ND-500/2 Double Precision Array Processing Functions

ND-05.018.01

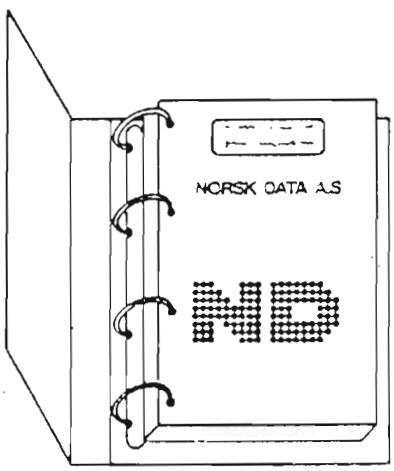
ND-500/2 Double Precision Array Processing Functions

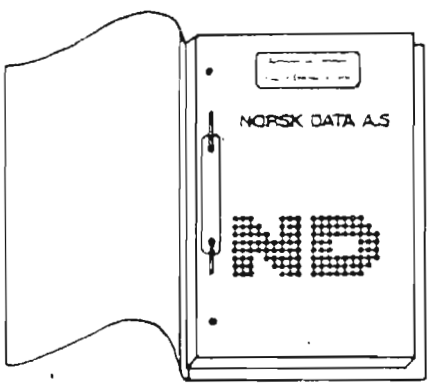
ND-05.018.01

This manual is in loose-leaf form for ease of updating. Old pages may be removed and new pages easily inserted if the manual is revised.

The loose-leaf form also allows you to place the manual in a ring binder (A) for greater protection and convenience of use. Ring binders with 4 rings corresponding to the holes in the manual may be ordered in two widths, 30 mm and 40 mm. Use the order form below.

The manual may also be placed in a plastic cover (B). This cover is more suitable for manuals of less than 100 pages than for large manuals. Plastic covers may also be ordered below.





A: Ring Binder

B: Plastic Cover

Please send your order to the local ND office or (in Norway) to:

Norsk Data A.S
Graphic Center
P.O. Box 25, Bogerud
0621 Oslo 6, Norway

ORDER FORM

I would like to order

..... Ring Binders, 30 mm, at nkr 20,- per binder

..... Ring Binders, 40 mm, at nkr 25,- per binder

..... Plastic Covers at nkr 10,- per cover

Name .....

Company .....

Address .....

.....

City .....

PRINTING RECORD

Printing	Notes
12/84	VERSION 01

ND-500/2 Double Precision Array Processing Functions
Publ.No. ND 05.018.01
December 1984

Norsk Data

Norsk Data A.S
Graphic Center
P.O.Box 25, Bogerud
0621 Oslo 6, Norway

Manuals can be updated in two ways, new versions and revisions. New Iversions consist of a complete new manual which replaces the old manual. New versions incorporate all revisions since the previous version. Revisions consist of one or more single pages to be merged into the manual by the user, each revised page being listed on the new printing record sent out with the revision. The old printing record should be replaced by the new one.

New versions and revisions are announced in the ND Bulletin and can be ordered as described below.

The reader's comments form at the back of this manual can be used both to report errors in the manual and to give an evaluation of the manual. Both detailed and general comments are welcome.

These forms and comments should be sent to:

Documentation Department
Norsk Data A.S
P.O. Box 25, Bogerud
0621 Oslo 6, Norway

Requests for documentation should be sent to the local ND office or (in Norway) to:

Graphic Center
Norsk Data A.S
P.O. Box 25, Bogerud
0621 Oslo 6, Norway

Preface:

THE PRODUCT

This manual describes the mathematical functions in the ND-500-2-APD library.

The APD library routines may be called from FORTRAN. The routines are designed to speed up the execution of operations on double precision real arrays.

The APD library utilizes a special microprogram for the ND-500/2 CPU. The special double precision functions within this microprogram are called ND-500/2 DAX functions. This microprogram is an extension of the CXA microprogram.

The ND-numbers for the product on the different ND-500/2 models are:

Computer	ND-number	Microprogram version
ND-550/2	ND-10786	152xx
ND-560/2	ND-10786	152xx
ND-570/2 (new CX)	ND-10786	152xx
ND-570/2 (old CXA)	ND-10786	152xx

NOTE The ND-530/2 can not use this product.

THE MANUAL

This manual provides a functional description of the APD library, and thereby the ND-500/2 DAX functions. A listing of each routine is used to describe the routines. Listing and parameter description is done in FORTRAN.

THE READER

This manual is written for people creating and running programs using array processing functions.

PREREQUISITE KNOWLEDGE

The reader is assumed to be familiar with FORTRAN. It is also assumed that the reader is familiar with using ND-500/2 computers, and knows how to generate, load and run programs on such computers. Some knowledge of the mathematical theory of the implemented functions is of advantage, to be able to use them effectively.

RELATED MANUALS

Documentation further describing the use of ND-500 computers is found in these manuals:

ND-500 Loader/Monitor	ND-60.136
ND FORTRAN Reference Manual	ND-60.145
ND-500 Reference Manual	ND-05.009
ND-500 Single Precision Array Processing Functions	ND-05.013

TABLE OF CONTENTS

Section Page

1 BASIC CONCEPTS

1.1 PREREQUISITE SOFTWARE FOR USING THE APD LIBRARY

Together with the microprogram, the APD library must be installed. The actual file name is: ND-500-2-APD:NRF.

These are the only modifications to be performed on the system.

The APD library utilizes a special microprogram for the ND-500/2 CPU. This microprogram contains the double precision array processing functions. It is an extension of the ND-500/2 CXA microprogram.

The ND-500/2 DAX functions are floating point operations performed in 64 bits floating point format by the ND-500/2 floating point arithmetic.

1.2 THE DOUBLE PRECISION FLOATING POINT FORMAT

The number range for the double precision floating point format is:

$$8.6 * 10^{-78} \leq |N| \leq 5.8 * 10^{76}$$

The accuracy corresponds approximately to 16 decimal digits.

1.3 HARDWARE CONCEPTS

The ND-500 CPU gives possibility of parallel processing. This means that indexing, memory access, floating point arithmetic, integer arithmetic and loop control may be run in parallel. This is done as far as possible to obtain high speed operations. Temporary results to be used in later calculations, are kept in registers in the ND-500/2 CPU, accessed directly by both the floating point arithmetic and integer arithmetic. In this way unnecessary memory accesses are avoided.

Arrays involved in an operation are accessed through the ND-500/2 memory management system. This system willautomatically cause allocation of memory and reservation of continuous memory space for array processing is not required. The result of an array processing function is present in output array when returning from the function.

The ND-500/2 DAX functions are fully interruptable to maintain the ND-500/2 CPU resources being shared by the different processes currently running on the system.

1.4 ARRAY PROCESSING DEFINITIONS

An array contains a group of numbers that are related to each other in some way, and the array may be multi-dimensional. An array is termed a matrix in mathematical terminology.

An array is used to represent equations of different kinds, for example, linear equations. Each row in the array represents one particular equation, thus the array represents a system of equations of the same kind.

