CONVEX VECLIB

Quick Reference

Document No. 710-010930-001

Third Edition
August 1991

CONVEX Computer Corporation
Richardson, Texas USA

CONVEX VECLIB Quick Reference
Order No. DSW-134
Third Edition

Printed in the United States of America

Contents

1 Quick Reference Explanation 1

2 Basic Vector Operations 4

3 Matrix Operations 12

4 Linear Equations 17

5 Eigenvalues and Eigenvectors 21

6 Sparse Linear Equations 22

7 Sparse Eigenvalues/Eigenvectors 26

8 Fast Fourier Transforms 31

9 Correlation and Convolution 34

10 Linear Recurrences 35

11 Miscellaneous Routines 37

1 Quick Reference Explanation

The *VECLIB Quick Reference* provides a quick and efficient method for obtaining information on VECLIB. This reference lists subprogram names, purposes, and usage examples for all the subprograms in VECLIB.

This quick reference has the following format:

```
Subprogram Names
Function
    Usage Example
```

Consider the following example:

```
SPBFA/DPBFA/CPBFA/ZPBFA
Choleski Factorization of a Positive Definite
Band Matrix
    INTEGER*4 lda, n, m, ier
    REAL*4 a( lda, n)
    CALL SPBFA( a, lda, n, m, ier)
```

Please note:

- The quick reference lists subprograms in the order that they appear in the *CONVEX VECLIB User's Guide*.
- The quick reference usage examples include only one subroutine type.
- The corresponding subprogram types in the usage examples must be substituted when using other types.

VECLIB PRODUCT

The VECLIB product is a collection of FORTRAN-callable subprograms that provide mathematical software and computational kernels for application programs involving arrays.

The VECLIB product includes two libraries:

- VECLIB, which is the standard library designed for use with the default FORTRAN INTEGER, REAL, COMPLEX, and DOUBLE PRECISION

data types.

- VECLIB8, which is compatible with certain CONVEX FORTRAN compiler options that affect FORTRAN data types.

ACCESSING VECLIB

You can incorporate VECLIB and VECLIB8 libraries of compiled subprograms into your programs with a compiler. To incorporate a subprogram, include the appropriate declarations and CALL statements in the FORTRAN source program and specify the use of VECLIB or VECLIB8 as an object library at link time.

To access VECLIB subprograms use the "-l" option on the *fc* command line:

fc [*options*] *file* [*linker-options*]

where [*linker-options*] includes one of the strings:

-lveclib

or

-lveclib8

VECLIB NAMING CONVENTION

The VECLIB naming convention uses the first letter in the subprogram name to indicate different subroutine types. When calling a subprogram, change the declaration statements and first letter of the subroutine name according to this naming convention to correspond with your data types:

Naming Convention	VECLIB Data Type	VECLIB8 Data Type
I[name]	INTEGER*4	INTEGER*8
S[name]	REAL*4	REAL*8
D[name]	REAL*8	--
C[name]	COMPLEX*8	COMPLEX*16
Z[name]	COMPLEX*16	--

Using the previous example and table you can derive the usage for single precision, complex version of the Cholesky Factorization routine:

```
INTEGER*4 lda, n, m, ier
COMPLEX*8 a( lda, n)
CALL CPBFA( a, lda, m, n, ier)
```

VECLIB8

This quick reference does not include usage examples for VECLIB8. To use VECLIB8, deduce notation from the VECLIB usage examples:

- use

  INTEGER*4 (I),
  REAL*4 (S), and
  COMPLEX*8 (C) subprograms

- replace

  REAL*4 with REAL*8
  INTEGER*4 with INTEGER*8
  COMPLEX*8 with COMPLEX*16

In VECLIB8, the complex version of the Cholesky Factorization routine would be:

```
INTEGER*8 lda, n, m, ier
COMPLEX*16 a( lda, n)
CALL CPBFA( a, lda, n, m, ier)
```

2 Basic Vector Operations

This section lists subprograms for performing dense and sparse vector operations. These subprograms include BLAS and BLAS extensions.

ISAMAX/IDAMAX/IIAMAX/ICAMAX/IZAMAX
Index of Maximum of Magnitudes
```
INTEGER*4 i, ISAMAX, n, incx
REAL*4 x( lenx)
i=ISAMAX( n, x, incx)
```

ISAMIN/IDAMIN/IIAMIN/ICAMIN/IZAMIN
Index of Minimum of Magnitudes
```
INTEGER*4 i, ISAMIN, n, incx
REAL*4 x( lenx)
i=ISAMIN( n, x, incx)
```

ISCTxx/IDCTxx/IICTxx/ICCTxx/IZCTxx
(xx = EQ, GE, GT, LE, LT, or NE)
Count Selected Vector Elements
```
INTEGER*4 i, ISCTxx, n, incx
REAL*4 a, x( lenx)
i=ISCTxx( n, x, incx, a)
```

ISMAX/IDMAX/IIMAX
Index of Maximum Element of Vector
```
INTEGER*4 i, ISMAX, n, incx
REAL*4 x( lenx)
i=ISMAX( n, x, incx)
```

ISMIN/IDMIN/IIMIN
Index of Minimum Element of Vector
```
INTEGER*4 i, ISMIN, n, incx
REAL*4 x( lenx)
i=ISMIN( n, x, incx)
```

ISSVxx/IDSVxx/IISVxx/ICSVxx/IZSVxx
(xx = EQ, GE, GT, LE, LT, or NE)
Search Vector for Element
```
INTEGER*4 i, ISSVxx, n, incx
REAL*4 a, x( lenx)
i=ISSVxx( n, x, incx, a)
```

SAMAX/DAMAX/IAMAX
Maximum of Magnitudes
```
INTEGER*4 n, incx
REAL*4 s, SAMAX, x( lenx)
s=SAMAX( n, x, incx)
```

SCAMAX/DZAMAX
Maximum of Magnitudes
```
INTEGER*4 n, incx
REAL*4 s, SCAMAX
COMPLEX*8 x( lenx)
s=SCAMAX( n, x, incx)
```

SAMIN/DAMIN/IAMIN
Minimum of Magnitudes
```
INTEGER*4 n, incx
REAL*4 s, SAMIN, x( lenx)
s=SAMIN( n, x, incx)
```

SCAMIN/DZAMIN
Minimum of Magnitudes
```
INTEGER*4 n, incx
REAL*4 s, SCAMIN
COMPLEX*8 x( lenx)
s=SCAMIN( n, x, incx)
```

SASUM/DASUM/IASUM
Sum of Magnitudes
```
INTEGER*4 n, incx
REAL*4 s, SASUM, x( lenx)
s=SASUM( n, x, incx)
```

