000038313 with the original HP-71 Math Pac 1A,
but is evaluated as 1.00000038333 on the HP-28S and HP-42S.

The Math Pac 2 matrix algorithms are now aligned with the HP-28S and HP-42S and return the same results as these machines.

Note that this alignment generally doesn't produce more accurate results (the exact determinant of the matrix above is 1), but it makes the HP-71 Math Pac 2 more consistent with the RPN/RPL calculators.

For 2x2 matrices, the DET function now uses a direct and generally more accurate formula to compute the determinant in accordance with the HP-28S and HP-42S calculators.

Shortcuts for INTEGRAL and FNROOT Operations

The Math Pac 2 now offers shortcuts to make the use of the `INTEGRAL` and `FNROOT` operations easier and faster especially in direct keyboard calculations:

Keywords	Shortcuts
INTEGRAL	INTEG
FNROOT	FROOT
IVAR	IX
FVAR	FX

The shortcuts can also be used for program line entry and are automatically replaced by the full keyword names.

Examples

Find the value of the integral $\int_0^1 x^3 - x^2 + x + 2 \, . dx$ with an accuracy of 10^{-5} :

```
> INTEG(0,1,1E-5,IX^3-IX^2+IX+2)
 2.41666664282
```

Find the real root (located between 0 and 1) of the equation $7x^3 + 6x^2 + 6x - 1 = 0$:

```
> FROOT(0,1,7*FX^3+6*FX^2+6*FX-1)
 .142857142858
```

Program line entry:

Key in the line:

```
 10 X=INTEG(0,1,E,IX^3-IX^2+IX+2) [END LINE]
```

The line is then displayed as:

```
 10 X=INTEGRAL(0,1,E,IVAR^3-IVAR^2+IVAR+2)
```

Section 6

Compatibility with Other HP Computers and Calculators

This section summarizes the compatibility of the HP-71 with other HP BASIC computers and with RPN/RPL scientific calculators.

Compatibility with the HP-75 and the Series 80

The Series 70 (HP-71, HP-75) and Series 80 (HP-85, HP-86) share many aspects of their Technical BASIC variants. Thus the compatibility is already very good and is further enhanced by the HP-71 Math Pac 2.

The tables below focus on the array and matrix functions. It is assumed the HP-75 is using the Math Pac ROM and the Series 80 models (noted S80 below) are using the Matrix ROM. The features marked as **X** in bold are provided by the Math Pac 2.

Scalar real-valued matrix functions

Keyword	71	75	S80	Comments
DET(A)	X	X	X	
DETL	X	X	X	
FNORM(A)	X	X	X	
CNORM(A)	X	X	X	
CNORMCOL	*		X	* use ACOL
RNORM(A)	X	X	X	
RNORMROW	*		X	* use AROW
AMIN(A)	**X**	X	X	Value of the minimum element
AMINCOL	*		X	* use ACOL
AMINROW	*		X	* use AROW
AMAX(A)	**X**	X	X	Value of the maximum element
AMAXCOL	*		X	* use ACOL
AMAXROW	*		X	* use AROW
MINAB(A)	**X**	X		Value of the smallest el. in abs value
MAXAB(A)	**X**	X	X	Value of the largest el. in abs value
MAXABCOL	*		X	* use ACOL
MAXABROW	*		X	* use AROW
SUM(A)	**X**	X	X	Sum of the values of all elements
ABSUM(A)	**X**	X	X	Sum of the absolute values of all el.
DOT(X,Y)	X	X	X	
LBND(A,N)	X	X	X	
UBND(A,N)	X	X	X	

Matrix operations:

Keyword	71	75	S80	Comments
MAT A=-B	X	X	X	
MAT A=B+C	X	X	X	
MAT A=B-C	X	X	X	
MAT A=B*C	X	X	X	matrix multiply
MAT A=B.C			X	element-per-element multiplication
MAT A=B/C			X	element-per-element division
MAT A=(X)*B	X	X	X	multiplication by a scalar
MAT A=(X)+B			X	
MAT A=(X)-B			X	
MAT A=(X)/B			X	
MAT A=(X)*B+(Y)*C			X	
MAT A=CROSS(B,C)	**X**	X	X	
MAT A=CSUM(B)	**X**	X	X	
MAT A=RSUM(B)	**X**	X	X	
MAT A=INV(B)	X	X	X	
MAT A=INV(B)*C	**X**		X	
MAT A=TRN(B)	X	X	X	
MAT A=TRN(B)*C	X		X	
MAT A=B*TRN(C)			X	
MAT X=SYS(A,B)	X	X	X	
MAT A=LUFACT(B)	**X**	X		