The entries in the array are the coefficients of the equations. They are called elements in this manual.

Each row of elements is named a vector in this manual, regardless of what kind of equation it mathematically represents. In those array processing functions which are closely related to mathematical vectors, the row of elements is referred to as a complex vector.

Most of the DAX functions are performed on one-dimensional vectors.

Example of Equations:

Mathematical equations:		Arrays:
$2x + 8y + 5z = 24$ $3x + 2y + 1z = 0$ $11x + 0y + 5z = 4$	⟶	$\begin{bmatrix} 2 & 8 & 5 & 24 \\ 3 & 2 & 1 & 0 \\ 11 & 0 & 5 & 4 \end{bmatrix}$
$2x_2^2 + y = 3$ $x^2 + y = 0$	⟶	$\begin{bmatrix} 2 & 1 & 3 \\ 1 & 1 & 0 \end{bmatrix}$

Three parameters are necessary to specify a vector:

V - The logical name of the vector. A double precision real.

INC - The step value (increment) for the index in the array.
An integer.

NN - Element count. Number of elements in the array. An integer.

NOTE It is assumed that the lower limit for the array index is 1!

Example of Use of a DAX Function:

```
PROGRAM CLEAR

DOUBLE PRECISION VA(2000); % Remember to declare the vector as
INTEGER INCA,NN;           % double precision real.

<Other program statements>

INCA = 2
NN = 1000

CALL VCLR(VA,INCA,NN)

<Other program statements>

END

This DAX function causes each second element of the vector VA
to be set to zero (increment is 2).
```

For vectors where the elements are stored in consecutive locations, the index increment is equal to 1. The flexibility to specify index increments is present to most of the functions.

For complex vectors, each complex equation is represented by two consecutive elements. This corresponds to the real and the imaginary part of the complex vector. The real element is immediately followed by the imaginary element, as the complex vector is represented in the rectangular coordinate system.

This means that for each index in the array there are one real and one imaginary element.

Example of Use of a Complex DAX Function:

```
PROGRAM CVMULIPLY

COMPLEX*16 VA(1:200)
COMPLEX*16 VB(1:200)
COMPLEX*16 VC(1:200)

<Other program statements>

CALL CVMUL(VA,1,VB,2,VC,1,100,1)

<Other program statements>

END
```

This DAX function causes each complex vector of the array VA to be multiplied with each second vector from VB. The results are stored in array VC. This function corresponds to mathematical multiplication of complex numbers.

2 USING THE ND-500/2 DAX FUNCTIONS

The ND-500/2 DAX functions can be called from FORTRAN. The array processing library is used to transfer the parameters from the call to the array processing instructions. Thus the array processing functions are connected to the main program at load time as a part of the program.

Writing a FORTRAN program for array processing with ND-500/2 DAX functions is much the same as using the FORTRAN equivalent for the array processing function. Before starting an array processing function, input and output arrays for the operations must be defined. Initialization of input arrays is also required. This means that data for processing must be placed in the input arrays for the actual array processing function. Then the array processing function may be called. The result of the operation is present in the output array when returning from an array processing function.

Example of Creating a FORTRAN Program Using DAX Functions

Source Program in FORTRAN:

```
          PROGRAM DOKKT

C         Set name and size of arrays to be used.
          PARAMETER NN = 100; NC = 20
          IMPLICIT DOUBLE PRECISION (A-H,O-Z)
          DIMENSION VA(NN),VB(NN),VC(NN),VL(NC)
          INTEGER*4 IL(NC)
C         Initiate scalar values, index increment and element count
C         for the VRAMP function.
          DELTA = 1.0D+01
          SC1   = 1.5
          SC2   = 2.0D-01
          INC1  =   1
          INC2  =   2
          INC4  =   4
C         Initiate VA with a ramp.
          CALL VRAMP(SC1,SC2,VA,INC1,NN)
C         Clear result array to be used in next operation.
          CALL VCLR(VC,INC1,NN)
          NB  = NN/2
          DO 100 M=1,NB
             CALL SVE(VA(M),INC1,VC(M),NB)
100       CONTINUE
C         Set scalar value.
          B     = 200.0
C         200.0 divided with each second element of VA.
C         Result in VB.
```

The program continues on next page....

....Continued from previous page

```
          CALL VDIVS(VA,INC2,B,VB,INC1,NC)
C         Set scalar value.
          B    = 150.0
C         Add 150.0 to each second of VA. Result in VB.
          CALL VSADD(VA,INC2,B,VB,INC1,NC)
C         Add elements from VB and VC with result in VA.
          CALL VADD(VB,INC4,VC,INC1,VA,INC1,NC)
          B    = 2.00D+03
C         2000.0 divided with each fourth element in VA.
C         Result in VA.
          CALL VDIVS(VA,INC4,B,VA,INC2,NC)
          B    = 2.50D-02
C         Each fourth element in VB is multiplied with 2.5E-2, and
C         added with each element of VA.
C         Result in VB.
          CALL VSMA(VB,INC4,B,VA,INC1,VB,INC1,NC)
C         Negate each second element of VB.
C         Result in VB.
          CALL VNEG(VB,INC2,VB,INC2,NC)
C         Move each element of VB to VL (only NC elements).
          CALL VMOV(VB,INC1,VL,INC1,NC)
C         Find maximum value in VB.
C         Value returned in VMA and index returned in IMA.
          CALL MAXV(VB,INC1,VMA,IMA,NC)
C         Maximum value + delta.
          VMA = VMA+DELTA
C         Sort the elements in rising order.
          DO 120 M=1,NC
             CALL MINV(VL,INC1,VA(M),IL(M),NC)
             IY = IL(M)
             VL(IY) = VMA
120     CONTINUE
          WRITE (1,1000)
          WRITE (1,1001)
          DO FOR I=1,NC
             WRITE (1,1002)I,VB(I),I,VA(I),IL(I)
          ENDDO
          WRITE (1,1001)
1000      FORMAT (' Finding minimum value and indices',/,
     + ' Input vector results in output vector found at VB index:')
1001      FORMAT (,
     + ' ..................................................')
1002      FORMAT (1X,'VB(',I2,')',F15.6,'   VA(',I2,')',F15.6,
     + ' at index ',I3)
          END
```

Compiling the Program:

```
@FORTRAN-500 ↵

ND-500 ANSI 77 FORTRAN COMPILER - 203054F
FTN: COMPILE ↵

SOURCE-FILE DOKKT ↵

LIST-FILE ↵

OBJECT-FILE "DOKKT" ↵

ND-500 ANSI 77 FORTRAN COMPILER - 203054F    8:45  20 SEP 1984
SOURCE FILE:   DOKKT
- CPU TIME USED: 0.8 SECONDS.  71 LINES COMPILED.
- NO MESSAGES
- PROGRAM SIZE=567 DATA SIZE=3188 COMMON SIZE=0

FTN: EXIT ↵
```