SCASUM/DZASUM
Sum of Magnitudes

```
INTEGER*4 n, incx
REAL*4 s, SCASUM
COMPLEX*8 x( lenx)
s=SCASUM( n, x, incx)
```

SAXPY/DAXPY/CAXPY/CAXPYC
ZAXPY/ZAXPYC
Elementary Vector Operation

```
INTEGER*4 n, incx, incy
REAL*4 a, x( lenx), y( leny)
CALL SAXPY( n, a, x, incx, y, incy)
```

SAXPYI/DAXPYI/CAXPYI/ZAXPYI
Sparse Elementary Vector Operation

```
INTEGER*4 m, indx( m)
REAL*4 a, x( m), y( n)
CALL SAXPYI( m, a, x, indx, y)
```

SCLIP/DCLIP/ICLIP
Two-Sided Vector Clip

```
INTEGER*4 n, incx, incy
REAL*4 a, b, x( lenx), y( leny)
CALL SCLIP( n, a, b, x, incx, y, incy)
```

SCLIPL/DCLIPL/ICLIPL
Left-Sided Vector Clip

```
INTEGER*4 n, incx, incy
REAL*4 a, x( lenx), y( leny)
CALL SCLIPL( n, a, x, incx, y, incy)
```

SCLIPR/DCLIPR/ICLIPR
Right-Sided Vector Clip

```
INTEGER*4 n, incx, incy
REAL*4 b, x( lenx), y( leny)
CALL SCLIPR( n, b, x, incx, y, incy)
```

SCOPY/DCOPY/ICOPY/CCOPY/CCOPYC
ZCOPY/ZCOPYC
Copy Vector

```
INTEGER*4 n, incx, incy
REAL*4 x( lenx), y( lenx)
CALL SCOPY( n, x, incx, y, incy)
```

SDOT/DDOT/CDOTC/CDOTU/ZDOTC/ZDOTU
Dot Product

```
INTEGER*4 n, incx, incy
REAL*4 s, SDOT, x( lenx), y( leny)
s=SDOT( n, x, incx, y, incy)
```

SDOTI/DDOTI/CDOTCI/CDOTUI
ZDOTCI/ZDOTUI
Sparse Dot Product

```
INTEGER*4 m, indx( m)
REAL*4 s, SOTI, x( m), y( n)
s=SDOTI( m, x, indx, y)
```

SFRAC/DFRAC
Extract Fractional Parts

```
INTEGER*4 n, incx, incy
REAL*4 x( lenx), y( leny)
CALL SFRAC
```

SGTHR/DGTHR/IGTHR/CGTHR/ZGTHR
Gather Sparse Vector

```
INTEGER*4 m, indx( m)
REAL*4 y( n), x( m)
CALL SGTHR(m, y, x, indx)
```

SGTHRZ/DGTHRZ/IGTHRZ/CGTHRZ/ZGTHRZ
Gather and Zero Sparse Vector

```
INTEGER *4 m, indx( m)
REAL*4 y( n), x( m)
CALL SGTHRZ( m, y, x, indx)
```

SLSTxx/DLSTxx/ILSTxx/CLSTxx/ZLSTxx
(xx = EQ, GE, GT, LE, LT, or NE)
List selected Vector Elements
INTEGER*4 n, incsx, nindx, indx(n)
REAL*4 a, x(lenx)
CALL SLSTxx(n, x, incx, a, nindx, indx)

SMAX/DMAX/IMAX
Maximum of Vector
INTEGER*4 n, incx
REAL*4 s, SMAX, x(lenx)
s=SMAX(n, x, incx)

SMIN/DMIN/IMIN
Minimum of Vector
INTEGER*4 n, incx
REAL*4 s, SMIN, x(lenx)
s=SMIN(n, x, incx)

SNRM2/DNRM2/SCNRM2/DZNRM2
Euclidean Norm
INTEGER*4 n, incx
REAL*4 s, SNRM2, x(lenx)
s=SNRM2(n, x, incx)

SNRSQ/DNRSQ
Euclidean Norm Squared
INTEGER*4 n, incx
REAL*4 s, SNRSQ, x(lenx)
s=SNRSQ(n, x, incx)

SCNRSQ/DZNRSQ
Euclidean Norm Squared
INTEGER*4 n, incx
REAL*4 s, SCNRSQ
COMPLEX*8 x(lenx)
s=SCNRSQ(n, x, incx)

SRAMP/DRAMP/IRAMP
Generate Linear Ramp
```
INTEGER*4 n, incx
REAL*4 a, h, x( lenx)
CALL SRAMP( n, a, h, x, incx)
```

SROT/DROT/CROT/CSROT/ZROT/ZDROT
Apply Givens Rotation
```
INTEGER*4 n, incx
REAL*4 x( lenx), y( leny), c, s
CALL SROT( n, x, incx, y, incy, c, s)
```

SROTG/DROTG/CROTG/ZROTG
Construct Givens Rotation
```
REAL*4 a, b, c, s
CALL SROTG( a, b, c, s)
```

SROTI/DROTI
Apply Sparse Givens Rotation
```
INTEGER*4 m, indx( m)
REAL*4 x( m), y( n), c, s
CALL SROTI( m, x, indx, y, c, s)
```

SROTM/DROTM
Apply Modified Givens Rotation
```
INTEGER*4 n, incx, incy
REAL*4 x( lenx), y( leny), param( 5)
CALL SROTM( n, x, incx, y, param)
```

SROTMG/DROTMG
Construct Modified Givens Rotation
```
REAL*4 d1, d2, x1, y1, param( 5)
CALL SROTG( d1, d2, x1, y1, param)
```

SRSCL/DRSCL/CRSCL/ZRSCL
Reciprocal Scale Vector
```
INTEGER*4 n, incx
REAL*4 a, x( lenx)
CALL SRSCL( n, a, x, incx)
```

CSRSCL/ZDRSCL
Reciprocal Scale Vector
```
INTEGER*8 n, incx
REAL*8 a
COMPLEX*16 x( lenx)
CALL CSRSCL( n, a, x, incx)
```

SSCAL/DSCAL/CSCAL/ZSCAL
Scale Vector
```
INTEGER*4 n, incx
REAL*4 a, x( lenx)
CALL SSCAL( n, a, x, incx)
```

CSSCAL/ZDSCAL
Scale Vector
```
INTEGER*4 n, incx
REAL*4 a
COMPLEX*8 x( lenx)
CALL CSSCAL( n, a, x, incx)
```

CSCALC/ZSCALC
Scale Vector
```
INTEGER*4 n, incx
COMPLEX*8 a, x( lenx)
CALL CSCALC( n, a, x, incx)
```