Array input and output:

Keyword	71	75	S80	Comments
MAT A=B	X	X	X	
MAT A(..)=B(..)			X	subarray copy
MAT A=(X)	X	X	X	
MAT A=CON	X	X	X	
MAT A=IDN	X	X	X	
MAT A=ZER	X	X	X	
MAT INPUT A	X	X	X	
MAT READ A	*	X	X	* use READ A
MAT DISP [USING] A	X	X	X	
MAT PRINT [USING] A	X	X	X	
REDIM A(..)	*	X	X	* use DIM A(..)

Complex matrix operations (HP-71 and HP-75 only):

Operation	71	75	Comments
det	Z=DET(A)	MAT Z=CDET(A)	
idn	MAT A=IDN	MAT A=CIDN	
zer	MAT A=ZER	MAT A=ZER	
inv	MAT A=INV(B)	MAT A=CINV(B)	
mult	MAT A=B*C	MAT A=CMMULT(B,C)	
sys	MAT Z=SYS(A,B)	MAT Z=CSYS(A,B)	
conj.trans	MAT A=TRN(B)	MAT A=CTRN(B)	

Compatibility with the Series 200/300 BASIC

The Series 200/300 BASIC (noted RMB below) differs in several aspects from the HP-71 BASIC.

The tables below focus on the array and matrix functions and refer to RMB 5 or later.

Scalar real- or complex-valued matrix functions:

RMB Keyword	71	Comments
DET(A)	X	real and complex matrices
DET (no operand)	X	last real matrix determinant
SUM(A)	**x**	
DOT(X,Y)	X	use DOT2(X,Y) with complex vectors
BASE(A,N)	*	use LBND(A,N)
SIZE(A,N)	*	use UBND(A,N)-LBND(A,N)+1
RANK(A)	*	use 1+(UBND(A,2)>=0)

Matrix operations:

RMB operation	71	Comments
MAT A=-B	X	
MAT A=B+C	X	
MAT A=B-C	X	
MAT A=B*C	X	matrix multiply
MAT A=B.C		element-per-element multiplication
MAT A=B/C		element-per-element division
MAT A=B.relop.C		element-per-element relation operator
MAT A=(X)*B	X	multiplication by a scalar
MAT A=(X)+B		
MAT A=(X)-B		
MAT A=(X)/B		
MAT A=(X).relop.B		relation operator
MAT A=CSUM(B)	**x**	
MAT A=RSUM(B)	**x**	
MAT A=INV(B)	X	
MAT A=TRN(B)	X	
MAT X=SYS(A,B)	X	

Array input and output:

RMB Keyword	71	Comments
MAT A=B	X	
MAT A(..)=B(..)		subarray copy
MAT A=(X)	X	
MAT A=IDN	X	
READ A(*)	*	use READ A
INPUT A(*)	*	use MAT INPUT A
DISP [USING] A(*)	*	use MAT DISP [USING] A
PRINT [USING] A(*)	*	use MAT PRINT [USING] A
REDIM A(..)	*	use DIM A(..)

Compatibility with RPN/RPL calculators

The HP-71 Math Pac 2 provides several features to increase the functional compatibility with RPN/RPL calculators. However, the names of the corresponding functions may differ between the HP-71 BASIC and RPN/RPL calculator languages. The tables below list the new Math Pac 2 functions and some correspondences between similar functions.

Complex functions:

71	28S	42S	Comments
(*complex expression*)^2	SQ	X^2	complex square
ASIN/ACOS/ATAN	same		
ASINH/ACOSH/ATANH	same		
LN	same		natural logarithm
EXP	EXP	E^X	
LOG10	LOG	LOG	base-10 logarithm
10^(*complex expression*)	ALOG	10^X	
SGN	SIGN	SIGN	

Array functions and matrix operations:

71	28S	42S	Comments
FNORM	ABS	FNRM	
CNORM	CNRM	n/a	
RNORM	RNRM	RNRM	
DOT/DOT2	DOT	DOT	See note below for complex vectors
MAT..CROSS	CROSS	CROSS	71: real values only
MAT..RSUM	n/a	RSUM	
MAT..INV	INV	INVRT	
MAT..INV()*	*	*	* 28S/42S: use matrix division
MAT..SYS	*	n/a	* 28S: use matrix division and RSD

Note: the HP-71's DOT function computes the inner product on complex vectors using the complex conjugates of the first vector elements. The HP-28S and HP-42S are using the ordinary product without conjugation. Use DOT2 for full compatibility.