Loading the Program:

```
@LINKAGE-LOADER ↵

ND-Linkage-Loader -  F          10. September  1983 Time: 00:07
Nll entered:                    20. September  1984 Time:  8:45
Nll: SET-DOMAIN DOKKT ↵

Nll: OPEN-SEGMENT DOKKT ,,, ↵

Nll: LOAD DOKKT,ND-500-2-APD-LIB ↵
Program:.........1073 P05   Data:............6170 D05
ND-500/2-APD-LIB-A
Program:.........1616 P05   Data:............7100 D05
Nll: CLOSE Y ↵
Segment no........30         is linked

-------------------------------------------------------------

20. September  1984 Time:  8:45

Unsatisfied references :

None!

-------------------------------------------------------------

Defined symbols :

DOKKT..............4 P05   VADD............1073 P05
VMOV............1136 P05   SVE.............1171 P05
MAXV............1224 P05   MINV............1273 P05
VSADD...........1342 P05   VDIVS...........1400 P05
VCLR............1436 P05   VSMA............1461 P05
VNEG............1530 P05   VRAMP...........1564 P05

Program:.........1616 P05   Data:..........11100 D05

Nll: EXIT ↵
```

Executing the Program:

```
@ND-500-MONITOR ↵

ND-500 MONITOR  VERSION E    83.12.20 / 83.12.21

N500: DOKKT ↵

Finding minimum value and indices
Input vector results in output vector found at VB index:
.....................................................
VB( 1)      -8.029282   VA( 1)    -363.636364 at index  11
VB( 2)     486.927500   VA( 2)    -317.460317 at index  13
VB( 3)      -7.729249   VA( 3)    -281.690141 at index  15
VB( 4)     510.207500   VA( 4)    -253.164557 at index  17
VB( 5)      -8.947500   VA( 5)    -229.885057 at index  19
VB( 6)     370.000000   VA( 6)      -8.947500 at index   5
VB( 7)      -4.545455   VA( 7)      -8.029282 at index   1
VB( 8)     390.000000   VA( 8)      -7.729249 at index   3
VB( 9)      -4.166667   VA( 9)      -4.545455 at index   7
VB(10)     410.000000   VA(10)      -4.166667 at index   9
VB(11)    -363.636364   VA(11)     370.000000 at index   6
VB(12)     430.000000   VA(12)     390.000000 at index   8
VB(13)    -317.460317   VA(13)     410.000000 at index  10
VB(14)     450.000000   VA(14)     430.000000 at index  12
VB(15)    -281.690141   VA(15)     450.000000 at index  14
VB(16)     470.000000   VA(16)     470.000000 at index  16
VB(17)    -253.164557   VA(17)     486.927500 at index   2
VB(18)     490.000000   VA(18)     490.000000 at index  18
VB(19)    -229.885057   VA(19)     510.000000 at index  20
VB(20)     510.000000   VA(20)     510.207500 at index   4

N500: EXIT ↵
```

3 ND-500/2 DAX FUNCTIONS PERFORMANCE

NAME	OPERATION	Typical execution time pr. loop (microsec.) Improvement ratio to FTN function)[1] Number of elements →	↓	↓
CVMUL	COMPLEX VECTOR MULTIPLY	750	4.88	5.30
DOTPR	DOT PRODUCT	750	4.13	1.61
MAXMGV	MAX. MAGNITUDE VALUE IN VECTOR	750	4.75	1.13
MAXMIN	MAX. AND MIN. VALUE IN VECTOR	750	4.00	1.73
MAXV	MAX. VALUE IN VECTOR	750	4.03	1.16
MEAMGV	MEAN MAGNITUDE OF VECTOR	750	5.70	0.83
MINMGV	MIN. MAGNITUDE VALUE IN VECTOR	750	4.80	1.13
MINV	MIN. VALUE IN VECTOR	750	4.03	1.15
MXMNMG	MAX. AND MIN. MAG. VALUE IN VECTOR	750	4.15	1.75
SVE	SUM OF VECTOR ELEMENTS	750	5.10	0.83
SVEMG	SUM OF VECTOR ELEMENTS MAGNITUDE	750	5.90	0.80
SVESQ	SUM OF VECTOR ELEMENTS SQUARE	750	3.53	1.40
SVS	SUM OF VECTOR ELEMENTS SIGNED SQUARE	750	4.04	1.40
VABS	VECTOR ABSOLUTE VALUE	750	4.30	1.19
VADD	VECTOR ADD	750	4.11	1.75
VCLR	VECTOR CLEAR	750	3.71	0.85
VDIV	VECTOR DIVIDE	750	2.46	3.63
VDIVS	VECTOR SCALAR DIVIDE	750	2.34	3.36
VFLNZ	VECTOR FIRST AND LAST NON-ZERO VALUE	750	8.00	0.51
VMAX	VECTOR MAXIMUM	750	3.90	2.03
VMAXMG	VECTOR MAXIMUM MAGNITUDE	750	4.05	2.07
VMIN	VECTOR MINIMUM	750	3.90	2.03
VMINMG	VECTOR MINIMUM MAGNITUDE	750	4.05	2.07
VMOV	VECTOR MOVE	750	4.20	1.22
VMUL	VECTOR MULTIPLY	750	4.41	1.75
VNEG	VECTOR NEGATIVE	750	4.49	1.22
VRAMP	VECTOR RAMP FUNCTION	750	5.70	0.96
VSADD	VECTOR SCALAR ADD	750	4.30	1.40
VSMA	VECTOR SCALAR MULTIPLY AND ADD	750	4.05	1.93
VSMUL	VECTOR SCALAR MULTIPLY	750	4.30	1.40
VSQ	VECTOR SQUARE	750	3.48	1.55
VSQRT	VECTOR SQUARE ROOT	750	1.45	10.78
VSSQ	VECTOR SIGNED SQUARE	750	4.00	1.52
VSUB	VECTOR SUBTRACT	750	4.41	1.75
VSWAP	VECTOR SWAP	750	3.45	2.27
VTAPER	VECTOR TAPER	750	2.64	1.95

Timing measurement is done on a ND-570 system with 64K-byte cache.

[1]) (Time used by machine code) / (Time used by microcode).

The next table shows how the ND-500/2 DAX functions may be used to improve the performance even on small arrays. With high cache hit ratio, an improvement ratio of 4.4 can be obtained with the VADD function, while low cache hit ratio will reduce the improvement ratio over FORTRAN.

Element count	Improvement ratio to FTN for function:		
	VADD	DOTPR	MAXV
5	1.6	1.7	1.4
10	2.2	2.2	2.1
50	3.5	3.5	2.8
100	3.8	3.9	3.6
500	4.1	4.3	4.0
1000	4.4	4.3	4.3
2000	2.5	3.7	4.1

4 ARRAY PROCESSING FUNCTIONS

4.1 INTRODUCTION

This chapter contains a listing of each array processing function to describe the different functions. For each routine the required parameter list for calling the array processing function is included together with definitions.

The ND-500/2 double precision array processing functions are implemented as one instruction.