SSCTR/DSCTR/ISCTR/CSCTR/ZSCTR
Scatter Sparse Vector
```
INTEGER*4 m, indx( m)
REAL*4 x( m), y( n)
CALL SSCTR( m, indx, y)
```

SSUM/DSUM/ISUM/CSUM/ZSUM
Vector Sum
```
INTEGER*4 n, incx
REAL*4 s, SSUM, x( lenx)
s=SSUM( n , x, incx)
```

SSWAP/DSWAP/ISWAP/CSWAP/ZSWAP
Swap Two Vectors

```
INTEGER*4 n, incx, incy
REAL*4 x( lenx), y( leny)
CALL SSWAP( n, x, incx, y, incy)
```

SWDOT/DWDOT
Weighted Dot Product

```
INTEGER*4 n, incw, incx, incy
REAL*4 s, SWDOT, w( lenw), x( lenx), y( leny)
s=SWDOT( n, w, incw, x, incx, y, incy)
```

CWDOTC/CWDOTU/ZWDOTC/ZWDOTU
Weighted Dot Product

```
INTEGER*4 n, incw, incy
REAL*4 w( lenw)
COMPLEX*8 s, CWDOTC, x( lenx), y( leny)
s=CWDOTC( n, w, incw, x, incx, y, incy)
```

SZERO/DZERO/IZERO/CZERO/ZZERO
Clear Vector

```
INTEGER*4 n, incx
REAL*4 x( lenx)
CALL SZERO( n, x, incx)
```

3 Matrix Operations

This section lists the subprograms in the Level 2 (two-loop) BLAS and the Level 3 (three-loop) BLAS.

SGBMV/DGBMV/CGBMV/ZGBMV
Band Matrix-Vector Multiply
```
CHARACTER*(*) trans
INTEGER*4 m, n, ml, mu, lda, incx, incy
REAL*4 alpha, beta, a( lda, n), x( lenx), y( leny)
CALL SGBMV( trans, m, n, ml, mu, alpha, a, lda,
     x, incx, beta, y, incy)
```

SGEMM/DGEMM/CGEMM/ZGEMM
Matrix-Matrix Multiply
```
CHARACTER*(*) transa, transb
INTEGER*4 m, n, k, lda, ldb, ldc
REAL*4 alpha, beta, a( lda, *), b( ldb*), c( ldc, n)
CALL SGEMM( transa, transb, m, n, k, alpha, a, lda,
     b, ldb, beta, c, ldc)
```

SGEMV/DGEMV/CGEMV/ZGEMV
Matrix-Vector Multiply
```
CHARACTER*(*) trans
INTEGER*4 m, n, lda, incx, incy
REAL*4 alpha, beta, a( lda, n), x( lenx), y( leny)
CALL SGEMV( trans, m, n, alpha, a, lda, x, incx,
     beta, y, incy)
```

SGER/DGER/CGERC/CGERU/ZGERC/ZGERU
Rank-1 Update
```
INTEGER*4 m, n, lda, incx, incy
REAL*4 alpha, a( lda, n), x( lenx), y( leny)
CALL SGER( m, n, alpha, x, incx, y, incy, a, lda)
```

SSBMV/DSBMV/CHBMV/ZHBMV
Symmetric Band Matrix-Vector Multiply
```
CHARACTER*(*) uplo
INTEGER*4 n, m, lda, incx, incy
REAL*4 alpha, beta, a( lda, n), x( lenx), y( leny)
CALL SSBMV( uplo, n, m, alpha, a, lda, x, incx,
     beta, y, incy)
```

SSPMV/DSPMV/CHPMV/ZHPMV
Symmetric Packed Matrix-Vector Multiply

```
CHARACTER*(*) uplo
INTEGER*4 n, incx, incy
REAL*4 alpha, betaa, ap( lenap, x( lenx), y( leny)
CALL SSPMV( uplo, n, alpha, ap, x, incx, beta, y, incy)
```

SSPR/DSPR
Symmetric Packed Rank-1 Update

```
CHARACTER*(*) uplo
INTEGER*4 n, incx
REAL*4 alpha, ap( lenap), x( lenx)
CALL SSPR( uplo, n, alpha, x, incx, ap)
```

CHPR/ZHPR
Symmetric Packed Rank-1 Update

```
CHARACTER*(*) uplo
INTEGER*4 n, incx
REAL*4 alpha
COMPLEX*8 ap( lenap), x( lenx)
CALL CHPR( uplo, n, alpha, x, incx, ap)
```

SSPR2/DSPR2/CHPR2/ZHPR2
Symmetric Packed Rank-2 Update

```
CHARACTER*(*) uplo
INTEGER*4 n, incx, incy
REAL*4 alpha, ap( lenap), x( lenx), y( leny)
CALL SSPR2( uplo, n, alpha, x, incx, y, incy, ap)
```

SSYMM/DSYMM/CSYMM/ZSYMM/CHEMM/ZHEMM
Symmetric Matrix-Matrix Multiply

```
CHARACTER*(*) side, uplo
INTEGER*4 m, n, lda, ldb, ldc
REAL*4 alpha, beta, a( lda, *), b( ldb, *), c( ldc, *)
CALL SSYMM( side, uplo, m, n, alpha, a, lda, b,
      ldb, beta, c, ldc)
```

SSYMV/DSYMV/CHEMV/ZHEMV
Symmetric Matrix-Vector Multiply

```
CHARACTER*(*) uplo
INTEGER*4 n, lda, incx, incy
REAL*4 alpha, beta, a( lda, n), x( lenx), y( leny)
CALL SSYMV( uplo, n, alpha, a, lda, x, incx, beta,
      y, incy)
```

SSYR/DSYR
Symmetric Rank-1 Update

```
CHARACTER*(*) uplo
INTEGER*4 n, lda, incx
REAL*4 alpha, a( lda, n), x( lenx)
CALL SSYR( uplo, n, alpha, x, incx, a, lda)
```

CHER/ZHER
Hermitian Rank-1 Update

```
CHARACTER*(*) uplo
INTEGER*4 n, lda, incx
REAL*4 alpha
COMPLEX*8 alpha, a( lda, n), x( lenx)
CALL CHER( uplo, n, alpha, x, incx, a, lda)
```

SSYR2/DSYR2/CHER2/ZHER2
Symmetric Rank-2 Update

```
CHARACTER*(*) uplo
INTEGER*4 n, lda, incx, incy
REAL*4 alpha, a( lda, n), x( lenx), y( leny)
CALL SSYR2( uplo, n, alpha, x, incx, y, incy, a, lda)
```