Appendices

A. Math Pac 1 and 2 Compatibility

The Math Pac 2 is fully backwards compatible with the original Math Pac.

Programs written for the Math Pac 1A will correctly run with the Math Pac 2 without modification. Some results may be slightly different due to the new improvements provided by the Math Pac 2, see section 5. This applies to:

- the complex square operation `Z^2`,
- some matrix operations internally using the summation operation: vector dot product, norms, matrix multiplication and inversion, determinant, system solving.

A program using the new Math Pac 2 keywords will not run with the original Math Pac. An attempt to execute a program statement that uses new Math Pac 2 functions or `MAT` operations will produce errors such as:

`XFN Not Found` or `XWORD Not Found`

When listing a program with the Math Lex 2 not installed, the new functions will be listed as `XFN20xx()` and the new `MAT` operations will be listed as `XWORD2064`.

Examples:

`100 S=SUM(A)` will be listed as `100 S=XFN2063(A)`
`200 MAT X=CSUM(A)` will be listed as `200 XWORD2064`

In any case, trying to list or execute a program that uses new Math Pac 2 statements, with Math Pac 1A installed, will neither crash nor corrupt any HP-71 operations.

B. New Memory Requirements

The Math Pac 2 reserves 59.5 bytes of read/write memory for its own uses (The Math Pac 1A was using 43.5 bytes). This section lists the amount of temporary memory used by new Math Pac 2 operations. Refer to the original Math Pac Owner's Manual, Appendix B, for details.

Matrix operations	Memory Required For Operation
`MAT B=LUFACT(A)`	Same as `MAT B=INV(A)`
`MAT X=INV(A)*B`	Same as `MAT X=SYS(A,B)` minus 8*NP* bytes if `B` is real or minus 16*NP* bytes if `B` is complex
`DET(A)/CDET(A)`	2*N*(8*N*+1) bytes if `A` is complex

C. New Error Conditions

The Math Pac 2 adds these new error conditions:

Error number	Error Message and Condition
1	`#Dims` • `MAT A=CROSS(B,C)` : `A` or `B` or `C` is a matrix • `MAT A=RSUM(B), MAT A=CSUM(B)` : `A` is a matrix or `B` is a vector • `MAT X=INV(A)*Y, MAT B=LUFACT(A)` : `A` or `B` is a vector
2	`Not Square` • `MAT X=INV(A)*B, MAT B=LUFACT(A)` : `A` is a not a square matrix
3	`Conformability` • `MAT X=INV(A)*B` : `A` and `B` are not conformable for multiplication • `DOT2(A,B)` : `A` and `B` don't have the same number of elements
8	`Bad Array Size` • `MAT A=CROSS(B,C)` : the number of elements of vectors `B` or `C` is not two or three.
11	`ATANH=Inf` • `ATANH((1,0))` or `ATANH((-1,0))` • `ATAN((0,1))` or `ATAN((0,-1))`

D. New Attention Key Actions

Refer to the original Math Pac Owner's Manual, Appendix D, for details on the [ATTN] key operation.

`MAT` Statements

The following `MAT` statements may be halted at any time by pressing [ATTN] *twice*.

`MAT` *result* = `CSUM`(*operand*)

`MAT` *result* = `RSUM`(*operand*)

`MAT` *result* = `INV`(*operand1*) `*` *operand2*

`MAT` *result* = `LUFACT`(*operand*)

Scalar-Valued Array Functions

The following scalar-valued functions can be halted at any time by pressing [ATTN] *twice*.

`SUM`(*operand*)

`ABSUM`(*operand*)

`AMIN`(*operand*)

`AMAX`(*operand*)

`MINAB`(*operand*)

`MAXAB`(*operand*)

`CDET`(*operand*)

`DOT2`(*operand*)

For these functions, the HP-71 will display the error message `Function Interrupted`.

Keyword Index

Page numbers starting with M refer to the original Math Pac Owner's Manual.
Page numbers **in bold** starting with **S** refer to this Math Pac 2 Manual Supplement.