The different functions are using the same instruction code. The contents of the record register are the only difference between the functions and are used to distinguish between them.

For each routine, an identification number is given as a cross reference between the object code and the array processing function. This identification number is given as two octal numbers :
'Ident (R:I) : xxx:nnnnnnB'. 'xxx' are the contents of the record register. 'nnnnnn' is the instruction code used for the DAX function.

The library for the ND-500 double precision array processing functions consists of one routine for each of the array processing functions. Each routine is building a data stack depending on the called array processing function. The data stack is used by the array processing functions to find addresses of input and output arrays, scalar values or addresses, index increments and element count.

4.2 VECTOR ADD (VADD)

Format

VADD(VA,INCA,VB,INCB,VC,INCC,NN) Ident (R:I) : 001:177515B

Explanation

Add the corresponding elements of two vectors. VCn = VAn + VBn, 'n' is the element index.

Parameters

VA : Name of input vector VA.
INCA : VA index increment.
VB : Name of input vector VB.
INCB : VB index increment.
VC : Name of output vector VC.
INCC : VC index increment.
NN : Element count.

Listing

```
      SUBROUTINE VADD(VA,INCA,VB,INCB,VC,INCC,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1),VB(1),VC(1)
      IA = 1
      IB = 1
      IC = 1
      DO FOR M = 1,NN
         VC(IC) = VB(IB) + VA(IA)
         IA = IA + INCA
         IB = IB + INCB
         IC = IC + INCC
      ENDDO
      RETURN
      END
```

4.3 VECTOR SUBTRACT (VSUB)

Format

VSUB(VA,INCA,VB,INCB,VC,INCC,NN) Ident (R:I) : 002:177515B

Explanation

Subtract the corresponding elements of two vectors. VCn = VBn - VAn, 'n' is the element index.

Parameters

VA : Name of vector VA.
INCA : VA index increment.
VB : Name of vector VB.
INCB : VB index increment.
VC : Name of vector VC.
INCC : VC index increment.
NN : Element count.

Listing

```
      SUBROUTINE VSUB(VA,INCA,VB,INCB,VC,INCC,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1),VB(1),VC(1)
      IA = 1
      IB = 1
      IC = 1
      DO FOR M = 1,NN
         VC(IC) = VB(IB) - VA(IA)
         IA = IA + INCA
         IB = IB + INCB
         IC = IC + INCC
      ENDDO
      RETURN
      END
```

4.4 VECTOR MULTIPLY (VMUL)

Format

VMUL(VA,INCA,VB,INCB,VC,INCC,NN) Ident (R:I) : 003:177515B

Explanation

Multiply the corresponding elements of two vectors. VCn = VBn * VAn, 'n' is the element index.

Parameters

VA	: Name of input vector VA.
INCA	: VA index increment.
VB	: Name of input vector VB.
INCB	: VB index increment.
VC	: Name of output vector VC.
INCC	: VC index increment.
NN	: Element count.

Listing

```
      SUBROUTINE VMUL(VA,INCA,VB,INCB,VC,INCC,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1),VB(1),VC(1)
      IA = 1
      IB = 1
      IC = 1
      DO FOR M = 1,NN
         VC(IC) = VB(IB) * VA(IA)
         IA = IA + INCA
         IB = IB + INCB
         IC = IC + INCC
      ENDDO
      RETURN 1
      END
```

4.5 VECTOR DIVIDE (VDIV)

Format

VDIV(VA,INCA,VB,INCB,VC,INCC,NN) Ident (R:I) : 004:177515B

Explanation

Divide the corresponding elements of two vectors. Cn = VBn/VAn, 'n' is the element index.

Parameters

VA : Name of input vector VA.
INCA : VA index increment.
VB : Name of input vector VB.
INCB : VB index increment.
VC : Name of output vector VC.
INCC : VC index increment.
NN : Element count.

Listing

```
      SUBROUTINE VDIV(VA,INCA,VB,INCB,VC,INCC,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1),VB(1),VC(1)
      IA = 1
      IB = 1
      IC = 1
      DO FOR M = 1,NN
         VC(IC) = VB(IB) / VA(IA)
         IA = IA + INCA
         IB = IB + INCB
         IC = IC + INCC
      ENDDO
      RETURN 1
      END
```

4.6 VECTOR MAXIMUM (VMAX)

Format

VMAX(VA,INCA,VB,INCB,VC,INCC,NN) Ident (R:I) : 005:177515B

Explanation

Form a vector from the maximum value of each corresponding pair of elements of two vectors. VCn = VAn if VAn > VBn, else VCn = VBn. 'n' is the element index.

Parameters

VA : Name of input vector VA.
INCA : VA index increment.
VB : Name of input vector VB.
INCB : VB index increment.
VC : Name of output vector VC.
INCC : VC index increment.
NN : Element count.

Listing

```
      SUBROUTINE VMAX(VA,INCA,VB,INCB,VC,INCC,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1),VB(1),VC(1)
      IA = 1
      IB = 1
      IC = 1
      DO FOR M = 1,NN
         VC(IC) = DMAX1(VA(IA),VB(IB))
         IA = IA + INCA
         IB = IB + INCB
         IC = IC + INCC
      ENDDO
      RETURN
      END
```

4.7 VECTOR MINIMUM (VMIN)

Format

VMIN(VA,INCA,VB,INCB,VC,INCC,NN) Ident (R:I) : 006:177515B

Explanation

Form a vector from the minimum value of each corresponding pair of elements of two vectors. VCn = VAn if VAn < VBn, else VCn = VBn. 'n' is the element index.

Parameters

VA : Name of input vector VA.
INCA : VA index increment.
VB : Name of input vector VB.
INCB : VB index increment.
VC : Name of output vector VC.
INCC : VC index increment.
NN : Element count.

Listing

```
      SUBROUTINE VMIN(VA,INCA,VB,INCB,VC,INCC,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1),VB(1),VC(1)
      IA = 1
      IB = 1
      IC = 1
      DO FOR M = 1,NN
         VC(IC) = DMIN1(VA(IA),VB(IB))
         IA = IA + INCA
         IB = IB + INCB
         IC = IC + INCC
      ENDDO
      RETURN
      END
```

4.8 VECTOR MAXIMUM MAGNITUDE (VMAXMG)

Format

VMAXMG(VA,INCA,VB,INCB,VC,INCC,NN) Ident (R:I) : 007:177515B

Explanation

Form a vector from the maximum absolute value of each corresponding pair of elements of two vectors. VCn = |VAn| if |VAn| > |VBn|, else VCn = |VBn|. 'n' is the element index.

Parameters

VA	: Name of input vector VA.
INCA	: VA index increment.
VB	: Name of input vector VB.
INCB	: VB index increment.
VC	: Name of output vector VC.
INCC	: VC index increment.
NN	: Element count.