SSYR2K/DSYR2K/CSYR2K/ZSYR2K
Symmetric Rank-2k Update

```
CHARACTER*(*) uplo, trans
INTEGER*4 n, k, lda, ldb, ldc
REAL*4 alpha, beta, a( lda, *), b( ldb, *), c( ldc, *)
CALL SSYR2K( uplo, trans, n, k, alpha, a, lda, b,
     ldb, beta, c, ldc)
```

CHER2K/ZHER2K
Hermitian Rank-2k Update

```
CHARACTER*(*) uplo, trans
INTEGER*4 n, k, lda, ldb, ldc
REAL*4 beta
COMPLEX*8 alpha, a( lda, *), b( ldb, *), c( ldc, *)
CALL CHER2K( uplo, trans, n, k, alpha, a, lda, b,
     ldb, beta, c, ldc)
```

SSYRK/DSYRK/CSYRK/ZSYRK
Symmetric Rank-k Update

```
CHARACTER*(*) uplo, trans
INTEGER*4 n, k, lda, ldc
REAL*4 alpha, beta, a( lda, *), c( ldc, *)
CALL SSYRK( uplo, trans, n, k, alpha, a, lda, beta,
     c, ldc)
```

CHERK/ZHERK
Hermitian Rank-k Update

```
CHARACTER*(*) uplo, trans
INTEGER*4 n, k, lda, ldc
REAL*8 alpha, beta
COMPLEX*8 a( lda, *), c( ldc, *)
CALL CHERK( uplo, trans, n, k, alpha, a, lda, beta,
     c, ldc)
```

STBMV/DTBMV/CTBMV/ZTBMV
Triangular Band Matrix-Vector Multiply

```
CHARACTER*(*) uplo, trans, diag
INTEGER*4 n, m, ldt, incx
REAL*4 t( ldt, n), x( lenx)
CALL STBMV( uplo, trans, diag, n, m, t, ldt, x, incx)
```

STBSV/DTBSV/CTBSV/ZTBSV
Solve Triangular Band System

```
CHARACTER*(*) uplo, trans, diag
INTEGER*4 n, m, ldt, incx
REAL*4 t( ldt, n), x( lenx)
CALL STBSV( uplo, trans, diag, n, m, t, ldt, x, incx)
```

STPMV/DTPMV/CTPMV/ZTPMV
Triangular Packed Matrix-Vector Multiply

```
CHARACTER*(*) uplo, trans, diag
INTEGER*4 n, incx
REAL*4 tp( lentp), x( lenx)
CALL STPMV( uplo, trans, diag, n, tp, x, inx)
```

STPSV/DTPSV/CTPSV/ZTPSV
Solve Packed Triangular System

```
CHARACTER*(*) uplo, trans, diag
INTEGER*4 n, incx
REAL*4 tp( lentp), x( lenx)
CALL STPSV( uplo, trans, diag, n, tp, x, incx)
```

STRMM/DTRMM/CTRMM/ZTRMM
Triangular Matrix-Matrix Multiply

```
CHARACTER*(*) side, uplo, transt, diag
INTEGER*4 m, n, ldt, ldb
REAL*4 alpha, t( ldt, *), b( ldb, *)
CALL STRMM( side, uplo, transt, diag, m, n, alpha,
      t, ldt, b, ldb)
```

STRMV/DTRMV/CTRMV/ZTRMV
Triangular Matrix-Vector Multiply

```
CHARACTER*(*) uplo, trans, diag
INTEGER*4 n, ldt, incx
REAL*4 t( ldt, n), x( lenx)
CALL STRMV( uplo, trans, diag, n, t, ldt, x, incx)
```

STRSM/DTRSM/CTRSM/ZTRSM
Solve Triangular Systems

```
CHARACTER*(*) side, uplo, transt, diag
INTEGER*4 m, n, ldt, ldb
REAL*4 alpha, t( ldt, *), b( ldb, *)
CALL STRSM( side, uplo, transt, diag, m, n, alpha,
      t, ldt, b, ldb)
```

STRSV/DTRSV/CTRSV/ZTRSV
Solve Triangular System

```
CHARACTER*(*) uplo, trans, diag
INTEGER*4 n, ldt, incx
REAL*4 t( ldt, n), x( lenx)
CALL STRSV( uplo, trans, diag, n, t, ldt, x, incx)
```

XERBLA
Error Handler

```
CHARACTER*6 name
INTEGER*4 iarg
CALL XERBLA( name, iarg)
```

4 Linear Equations

This section lists the LINPACK software library subprograms documented in the *CONVEX VECLIB User's Guide.*

SGBCO/DGBCO/CGBCO/ZGBCO
Factor a General Band Matrix and Estimate its Condition Number

```
INTEGER*4 lda, n, ml, mu, ipvt( n)
REAL*4 a( lda, n), rcond, work( n)
CALL SGBCO( a, lda, n, ml, mu, ipvt, rcond, work)
```

SGBDI/DGBDI/CGBDI/ZGBDI
Determinant of a General Band Matrix

```
INTEGER*4 lda, n, ml, mu, ipvt( n)
REAL*4 a( lda, n), det( 2)
CALL SGBDI( a, lda, n, ml, mu, ipvt, det)
```

SGBFA/DGBFA/CGBFA/ZGBFA
Factor a General Band Matrix

```
INTEGER*4 lda, n, ml, mu, ipvt( n), ier
REAL*4 a( lda, n)
CALL SGBFA( a, lda, n, ml, mu, ipvt, ier)
```

SGBSL/DGBSL/CGBSL/ZGBSL
Solve Linear Equations with a General Band Matrix

```
INTEGER*4 lda, n, ml, mu, ipvt( n), job
REAL*4 a( lda, n), b( n)
CALL SGBSL( a, lda, n, ml, mu, ipvt, b, job)
```

SGECO/DGECO/CGECO/ZGECO
Factor a General Matrix and Estimate its Condition Number

```
INTEGER*4 lda, n, ipvt( n)
REAL*4 a( lda, n), rcond, work( n)
CALL SGECO( a, lda, n, ipvt, rcond, work)
```

SGEDI/DGEDI/CGEDI/ZGEDI
Determinant and Inverse of a General Matrix
```
INTEGER*4 lda, n, ipvt( n), job
REAL*4 a( lda, n), det( 2), work( n)
CALL SGEDI( a, lda, n, ipvt, det, work, job)
```

SGEFA/DGEFA/CGEFA/ZGEFA
Factor a General Matrix
```
INTEGER*4 lda, n, ipvt( n), ier
REAL*4 a( lda, n)
CALL SGEFA( a, lda, n, ipvt, ier)
```

SGESL/DGESL/CGESL/ZGESL
Solve Linear Equations with a General Matrix
```
INTEGER*4 lda, n, ipvt( n), job
REAL*4 a( lda, n), b( n)
CALL SGESL( a, lda, n, ipvt, b, job)
```