DET (no operand)	M69	Determinant of last real matrix used as operand of INV or LUFACT, or first operand of INV()* or SYS.
DETL	M69	Same as DET (no operand).
DOT	M71	Dot (inner) product.
DOT2	**S21**	Alternate dot (inner) product.
EXP	M37	Complex exponential (e^z).
FGUESS	M90	Second-best guess to value returned by last FNROOT.
FNORM	M70	Frobenius norm.
FNROOT	M89	Rootfinding for functions.
FROOT	**S31**	Shortcut for FNROOT.
FVALUE	M90	Functional value of last FNROOT.
FVAR	M90	Variable to solve for in FNROOT.
FX	**S31**	Shortcut for FVAR.
GAMMA	M28	Gamma function.
IBOUND	M103	Uncertainty of last INTEGRAL.
IMPT	M21	Imaginary part of complex number.
INTEGRAL	M101	Integration of functions.
INTEG	**S31**	Shortcut for INTEGRAL.
IROUND	M30	Integer round.
IVALUE	M102	Current approximation to an INTEGRAL.
IVAR	M102	Variable of integration in INTEGRAL.
IX	**S31**	Shortcut for IVAR.
LBND	M72	Array subscript lower bound.
LBOUND	M71	Same as LBND.
LGT	**S11**	Complex base-10 logarithm.
LOG	M37	Complex natural logarithm.
LOG10	**S11**	Same as LGT.
LOG2	M29	Log base 2.
MAT DISP	M54	Array display (unformatted).
MAT DISP USING	M55	Array display (formatted).
MAT INPUT	M53	Interactive array input.
MAT PRINT	M55	Array printing (unformatted).
MAT PRINT USING	M56	Array printing (formatted).
MAT .. =	M51	Array copying (simple assignment).
MAT .. +	M64	Array addition.
MAT .. -	M63	Array negation.
MAT .. -	M64	Array subtraction.
MAT .. *	M65	Matrix multiplication.
MAT .. ()	M52	Scalar to array assignment (numeric expression assignment).
MAT .. ()*	M65	Scalar multiplication.
MAT .. CON	M52	Constant array with optional redimensioning.
MAT .. CROSS	**S22**	Vector cross product.
MAT .. CSUM	**S23**	Array column sum.

MAT .. FOUR	M135	Finite Fourier transform.
MAT .. IDN	M52	Identity array with optional redimensioning.
MAT .. INV	M77	Matrix inversion.
MAT .. INV()*	**S24**	Matrix division.
MAT .. LUFACT	**S26**	Matrix *LU* decomposition.
MAT .. PROOT	M120	Polynomial rootfinding.
MAT .. RSUM	**S23**	Array row sum.
MAT .. SYS	M79	System solution.
MAT .. TRN	M77	Transpose or conjugate transpose.
MAT .. TRN()*	M66	Transpose or conjugate transpose multiply.
MAT .. ZER	M53	Zero array with optional redimensioning.
MAT .. ZERO	M53	Same as MAT..ZER.
MAXAB	**S16**	Array element maximum absolute value.
MINAB	**S16**	Array element minimum absolute value.
NAN$	M30	NaN diagnostic function.
NEIGHBOR	M30	Successor/predecessor function.
POLAR	M40	Rectangular to polar function.
PROJ	M42	Conversion of complex infinities to projective infinities.
RECT	M40	Polar to rectangular function.
RNORM	M70	Infinity (row) norm of an array.
SCALE10	M29	Exponent scaling function.
SEC	**S8**	Secant function.
SGN	M41	Complex unit vector.
SIN	M38	Complex sine.
SINH	M27	Hyperbolic sine.
SINH	M39	Complex hyperbolic sine.
SQR	M40	Complex square root.
SQRT	M40	Same as SQR.
SUM	**S15**	Array element sum.
TAN	M38	Complex tangent.
TANH	M27	Hyperbolic tangent.
TANH	M39	Complex hyperbolic tangent.
TYPE	M31	Data type function.
UBND	M71	Array subscript upper bound.
UBOUND	M71	Same as UBND.
+	M35	Complex addition.
–	M35	Complex unary minus.
–	M36	Complex subtraction.
*	M36	Complex multiplication.
/	M36	Complex division.
^	M36	Complex exponentiation (Z^W).
= < > # ?	M43	Complex relational operators.