Listing

```
      SUBROUTINE VMAXMG(VA,INCA,VB,INCB,VC,INCC,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1),VB(1),VC(1)
      IA = 1
      IB = 1
      IC = 1
      DO FOR M = 1,NN
         VC(IC) = DMAX1(ABS(VA(IA)),ABS(VB(IB)))
         IA = IA + INCA
         IB = IB + INCB
         IC = IC + INCC
      ENDDO
      RETURN
      END
```

4.9 VECTOR MINIMUM MAGNITUDE (VMINMG)

Format

VMINMG(VA,INCA,VB,INCB,VC,INCC,NN) Ident (R:I) : 010:177515B

Explanation

Form a vector from the minimum absolute value of each corresponding pair of elements of two vectors. VCn = |VAn| if |VAn| < |VBn|, else VCn = |VBn|. 'n' is the element index.

Parameters

VA : Name of input vector VA.
INCA : VA index increment.
VB : Name of input vector VB.
INCB : VB index increment.
VC : Name of output vector VC.
INCC : VC index increment.
NN : Element count.

Listing

```
      SUBROUTINE VMINMG(VA,INCA,VB,INCB,VC,INCC,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1),VB(1),VC(1)
      IA = 1
      IB = 1
      IC = 1
      DO FOR M = 1,NN
         VC(IC) = DMIN1(ABS(VA(IA)),ABS(VB(IB)))
         IA = IA + INCA
         IB = IB + INCB
         IC = IC + INCC
      ENDDO
      RETURN
      END
```

4.10 VECTOR SQUARE (VSQ)

Format

VSQ(VA,INCA,VC,INCC,NN) Ident (R:I) : 042:177515B

Explanation

Square the elements of a vector. VCn = $(VAn)^2$. 'n' is the element index.

Parameters

VA : Name of input vector VA.
INCA : VA index increment.
VC : Name of output vector VC.
INCC : VC index increment.
NN : Element count.

Listing

```
      SUBROUTINE VSQ(VA,INCA,VC,INCC,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1),VC(1)
      IA = 1
      IC = 1
      DO FOR M = 1,NN
         VC(IC) = VA(IA)**2
         IA = IA + INCA
         IC = IC + INCC
      ENDDO
      RETURN
      END
```

4.11 VECTOR SIGNED SQUARE (VSSQ)

Format

VSSQ(VA,INCA,VC,INCC,NN) Ident (R:I) : 011:177515B

Explanation

Multiply each element of a vector with the absolute value of itself. VCn = VAn * |VAn|. 'n' is the element index.

Parameters

VA : Name of input vector VA.
INCA : VA index increment.
VC : Name of output vector VC.
INCC : VC index increment.
NN : Element count.

Listing

```
      SUBROUTINE VSSQ(VA,INCA,VC,INCC,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1),VC(1)
      IA = 1
      IC = 1
      DO FOR M = 1,NN
         VC(IC) = VA(IA)*ABS(VA(IA))
         IA = IA + INCA
         IC = IC + INCC
      ENDDO
      RETURN
      END
```

4.12 VECTOR ABSOLUTE VALUE (VABS)

Format

VABS(VA,INCA,VC,INCC,NN) Ident (R:I) : 012:177515B

Explanation

Form a vector from the absolute value of the elements in a vector.
VCn = |VAn|. 'n' is the element index.

Parameters

VA : Name of input vector VA.
INCA : VA index increment.
VC : Name of output vector VC.
INCC : VC index increment.
NN : Element count.

Listing

```
      SUBROUTINE VABS(VA,INCA,VC,INCC,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1),VC(1)
      IA = 1
      IC = 1
      DO FOR M = 1,NN
         VC(IC) = ABS(VA(IA))
         IA = IA + INCA
         IC = IC + INCC
      ENDDO
      RETURN
      END
```

4.13 VECTOR SQUARE ROOT (VSQRT)

Format

VSQRT(VA,INCA,VC,INCC,NN) Ident (R:I) : 013:177515B

Explanation

Take the square roots of the elements in a vector. VCn = √VAn. 'n' is the element index.

Parameters

VA : Name of input vector VA.
INCA : VA index increment.
VC : Name of output vector VC.
INCC : VC index increment.
NN : Element count.

Listing

```
      SUBROUTINE VSQRT(VA,INCA,VC,INCC,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1),VC(1)
      IA = 1
      IC = 1
      DO FOR M = 1,NN
         VC(IC) = SQRT(VA(IA))
         IA = IA + INCA
         IC = IC + INCC
      ENDDO
      RETURN
      END
```

4.14 VECTOR MOVE (VMOV)

Format

VMOV(VA,INCA,VC,INCC,NN) Ident (R:I) : 016:177515B

Explanation

Move the elements from one vector into another. VCn = VAn, 'n' is the element index.

Parameters

VA : Name of input vector VA.
INCA : VA index increment.
VC : Name of output vector VC.
INCC : VC index increment.
NN : Element count.

Listing

```
      SUBROUTINE VMOV(VA,INCA,VC,INCC,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1),VC(1)
      IA = 1
      IC = 1
      DO FOR M = 1,NN
         VC(IC) = VA(IA)
         IA = IA + INCA
         IC = IC + INCC
      ENDDO
      RETURN
      END
```

4.15 VECTOR SWAP (VSWAP)

Format

VSWAP(VA,INCA,VC,INCC,NN) Ident (R:I) : 063:177515B

Explanation

Swap the elements between two vectors. VAn → VBn and VBn → VAn, 'n' is the element index.

Parameters

VA : Name of input and output vector VA.
INCA : VA index increment.
VC : Name of input and output vector VC.
INCC : VC index increment.
NN : Element count.

Listing

```
      SUBROUTINE VSWAP(VA,INCA,VC,INCC,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1),VC(1)
      IA = 1
      IC = 1
      DO FOR M = 1,NN
         HOLD = VC(IC)
         VC(IC) = VA(IA)
         VA(IA) = HOLD
         IA = IA + INCA
         IC = IC + INCC
      ENDDO
      RETURN
      END
```

4.16 VECTOR NEGATIVE (VNEG)

Format

VNEG(VA,INCA,VC,INCC,NN) Ident (R:I) : 064:177515B

Explanation

Form a vector from the elements of another vector multiplied with -1.
VCn = -VAn, 'n' is the element index.

Parameters

VA : Name of input vector VA.
INCA : VA index increment.
VC : Name of output vector VC.
INCC : VC index increment.
NN : Element count.

Listing

```
      SUBROUTINE VNEG(VA,INCA,VC,INCC,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1),VC(1)
      IA = 1
      IC = 1
      DO FOR M = 1,NN
         VC(IC) = - VA(IA)
         IA = IA + INCA
         IC = IC + INCC
      ENDDO
      RETURN
      END
```

4.17 SUM OF VECTOR ELEMENTS (SVE)

Format

SVE(VA,INCA,VC,NN) Ident (R:I) : 021:177515B

Explanation

Add the elements of a vector. VC = VA_1 + VA_2 + ...+ VAnn, 'nn' is the element count.

Parameters

VA : Name of input vector VA.
INCA : VA index increment.
VC : Name of output scalar VC.
NN : Element count.