SGTSL/DGTSL/CGTSL/ZGTSL
Solve Linear Equations with a Tridiagonal Matrix
```
INTEGER*4 n, ier
REAL*4 c( n), d( n), e( n), b( n)
CALL SGTSL( n, c, d, e, b, ier)
```

SGTSV/DGTSV/CGTSV/ZGTSV
Solve Linear Equations with a Diagonally-Dominant Tridiagonal Matrix
```
INTEGER*4 n, ier
REAL*4 c( n), d( n), e( n), b( n)
CALL SGTSV( n, c, d, e, b, ier)
```

SPBCO/DPBCO/CPBCO/ZPBCO
Factor Positive Definite Band Matrix, Estimate Condition Number
```
INTEGER*4 lda, n, m, ier
REAL*4 a( lda, n), rcond, work( n)
CALL SPBCO( a, lda, n, m, rcond, work, ier)
```

SPBDI/DPBDI/CPBDI/ZPBDI
Determinant or a Positive Definite Band matrix

```
INTEGER*4 lda, n, m
REAL*4 a( lda, n), det( 2)
CALL SPBDI( a, lda, n, m, det)
```

SPBFA/DPBFA/CPBFA/ZPBFA
Choleski Factorization of a Positive Definite Band Matrix

```
INTEGER*4 lda, n, m, ier
REAL*4 a( lda, n)
CALL SPBFA( a, lda, n, m, ier)
```

SPBSL/DPBSL/CPBSL/ZPBSL
Solve Linear Equations with a Positive Definite Band Matrix

```
INTEGER*4 lda, n, m
REAL*4 a( lda, n), b( n)
CALL SPBSL( a, lda, n, m, b)
```

SPOCO/DPOCO/CPOCO/ZPOCO
Factor a Positive Definite Matrix and Estimate its Condition Number

```
INTEGER*4 lda, n, ier
REAL*4 a( lda, n), rcond, work( n)
CALL SPOCO( a, lda, n, rcond, work, ier)
```

SPODI/DPODI/CPODI/ZPODI
Determinant and Inverse of a Positive Definite Matrix

```
INTEGER*4 lda, n, job
REAL*4 a( lda, n), det( 2)
CALL SPODI( a, lda, n, det, job)
```

SPOFA/DPOFA/CPOFA/ZPOFA
Choleski Factorization of a Positive Definite Matrix

```
INTEGER*4 lda, n, ier
REAL*4 a( lda, n)
CALL SPOFA( a, lda, n, ier)
```

SPOSL/DPOSL/CPOSL/ZPOSL
Solve Linear Equations with a Positive Definite Matrix

```
INTEGER*4 lda, n
REAL*4 a( lda, n), b( n)
CALL SPOSL( a, lda, n, b)
```

SPTSL/DPTSL/CPTSL/ZPTSL
Solve Linear Equations with a Positive Definite Tridiagonal Matrix

```
INTEGER*4 n
REAL*4 d( n), e( n - 1), b( n)
CALL SPTSL( n, d, e, b)
```

5 Eigenvalues and Eigenvectors

This section lists the EISPACK library subprograms documented in the *CONVEX VECLIB User's Guide*. These subprograms compute the eigenvalues and eigenvectors of matrices.

RS
Eigenvalues and Eigenvectors of a Real Symmetric Tridiagonal Matrix

```
INTEGER*4 ldax, n, job, ier
REAL*8 a( ldax, n), w( n), x( ldax, n)
REAL*8 work1( n), work2( n)
CALL RS( ldax, n, a, w, job, x, work1, work2, ier)
```

TQL2
Eigenvalues and Eigenvectors of a Real Symmetric Tridiagonal Matrix

```
INTEGER*4 ldx, n, ier
REAL*8 d( n), e( n), x( ldx, n)
CALL TQL2( ldx, n, d, e, x, ier)
```

TQLRAT
Eigenvalues of a Real Symmetric Matrix

```
INTEGER*4 n, ier
REAL*8 d( n), e2( n)
CALL TQLRAT( n, d, e2, ier)
```

TRED1
Reduce Real Symmetric Matrix to Tridiagonal Form

```
INTEGER*4 lda, n
REAL*8 a( lda, n), d( n), e( n), e2( n)
CALL TRED1( lda, n, a, d, e, e2)
```

TRED2
Reduce Real Symmetric Matrix to Tridiagonal Form

```
INTEGER*4 ldax, n
REAL*8 a( ldax, n), d( n), e( n), x( ldax, n)
CALL TRED2( ldax, n, a, d, e, x)
```

6 Sparse Linear Equations

This section lists the VECLIB sparse linear equation subprograms.

DSLEFS
One-Call Usage

```
INTEGER*4 neqns, maxzer, msglvl, output
INTEGER*4 colstr( neqns+1), rowind( nnzero)
INTEGER*4 nrhs, ldrhs, inrtia( 3), ier
REAL*8 pvttol, value( nnzero), rhs( ldrhs, nrhs)
REAL*8 cond, global( 150)
CALL DSLEFS( neqns, maxzer, pvttol, msglvl,
     output, colstr, rowind, value, nrhs, rhs, ldrhs,
     cond, inrtia, global, ier)
```

DSLEIN
Initialize Sparse Linear Equations

```
INTEGER*4 neqns, msglvl, output, ier
REAL*8 global( 150)
CALL DSLEIN( neqns, msglvl, output, global, ier)
```

DSLEI1
Matrix Structure Input by Single Entry

```
INTEGER*4 irow, jcol, ier
REAL*8 global( 150)
CALL DSLEI1( irow, jcol, global, ier)
```

DSLEIC
Matrix Structure Input by Column

```
INTEGER*4 jcol, nzcol, jrowin( nzcol), ier
REAL*8 global( 150)
CALL DSLEIC( jcol, nzcol, jrowin, global, ier)
```

DSLEIE
Matrix Structure Input by Finite Element

```
INTEGER*4 nnode, nodlst( nnode), ier
REAL*8 global( 150)
CALL DSLEIE( nnode, nodlst, global, ier)
```

DSLEIM
Matrix Structure Input by Matrix
```
INTEGER*4 neqns, nnzero, colstr( neqns+1)
INTEGER*4 rowind( nnzero), ier
REAL*8 global( 150)
CALL DSLEIM ( colstr, rowind, global, ier)
```

DSLEIF
End of Matrix Structure Input
```
INTEGER*4 ier
REAL*8 global( 150)
CALL DSLEIF( global, ier)
```

DSLEOR
Reordering and Symbolic Ractorization
```
INTEGER*4 maxzer, ier
REAL*8 global( 150)
CALL DSLEOR( maxzer, global, ier)
```