Listing

```
      SUBROUTINE SVE(VA,INCA,VC,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1)
      IA = 1
      SUM = 0.0
      DO FOR M = 1,NN
         SUM = SUM + VA(IA)
         IA = IA + INCA
      ENDDO
      VC = SUM
      RETURN
      END
```

4.18 SUM OF VECTOR ELEMENTS MAGNITUDE (SVEMG)

Format

SVEMG(VA,INCA,VC,NN) Ident (R:I) : 065:177515B

Explanation

Form the sum of the absolute values of the elements of a vector.
VC = $|VA_1| + |VA_2| + ...+ |VAnn|$, 'nn' is the element count.

Parameters

VA : Name of input vector VA.
INCA : VA index increment.
VC : Name of output scalar VC.
NN : Element count.

Listing

```
      SUBROUTINE SVEMG(VA,INCA,VC,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1)
      IA = 1
      SUM = 0.0
      DO FOR M = 1,NN
         SUM = SUM + ABS(VA(IA))
         IA = IA + INCA
      ENDDO
      VC = SUM
      RETURN
      END
```

4.19 SUM OF VECTOR ELEMENTS SQUARE (SVESQ)

Format

SVESQ(VA,INCA,VC,NN) Ident (R:I) : 066:177515B

Explanation

Form the sum of the squared elements of a vector.
VC = $(VA_1)^2$ + $(VA_2)^2$ + ...+ $(VAnn)^2$, 'nn' is the element count.

Parameters

VA : Name of input vector VA.
INCA : VA index increment.
VC : Name of output scalar VC.
NN : Element count.

Listing

```
      SUBROUTINE SVESQ(VA,INCA,VC,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1)
      IA = 1
      SUM = 0.0
      DO FOR M = 1,NN
         SUM = SUM + VA(IA)**2
         IA = IA + INCA
      ENDDO
      VC = SUM
      RETURN
      END
```

4.20 SUM OF VECTOR ELEMENTS SIGNED SQUARE (SVS)

Format

SVS(VA,INCA,VC,NN) Ident (R:I) : 022:177515B

Explanation

Form the sum of the elements of a vector, where each element at first is multiplied with the absolute value of itself.
VC = $VA_1 * |VA_1| + VA_2 * |VA_2| + ... + VAnn * |VAnn|$, 'nn' is the element count.

Parameters

VA : Name of input vector VA.
INCA : VA index increment.
VC : Name of output scalar VC.
NN : Element count.

Listing

```
      SUBROUTINE SVS(VA,INCA,VC,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1)
      IA = 1
      SUM = 0.0
      DO FOR M = 1,NN
         SUM = SUM + VA(IA)*ABS(VA(IA))
         IA = IA + INCA
      ENDDO
      VC = SUM
      RETURN
      END
```

4.21 MEAN MAGNITUDE VALUE OF VECTOR (MEAMGV)

Format

MEAMGV(VA,INCA,VC,NN) Ident (R:I) : 023:177515B

Explanation

Form the mean value of the absolute values of the elements of a vector. VC = ($|VA_1|$ + $|VA_2|$ + ...+ |VAnn|) / nn , 'nn' is the element count.

Parameters

VA : Name of input vector VA.
INCA : VA index increment.
VC : Name of output scalar VC.
NN : Element count.

Listing

```
SUBROUTINE MEAMGV(VA,INCA,VC,NN)
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
DIMENSION VA(1)
IA = 1
SUM = 0.0
DO FOR M = 1,NN
   SUM = SUM + ABS(VA(IA))
   IA = IA + INCA
ENDDO
VC = SUM/NN
RETURN
END
```

4.22 MAXIMUM VALUE IN VECTOR (MAXV)

Format

MAXV(VA,INCA,VC,IC,NN) Ident (R:I) : 024:177515B

Explanation

Scan a vector for its element with maximum value and return this together with the corresponding index.

Parameters

VA : Name of input vector VA.
INCA : VA index increment.
VC : Name of output scalar VC, containing max. value.
IC : Index in VA for max. value.
NN : Element count.

Listing

```
      SUBROUTINE MAXV(VA,INCA,VC,IC,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1)
      IA = 1
      VC = VA(IA)
      IC = IA
      DO FOR M = 2,NN
         IA = IA + INCA
         IF (VA(IA) .GT. VC) THEN
           VC = VA(IA)
           IC = IA
         ENDIF
      ENDDO
      RETURN
      END
```

4.23 MINIMUM VALUE IN VECTOR (MINV)

Format

MINV(VA,INCA,VC,IC,NN) Ident (R:I) : 025:177515B

Explanation

Scan a vector for its element with minimum value and return this together with the corresponding index.

Parameters

VA : Name of input vector VA.
INCA : VA index increment.
VC : Name of output scalar VC, containing min. value.
IC : Index in VA for min. value.
NN : Element count.

Listing

```
      SUBROUTINE MINV(VA,INCA,VC,IC,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1)
      IA = 1
      VC = VA(IA)
      IC = IA
      DO FOR M = 2,NN
         IA = IA + INCA
         IF (VA(IA) .LT. VC) THEN
           VC = VA(IA)
           IC = IA
         ENDIF
      ENDDO
      RETURN
      END
```

4.24 MAXIMUM MAGNITUDE VALUE IN VECTOR (MAXMGV)

Format

MAXMGV(VA,INCA,VC,IC,NN) Ident (R:I) : 026:177515B

Explanation

Scan a vector for its element with maximum absolute value, and return this together with the corresponding index.

Parameters

VA : Name of input vector VA.
INCA : VA index increment.
VC : Name of output scalar VC, containing max. absolute value.
IC : Index in VA for max. absolute value.
NN : Element count.

Listing

```
      SUBROUTINE MAXMGV(VA,INCA,VC,IC,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1)
      IA = 1
      VC = ABS(VA(IA))
      IC = IA
      DO FOR M = 2,NN
         IA = IA + INCA
         VAABS = ABS(VA(IA))
         IF (VAABS .GT. VC) THEN
           VC = VAABS
           IC = IA
         ENDIF
      ENDDO
      RETURN
      END
```

4.25 MINIMUM MAGNITUDE VALUE IN VECTOR (MINMGV)

Format

MINMGV(VA,INCA,VC,IC,NN) Ident (R:I) : 027:177515B

Explanation

Scan a vector for its element with minimum absolute value, and return this together with the corresponding index.

Parameters

VA : Name of input vector VA.
INCA : VA index increment.
VC : Name of output scalar VC, containing min. absolute value.
IC : Index in VA for min. absolute value.
NN : Element count.

Listing

```
      SUBROUTINE MINMGV(VA,INCA,VC,IC,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1)
      IA = 1
      VC = ABS(VA(IA))
      IC = IA
      DO FOR M = 2,NN
         IA = IA + INCA
         VAABS = ABS(VA(IA))
         IF (VAABS .LT. VC) THEN
           VC = VAABS
           IC = IA
         ENDIF
      ENDDO
      RETURN
      END
```

4.26 MAXIMUM AND MINIMUM VALUE IN VECTOR (MAXMIN)

Format

MAXMIN(VA,INCA,VC,IC,VD,ID,NN) Ident (R:I) : 030:177515B

Explanation

Scan a vector for its element with minimum value and its element with maximum value, and return these together with the corresponding indices.