DSLEV1
Matrix Value Input by Single Entry
```
INTEGER*4 irow, jcol, ier
REAL*8 value, global( 150)
CALL DSLEV1( irow, jcol, value, global, ier)
```

DSLEVC
Matrix Value Input by Column
```
INTEGER*4 jcol, nzcol, jrowin( nzcol), ier
REAL*8 values( nzcol), global( 150)
CALL DSLEVC( jcol, nzcol, jrowin, values, global, ier)
```

DSLEVE
Matrix Value Input by Finite Element
```
INTEGER*4 nnode, ldelmx, nodlst( nnode), ier
REAL*8 elmmtx( ldelmx, nnode), global( 150)
CALL DSLEVE( nnode, nodlst, elmmtx, ldelmx,
      global, ier)
```

DSLEVM
Matrix Value Input by Matrix
```
INTEGER*4 neqns, nnzero, colstr( neqns+1)
INTEGER*4 rowind( nnzero), ier
REAL*8 values( nnzero), global( 150)
CALL DSLEVM( colstr, rowind, values, global, ier)
```

DSLECO
Numeric Factorization and Condition Number Estimation
```
INTEGER*4 inrtia( 3), ier
REAL*8 pvttol, cond, global( 150)
CALL DSLECO( pvttol, cond, inrtia, global, ier)
```

DSLEFA
Numeric Factorization
```
INTEGER*4 inrtia( 3), ier
REAL*8 pvttol, global( 150)
CALL DSLEFA( pvttol, inrtia, global, ier)
```

DSLESL
Solve
```
INTEGER*4 nrhs, ldrhs, ier
REAL*8 rhs( ldrhs, nrhs), global( 150)
CALL DSLESL( nrhs, rhs, ldrhs, global, ier)
```

DSLEDA
Deallocate Working Storage
```
INTEGER*4 ier
REAL*8 global( 150)
CALL DSLEDA( global, ier)
```

DSLEOC
Output Control
```
INTEGER*4 msglvl, output
REAL*8 global( 150)
CALL DSLEOC( msglvl, output, global)
```

DSLEPS
Print Statistics
```
REAL*8 global( 150)
CALL DSLEPS( global)
```

DSLERS
Restore Problem State from a Savefile
```
INTEGER*4 svfile, ier
REAL*8 global( 150)
CALL DSLERS( svfile, global, ier)
```

DSLESR
Retrieve Runtime Statistics
```
INTEGER*4 inuse, mxused, opcnts( 2)
REAL*8 global( 150), time( 6)
CALL DSLESR( global, inuse, mxused, time, opcnts)
```

DSLESV
Save Problem State to a Savefile
```
INTEGER*4 svfile, ier
REAL*8 global( 150)
CALL DSLESV( svfile, global, ier)
```

7 Sparse Eigenvalues/Eigenvectors

This section lists the VECLIB subprograms available for solving sparse symmetric and generalized symmetric eigenvalue problems.

DSEVE1
One Call Usage

```
CHARACTER*(*) bmxtyp, which, pbtype
INTEGER*4 annzer, bnnzer, norder, msglvl, output
INTEGER*4 acolst( norder+1), arowin( annzer)
INTEGER*4 bcolst( norder+1), browin( bnnzer)
INTEGER*4 neigvl, nfound, ndiscd, ier, warnng
LOGICAL*4 lfinit, rfinit
REAL*8 avalue( annzer), bvalue( bnnzer), lftend
REAL*8 rhtend, center, global( 150)
CALL DSEVE1( norder, msglvl, output, acolst, arowin,
     avalue, bmxtyp, bcolst, browin, bvalue, neigvl,
     which, pbtype, lfinit, lftend, rfinit, rhtend,
     center, nfound, ndiscd, global, ier, warnng)
```

DSEVIN
Initialize Sparse Eigenvalues/Eigenvectors

```
CHARACTER*(*) bmxtyp
INTEGER*4 norder, msglvl, output, ier
REAL*8 global( 150)
CALL DSEVIN( norder, bmxtyp, msglvl, output,
  global, ier)
```

DSEVI1
Matrix Structure Input by Single Entry

```
CHARACTER*(*) matrix
INTEGER*4 irow, jcol, ier
REAL*8 global( 150)
CALL DSEVI1( matrix, irow, jcol, global, ier)
```

DSEVIC
Matrix Structure Input by Column

```
CHARACTER*(*) matrix
INTEGER*4 jcol, nzcol, jrowin( nzcol), ier
REAL*8 global( 150)
CALL DSEVIC( matrix, jcol, nzcol, jrowin, global, ier)
```

DSEVIE
Matrix Structure Input by Finite Element
```
CHARACTER*(*) matrix
INTEGER*4 nnode, nodlst( nnode), ier
REAL*8 global( 150)
CALL DSEVIE( matrix, nnode, nodlst, global, ier)
```

DSEVIM
Matrix Structure Input by Matrix
```
CHARACTER*(*) matrix
INTEGER*4 norder, nnzero, colstr( norder+1)
INTEGER*4 rowind( nnzero), ier
REAL*8 global( 150)
CALL DSEVIM( matrix, colstr, rowind, global, ier)
```

DSEVIF
End of Matrix Structure Input
```
INTEGER*4 ier
REAL*8 global( 150)
CALL DSEVIF( global, ier)
```

DSEVOR
Reordering and Symbolic Factorization
```
INTEGER*4 ier
REAL*8 global( 150)
CALL DSEVOR( global, ier)
```

DSEVV1
Matrix Value Input by Single Entry
```
CHARACTER*(*) matrix
INTEGER*4 irow, jcol, ier
REAL*8 value, global( 150)
CALL DSEVV1( matrix, irow, jcol, value, global, ier)
```

DSEVVC
Matrix Value Input by Column
```
CHARACTER*(*) matrix
INTEGER*4 jcol, nzcol, jrowin( nzcol), ier
REAL*8 values( nzcol), global( 150)
CALL DSEVVC( matrix, jcol, nzcol, jrowin, values,
      global, ier)
```

DSEVVD
Matrix Value Input to Main Diagonal
```
CHARACTER*(*) matrix
INTEGER*4 norder, ier
REAL*8 values( norder), global( 150)
CALL DSEVVD( matrix, values, global, ier)
```

DSEVVE
Matrix Value Input by Finite Element
```
CHARACTER*(*) matrix
INTEGER*4 nnode, ldelmx, nodlst( nnode), ier
REAL*8 elmmtx( ldelmx, nnode), global( 150)
CALL DSEVVE( matrix, nnode, nodlst, elmmtx,
      ldelmx, global, ier)
```

DSEVVM
Matrix Value Input by Matrix
```
CHARACTER*(*) matrix
INTEGER*4 norder, nnzero, colstr( norder+1)
INTEGER*4 rowind( nnzero), ier
REAL*8 values( nnzero), global( 150)
CALL DSEVVM( matrix, colstr, rowind, values,
      global, ier)
```

DSEVES
Eigen Extraction
```
CHARACTER*(*) which, pbtype
INTEGER*4 neigvl, nfound, ndiscd, ier, warnng
LOGICAL*4 lfinit, rfinit
REAL*8 lftend, rhtend, center, global( 150)
CALL DSEVES( neigvl, which, pbtype, lfinit, lftend,
      rfinit, rhtend, center, nfound, ndiscd, global, ier,
      warnng)
```

DSEVEX
Eigen Extraction
```
CHARACTER*(*) which, pbtype
INTEGER*4 neigvl, mxbksz, usrsvc, svcfil
INTEGER*4 nfound, ndiscd, ier, warnng
LOGICAL*4 lfinit, rfinit
REAL*8 lftend, rhtend, center, tolact, shfscl
REAL*8 global( 150)
CALL DSEVEX( neigvl, which, pbtype, lfinit, lftend
  rfinit, rhtend, center, mxbksz, tolact, shfscl
  usrsvc, svcfil, nfound, ndiscd, global, ier, warnng)
```

DSEVRC
Return Eigenvalue/Eigenvector Results
```
INTEGER*4 levalu, fstval, lstval, ldvevct, ier
REAL*8 evalue( levalu), evectr( ldevct, levalu)
REAL*8 global( 150)
CALL DSEVRC( levalu, evalue, fstval, lstval, evectr,
  ldevct, global, ier)
```

DSEVRL
Return Eigenvalue Results
```
INTEGER*4 levalu, fstval, lstval, ier
REAL*8 evalue( levalu), global( 150)
CALL DSEVRL( levalu, evalue, fstval, lstval, global,
  ier)
```

DSEVCK
Check Accuracy of Results
```
INTEGER*4 ier
LOGICAL*4 nodscd, ortwrn
REAL*8 discrp, global( 150)
CALL DSEVCK( nodscd, ortwrn, discrp, global, ier)
```

DSEVDA
Deallocate Working Storage
```
INTEGER*4 ier
REAL*8 global( 150)
CALL DSEVDA( global, ier)
```

DSEVOC
Output Control
```
INTEGER*4 msglvl, output
REAL*8 global( 150)
CALL DSEVOC( msglvl, output, global)
```

DSEVRS
Restore Problem State from a Savefile
```
INTEGER*4 svfile, ier
REAL*8 global( 150)
CALL DSEVRS( svfile, global, ier)
```

DSEVSV
Save Problem State to a Savefile

```
INTEGER*4 svfile, ier
REAL*8 global( 150)
CALL DSEVSV( svfile, global, ier)
```

8 Fast Fourier Transforms

This section lists the VECLIB fast Fourier transform (FFT) subprograms.

C1DFFT/Z1DFFT
One-Dimensional FFT

```
INTEGER*4 l, iopt, ier
COMPLEX*8 z( l)
REAL*4 work( lwork)
CALL C1DFFT ( z, l, work, iopt, ier )
```

D1DFFT/S1DFFT
One-Dimensional FFT

```
INTEGER*4 l, iopt, ier
REAL*4 x( l), y( l), work( lwork)
CALL S1DFFT ( x, y, l, work, iopt, ier)
```

C2DFFT/Z2DFFT
Two-Dimensional FFT

```
INTEGER*4 l1, l2, ldz, iopt, ier
COMPLEX*8 z( ldz, l2)
CALL C2DFFT( z, l1, l2, ldz, iopt, ier)
```

D2DFFT/S2DFFT
Two-Dimensional FFT

```
NTEGER*4 l1, l2, ldxy, iopt, ier
REAL*4 x( ldxy, l2), y( ldxy, l2)
CALL S2DFFT ( x, y, l1, l2, ldxy, iopt, ier)
```

C3DFFT/Z3DFFT
Three-Dimensional FFT

```
INTEGER*4 l1, l2, l3, ldz, mdz, iopt, ier
COMPLEX*8 z( ldz, mdz, l3)
CALL C3DFFT( z, l1, l2, l3, ldz, mdz, iopt, ier)
```

S3DFFT/D3DFFT
Three-Dimensional FFT

```
INTEGER*4 l1, l2, l3, ldxy, mdxy, iopt, ier
REAL*4 x( ldxy, mdxy, l2), y( ldxy, mdxy, l2)
CALL S3DFFT ( x, y, l1, l2, l3, ldxy, mdxy, iopt, ier)
```

CFFTS/ZFFTS
Simultaneous One-Dimensional FFT

```
INTEGER*4 l, incl, n, incn, iopt, ier
COMPLEX*8 z( lenz)
CALL CFFTS( z, l, incl, n, incn, iopt, ier)
```

SFFTS/DFFTS
Simultaneous One-Dimensional FFT

```
INTEGER*4 l, incl, n, incn, iopt, ier
REAL*4 x( lenxy), y( lenxy)
CALL SFFTS( x, y, l, incl, n, incn, iopt, ier)
```

CRC1FT/ZRC1FT
Real-to-Complex One-Dimensional FFT

```
INTEGER* l, iopt, ier
COMPLEX*8 z( l)
REAL*4 work( lwork)
CALL CRC1FT( z, l, work, iopt, ier)
```

SRC1FT/DRC1FT
Real-to-Complex One-Dimensional FFT

```
INTEGER*4 l, iopt, ier
REAL*4 x( l+2), work( lwork)
CALL SRC1FT( x, l, work, iopt, ier)
```

CRC2FT/ZRC2FT
Real-to-Complex Two-Dimensional FFT

```
INTEGER*4 l1, l2, ldz, iopt, ier
COMPLEX*8 z( ldz, l2)
CALL CRC2FT( z, l1, l2, ldz, iopt, ier)
```

SRC2FT/DRC2FT
Real-to-Complex Two-Dimensional FFT

```
INTEGER*4 l1, l2, ldx, iopt, ier
REAL*4 x( ldx, l2)
CALL SRC2FT( x, l1, l2, ldx, iopt, ier)
```

CRC3FT/ZRC3FT
Real-to-Complex Three-Dimensional FFT

```
INTEGER*4 l1, l2, l3, ldz, mdz, iopt, ier
COMPLEX*8 z( ldz, mdz, l3)
CALL CRC3FT( z, l1, l2, l3, ldz, mdz, iopt, ier)
```

DRC3FT/SRC3FT
Real-to-Complex Three-Dimensional FFT

```
INTEGER*4 l1, l2, l3, ldz, mdz, iopt, ier
COMPLEX*8 z( ldz, mdz, l3)
CALL CRC3FT( z, l1, l2, l3, ldz, mdz, iopt, ier)
```

CRCFTS/ZRCFTS
Simultaneous Real-to-Complex 1-D FFT

```
INTEGER*4 l, incl, n, incn, iopt, ier
COMPLEX*8 z( lenz)
CALL CRCFTS( z, l, incl, n, incn, iopt, ier)
```

DRCFTS/SRCFTS
Simultaneous Real-to-Complex 1-D FFT

```
INTEGER*4 l, incl, n, incn, iopt, ier
REAL*4 x( lenx)
CALL SRCFTS( x, l, incl, n, incn, iopt, ier)
```

9 Correlation and Convolution

This section lists the VECLIB subprograms available for correlations, convolutions, and related operations, such as filtering by means of convolutions.