Parameters

VA : Name of input vector VA.
INCA : VA index increment.
VC : Name of output scalar VC, containing max. value.
IC : Index in VA for max. value.
VD : Name of output scalar VC, containing min. value.
ID : Index in VA for min. value.
NN : Element count.

Listing

```
      SUBROUTINE MAXMIN(VA,INCA,VC,IC,VD,ID,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1)
      IA = 1
      VC = VA(IA)
      VD = VA(IA)
      IC = IA
      ID = IA
      DO FOR M = 2,NN
         IA = IA + INCA
         IF (VA(IA) .GT. VC) THEN
           VC = VA(IA)
           IC = IA
         ELSEIF (VA(IA) .LT. VD) THEN
           VD = VA(IA)
           ID = IA
         ENDIF
      ENDDO
      RETURN
      END
```

4.27 MAXIMUM AND MINIMUM MAGNITUDE VALUE IN VECTOR (MXMNMG)

Format

MXMNMG(VA,INCA,VC,IC,VD,ID,NN) Ident (R:I) : 031:177515B

Explanation

Scan a vector for its element with minimum absolute value and its element with maximum absolute value, and return these values together with the corresponding indices.

Parameters

VA : Name of input vector VA.
INCA : VA index increment.
VC : Name of output scalar VC, containing max. absolute value.
IC : Index in VA for max. absolute value.
VD : Name of output scalar VC, containing min. absolute value.
ID : Index in VA for min. absolute value.
NN : Element count.

Listing

```
      SUBROUTINE MXMNMG(VA,INCA,VC,IC,VD,ID,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1)
      IA = 1
      VC = ABS(VA(IA))
      VD = ABS(VA(IA))
      IC = IA
      ID = IA
      DO FOR M = 2,NN
         IA = IA + INCA
         VAABS = ABS(VA(IA))
         IF (VAABS .GT. VC) THEN
           VC = VAABS
           IC = IA
          ELSEIF (VAABS .LT. VD) THEN
           VD = VAABS
           ID = IA
         ENDIF
      ENDDO
      RETURN
      END
```

4.28 VECTOR SCALAR MULTIPLY AND ADD (VSMA)

Format

VSMA(VA,INCA,SC,VB,INCB,VC,INCC,NN) Ident (R:I) : 053:177515B

Explanation

Add the corresponding elements from two vectors, where the elements of one of the vectors are multiplied with a scalar value.
VCn = VAn * sc + VBn, where 'sc' denotes the scalar, and 'n' denotes the element index.

Parameters

VA : Name of input vector VA.
INCA : VA index increment.
SC : Scalar value.
VB : Name of input vector VB.
INCB : VB index increment.
VC : Name of output vector VC.
INCC : VC index increment.
NN : Element count.

Listing

```
      SUBROUTINE VSMA(VA,INCA,SC,VB,INCB,VC,INCC,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1), VB(1), VC(1)
      IA = 1
      IB = 1
      IC = 1
      DO FOR I = 1,NN
         VC(IC) = VA(IA)*SC + VB(IB)
         IA = IA + INCA
         IB = IB + INCB
         IC = IC + INCC
      ENDDO
      RETURN
      END
```

4.29 VECTOR SCALAR ADD (VSADD)

Format

VSADD(VA,INCA,B,VC,INCC,NN) Ident (R:I) : 032:177515B

Explanation

Add the elements of a vector together with a scalar value.
VCn = VAn + b, where 'b' denotes the scalar, and 'n' denotes the element index.

Parameters

VA : Name of input vector VA.
INCA : VA index increment.
B : Scalar B.
VC : Name of output vector VC.
INCC : VC index increment.
NN : Element count.

Listing

```
      SUBROUTINE VSADD(VA,INCA,B,VC,INCC,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1), VB(1), VC(1)
      IA = 1
      IC = 1
      DO FOR M = 1,NN
         VC(IC) = B + VA(IA)
         IA = IA + INCA
         IC = IC + INCC
      ENDDO
      RETURN
      END
```

4.30 VECTOR SCALAR MULTIPLY (VSMUL)

Format

VSMUL(VA,INCA,B,VC,INCC,NN) Ident (R:I) : 033:177515B

Explanation

Multiply the elements of a vector with a scalar value.
VCn = VAn * b, where 'b' denotes the scalar, and 'n' denotes the element index.

Parameters

VA : Name of input vector VA.
INCA : VA index increment.
B : Scalar B.
VC : Name of output vector VC.
INCC : VC index increment.
NN : Element count.

Listing

```
      SUBROUTINE VSMUL(VA,INCA,B,VC,INCC,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1), VB(1), VC(1)
      IA = 1
      IC = 1
      DO FOR M = 1,NN
         VC(IC) = B * VA(IA)
         IA = IA + INCA
         IC = IC + INCC
      ENDDO
      RETURN
      END
```

4.31 VECTOR SCALAR DIVIDE (VDIVS)

Format

VDIVS(VA,INCA,B,VC,INCC,NN) Ident (R:I) : 034:177515B

Explanation

Form a vector from a scalar value divided with the elements of another vector. VCn = b/VAn, 'b' denotes the scalar and 'n' denotes the element index.

Parameters

VA : Name of input vector VA.
INCA : VA index increment.
B : Scalar B.
VC : Name of output vector VC.
INCC : VC index increment.
NN : Element count.

Listing

```
      SUBROUTINE VDIVS(VA,INCA,B,VC,INCC,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1), VB(1), VC(1)
      IA = 1
      IC = 1
      DO FOR M = 1,NN
         VC(IC) = B / VA(IA)
         IA = IA + INCA
         IC = IC + INCC
      ENDDO
      RETURN
      END
```

4.32 DOT PRODUCT (DOTPR)

Format

DOTPR(VA,INCA,VB,INCB,VC,NN) Ident (R:I) : 035:177515B

Explanation

Add the product of the corresponding elements of two vectors. This function corresponds to the mathematical dot product, also called scalar product, of two vectors.
VC = $VA_1 * VB_1 + VA_2 * VB_2$ + ... + VAnn * VBnn , 'nn' is the element count.

Parameters

VA : Name of input vector VA.
INCA : VA index increment.
VB : Name of input vector VB.
INCB : VB index increment.
VC : Name of output scalar VC.
NN : Element count.

Listing

```
      SUBROUTINE DOTPR(VA,INCA,VB,INCB,VC,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1),VB(1)
      IA = 1
      IB = 1
      SUM = 0.0
      DO FOR M = 1,NN
         SUM = SUM + VA(IA)*VB(IB)
         IA = IA + INCA
         IB = IB + INCB
      ENDDO
      VC = SUM
      RETURN
      END
```

4.33 VECTOR CLEAR (VCLR)

Format

VCLR(VC,INCC,NN) Ident (R:I) : 036:177515B

Explanation

Set the elements of a vector to all zeros.