SCONV/DCONV
Correlation and Convolution

```
INTEGER*4 incx, incw, incy, m, n
REAL*4 x( lenx), w( lenw), y( leny)
CALL SCONV( x, incx, w, incw, y, incy, m, n)
```

10 Linear Recurrences

This section lists the VECLIB subprograms for a variety of linear recurrences.

SFLR1M/SFLR1P/DFLR1M/DFLR1P
Solve a First Order Linear Recurrence

```
INTEGER*4 n,inca,incx
REAL*4 a(lena),x(lenx)
CALL SFLR1M (n, a, inca, x, incx)
```

SFLR1C/DFLR1C
Solve a First Order Linear Recurrence with Constant Coefficient

```
INTEGER*4 n,incx
REAL*4 alpha,x(lenx)
CALL SFLR1C (n, alpha, x, incx)
```

SFLR2M/SFLR2P/DFLR2M/DFLR2P
Solve a First Order Linear Recurrence

```
INTEGER*4 n,inca,incx
REAL*4 a(lena),x(lenx),y(leny)
CALL SFLR2M (n, a, inca, x, incx, y, incy)
```

SFLR2C/DFLR2C
Solve a First Order Linear Recurrence with Constant Coefficient

```
INTEGER*4 n,incx,incy
REAL*4 alpha,x(lenx),y(leny)
CALL SFLR2C (n, alpha, x, incx, y, incy)
```

SFLRLM/SFLRLP/DFLRLM/DFLRLP
Solve for the Last Term of a First Order Linear Recurrence

```
INTEGER*4 n,inca,incx
REAL*4 yn,SFLRLM,a(lena),x(lenx)
yn = SFLRLM (n, a, inca, x, incx)
```

SPPROD/DPPROD
Compute the Vector of Partial Products of a Vector
```
INTEGER*4 n,incx,incy
REAL*4 x(lenx),x(leny)
CALL SPPROD (n, x, incx, y, incy)
```

SPSUM/DPSUM/IPSUM
Compute the Vector of Partial Sums of a Vector
```
INTEGER*4 n,incx,incy
REAL*8 x(lenx),y(leny)
CALL SPSUM (n, x, incx, y, incy)
```

SSLRL/DSLRL
Solve for the Last Term of a Second Order Linear Recurrence
```
INTEGER*4 n,inca,incb,incx
REAL*4 xn,SSLRL,a(lena),b(lenb),x(lenx)
xn = SSLRL (n, a, inca, b, incb, x, incx)
```

SSLR2/DSLR2
Solve a Second Order Linear Recurrence
```
INTEGER*4 n,inca,incb,incx
REAL*4 a(lena),b(lenb),x(lenx)
CALL SSLR2 (n, a, inca, b, incb, x, incx)
```

SSLR3/DSLR3
Solve a Second Order Linear Recurrence
```
INTEGER*4 n,inca,incb,incx
REAL*4 a(lena),b(lenb),x(lenx)
CALL SSLR3 (n, a, inca, b, incb, x, incx)
```

11 Miscellaneous Routines

This section lists VECLIB subprograms that perform a variety of operations.

CPUTIME
Measure CPU Time Used

To read the clock at the beginning of the code segment:

```
REAL*4 CPUTIME, time, tzero
tzero = CPUTIME( 0.0)
```

To read the clock at the end of the code segment:

```
time=CPUTIME( tzero)
```

DALLOC
Deallocate Nonvolatile Dynamic Memory

```
INTEGER*4 iptr, ier
CALL DALLOC( iptr, ier)
IF ( ier .LT. 0 ) THEN
  handle error
END IF
```

DYNAMIC
Allocate Dynamic Memory

```
CALL sub ( <nondynamic actual arguments>)
  .
  .
SUBROUTINE sub ( <nondynamic dummy args>,
                 <dynamic dummy args>)
INTEGER*4 ier, DYNAMIC, n1, l1, n2, l2, ..., nk, lk
array declarations for <dynamic dummy args>
ier = DYNAMIC ( n1, l1, n2, l2, ..., nk, lk)
IF ( ier .EQ. 0 ) THEN
    handle stack overflow
ENDIF
```

MALLOC
Allocate Dynamic Memory

```
INTEGER*4 MALLOC, iptr, l
iptr = MALLOC( l)
CALL sub (...,%VAL( iptr),...)
```

NALLOC
Allocate Nonvolatile Dynamic Memory

```
INTEGER*4 l, iptr, ier
CALL NALLOC( l, iptr, ier)
IF ( ier .LT. 0 ) THEN
  handle error
END IF
CALL sub(...,%VAL( iptr),...)
```

RALLOC
Reallocate Nonvolatile Dynamic Memory

```
INTEGER*4 l, iptr, ier
CALL RALLOC ( l, iptr, ier)
IF ( ier .LT. 0 ) THEN
  handle error
END IF
CALL sub(...,%VAL( iptr),...)
```

RAN/RANV
VAX-Compatible Random Numbers

Scalar version:

```
INTEGER*4 iseed
REAL*4 RAN, r
r = RAN( iseed)
```

Vector version:

```
INTEGER*4 iseed, n
REAL*4 v( n)
CALL RANV ( iseed, n, v)
```

SC2IBM
CONVEX to IBM Floating-point Conversion

```
INTEGER*4 n, incx, incy
REAL*4 x( lenx), y( leny)
CALL SC2IBM ( n, x, incx, y, incy)
```

SIBM2C
IBM Floating-point to CONVEX Conversion

```
INTEGER*4 n, incx, incy
REAL*4 x( lenx), y( leny)
CALL SIBM2C ( n, x, incx, y, incy)
```

SSORT/DSORT/ISORT
Sort Array

```
CHARACTER*(*) order
INTEGER*4 n, incx
REAL*4 x( lenx)
CALL SSORT ( order, n, x, incx)
```