Parameters

VC : Name of output vector VC.
INCC : VC index increment.
NN : Element count.

Listing

```
      SUBROUTINE VCLR(VC,INCC,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VC(1)
      IC = 1
      DO FOR M = 1,NN
         VC(IC) = 0.0
         IC = IC + INCC
      ENDDO
      RETURN
      END
```

4.34 COMPLEX VECTOR MULTIPLY (CVMUL)

Format

CVMUL(VA,INCA,VB,INCB,VC,INCC,NN,NF) Ident (R:I) : 040:177515B

Explanation

Multiply two complex vectors. This function corresponds to mathematical multiplication of complex numbers. An own flag selects whether the result should be conjugated or not. VA = VAr + VAi, VB = VBr + VBi.

If the conjugate flag $\geq$ 0 then:

VC = (VAr * VBr - VAi*VBi)r + (VAr*VBi + VAi*VBr)i, else:

VC = (VAr * VBr - VAi*VBi)r - (VAr*VBi + VAi*VBr)i.

'r' and 'i' denotes real and imaginary elements.

Parameters

VA : Name of input vector VA.
INCA : VA index increment.
VB : Name of input vector VB.
INCB : VB index increment.
VC : Name of output vector VC.
INCC : VC index increment.
NN : Element count.
NF : Conjugate flag.

NF = +1 : Normal complex multiply.
NF = -1 : Multiply with conjugate of VA.

Listing

```
      SUBROUTINE CVMUL(VA,INCA,VB,INCB,VC,INCC,NN,NF)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMPLEX*16 VA(1),VB(1),VC(1)
      IA = 1
      IB = 1
      IC = 1
      DO FOR M = 1,NN
         ARE = VA(IA)
         BRE = VB(IB)
         AIM = DIMAG(VA(IA))
         BIM = DIMAG(VB(IB))
         IF (NF .LT. 0) AIM = -AIM
         CRE = ARE*BRE - AIM*BIM
         CIM = ARE*BIM + AIM*BRE
         VC(IC) = CMPLX(CRE,CIM)
         IA = IA + INCA
         IB = IB + INCB
         IC = IC + INCC
      ENDDO
      RETURN
      END
```

4.35 VECTOR TAPER (VTAPER)

Format

VTAPER(VA,VC,NN,IFLAG) Ident (R:I) : 041:177515B

Explanation

Multiply each element of a vector with an increasing or decreasing factor. An own flag selects either the decreasing or the increasing factor. The factor is a function of the element count.

If flag > 0 then:

$VC_1 = VA_1 * (1/nn)$, $VC_2 = VA_2 * (2/nn)$ .. VCnn = VAnn * (1).

So the general element equation is: VCn = VAn * (n/nn).

If flag $\leq$ 0 then:

$VC_1 = VA_1 * (1 - 1/nn)$, $VC_2 = VA_2 * (1 - 2/nn)$ .. VCnn = VAnn * (0).

So the general element equation is: VCn = VAn * (1 - n/nn).

'nn' denotes the element count and 'n' the element index.

Parameters

VA : Name of input vector VA.
VC : Name of output vector VC.
NN : Element count.
IFLAG : Flag.

Listing

```
      SUBROUTINE VTAPER(VA,VC,NN,IFLAG)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1),VC(1)
      ONE = 1.0
      IF (IFLAG .LE. 0) GO TO 10
      DMULT = ONE/NN
      DMINC = DMULT
      GO TO 20
 10   CONTINUE
      DMULT = (NN-1)*ONE/NN
      DMINC = -ONE/NN
 20   CONTINUE
      DO 30 I = 1,NN
         VC(I) = VA(I)*DMULT
         DMULT = DMULT + DMINC
 30   CONTINUE
      RETURN
      END
```

4.36 VECTOR RAMP FUNCTION (VRAMP)

Format

VRAMP(SCALAR,SCINC,VC,INCC,NN) Ident (R:I) : 043:177515B

Explanation

Form a vector as a ramp function with a start value and a slope as input parameters.

VC_1 = sc + scinc, VC_2 = sc + 2 * scinc .. VCnn = sc + nn * scinc.

So the general element expression is: VCn = sc + n * scinc.

'nn' denotes the element count, 'n' the element index, 'sc' start value, and 'scinc' slope.

Parameters

SCALAR : Scalar for start value.
SCINC : Scalar for increment.
VC : Name of output vector VC.
INCC : VC index increment.
NN : Element count.

Listing

```
      SUBROUTINE VRAMP(SCALAR,SCINC,VC,INCC,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VC(1)
      IC =1
      SC = SCALAR
      DO FOR I = 1,NN
          VC(IC) = SC
          IC = IC + INCC
          I    SC = SC + SCINC
      ENDDO
      RETURN
      END
```

4.37 VECTOR FIRST AND LAST NON-ZERO VALUE (VFLNZ)

Format

VFLNZ(VA,INDF,INDL,NN) Ident (R:I) : 051:177515B

Explanation

Find the indices of the first and last non-zero elements in a vector.

Parameters

VA : Name of input vector VA.
INDF : Name for first non-zero index.
INDL : Name for last non-zero index.
NN : Element count.

Listing

```
      SUBROUTINE VFLNZ(VA,INDF,INDL,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION VA(1)
      DO 100 I = 1,NN
         IF (VA(I) .NE. 0.0) THEN
           INDF = I
           GO TO 105
         ENDIF
 100  CONTINUE
      INDF = NN
      INDL = 1
      GO TO 205
 105  CONTINUE
      DO 200 I = NN,1,-1
         IF (VA(I) .NE. 0.0) THEN
           INDL = I
           GO TO 205
         ENDIF
 200  CONTINUE
 205  RETURN
      END
```

Index

********** SEND US YOUR COMMENTS!!! **********

Are you frustrated because of unclear information in this manual? Do you have trouble finding things? Why don't you join the Reader's Club and send us a note? You will receive a membership card — and an answer to your comments.

Please let us know if you

- find errors
- cannot understand information
- cannot find information
- find needless information

Do you think we could improve the manual by rearranging the contents? You could also tell us if you like the manual!

********** HELP YOURSELF BY HELPING US!! **********

Manual name: ND-500/2 Double Precision Array Processing Functions Manual number: ND-05.018.01

What problems do you have? (use extra pages if needed) _____

_____

_____

_____

_____

Do you have suggestions for improving this manual ? _____

_____

_____

_____

_____

_____

Your name: _____ Date: _____

Company: _____ Position: _____

Address: _____

_____

What are you using this manual for ? _____

_____

NOTE!
This form is primarily for documentation errors. Software and system errors should be reported on Customer System Reports.

Send to:
Norsk Data A.S
Documentation Department
P.O. Box 25, Bogerud
0621 Oslo 6, Norway

Norsk Data's answer will be found on reverse side

Answer from Norsk Data

Answered by Date

Norsk Data A.S
Documentation Department
P.O. Box 25, Bogerud
0621 Oslo6, Norway