CONVEX VECLIB

Quick Reference

Document No. 740-000252-200

First Edition
December 1989

CONVEX Computer Corporation
Richardson, Texas USA

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

Printed in the United States of America

Contents

1 Introduction

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

This quick reference has the following format:

Purpose
Subprogram Names
    Usage Example

Example:

Cholesky Factorization
SPBFA/DPBFA/CPBFA/ZPBFA

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

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 data type.
- The corresponding data types and subprogram names in the usage examples must be substituted when using other types.

THE 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 by the loader. 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 load time.

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

fc [*options*] *file* [*loader-options*]

where [*loader-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 data 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	--

Example:

Use the previous example and table to derive the 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

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 Level 1 (one-loop) Basic Linear Algebra Subprograms (BLAS), Sparse BLAS, and BLAS extensions.

Index of the Element of a Vector of Maximum Magnitude
ISAMAX, IDAMAX, IIAMAX, ICAMAX, IZAMAX

```
INTEGER*4 i,ISAMAX,n,incx
REAL*4 x(lenx)
i = ISAMAX (n, x, incx)
```

Index of the Element of a Vector of Minimum Magnitude
ISAMIN, IDAMIN, IIAMIN, ICAMIN, IZAMIN

```
INTEGER*4 i,ISAMIN,n,incx
REAL*4 x(lenx)
i = ISAMIN (n, x, incx)
```

Count the Number of Occurrences of Selected Elements of a Vector
ISCTxx, IDCTxx, IICTxx, ICCTxx, IZCTxx
(xx = EQ, GE, GT, LE, LT, or NE)

```
INTEGER*4 i,ISCTxx,n,incx
REAL*4 a,x(lenx)
i = ISCTxx (n, x, incx, a)
```

Index of the Maximum Element of a Vector
ISMAX, IDMAX, IIMAX

```
INTEGER*4 i,ISMAX,n,incx
REAL*4 x(lenx)
i = ISMAX (n, x, incx)
```

Index of the Minimum Element of a Vector
ISMIN, IDMIN, IIMIN

```
INTEGER*4 i,ISMIN,n,incx
REAL*4 x(lenx)
i = ISMIN (n, x, incx)
```

Search a Vector for a Specified Element
ISSVxx, IDSVxx, IISVxx, ICSVxx, IZSVxx
(xx = EQ, GE, GT, LE, LT, or NE)

```
INTEGER*4 i,ISSVxx,n,incx
REAL*4 a,x(lenx)
i = ISSVxx (n, x, incx, a)
```

Maximum of Magnitudes of the Elements of a Vector
SAMAX, DAMAX, IAMAX

```
INTEGER*4 n,incx
REAL*4 s,SAMAX,x(lenx)
s = SAMAX (n, x, incx)
```

Maximum of Magnitudes of the Elements of a Vector
SCAMAX, DZAMAX

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

Minimum of Magnitudes of the Elements of a Vector
SAMIN, DAMIN, IAMIN

```
INTEGER*4 n,incx
REAL*4 s,SAMIN,x(lenx)
s = SAMIN (n, x, incx)
```

Minimum of Magnitudes of the Elements of a Vector
SCAMIN, DZAMIN

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

Sum of Magnitudes of the Elements of a Vector
SASUM, DASUM, IASUM

```
INTEGER*4 n,incx
REAL*4 s,SASUM,x(lenx)
s = SASUM (n, x, incx)
```

Sum of Magnitudes of the Elements of a Vector
SCASUM, DZASUM

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

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

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

Sparse Elementary Vector Operation
SAXPYI, DAXPYI, CAXPYI, ZAXPYI

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

Two-Sided Vector Clip
SCLIP, DCLIP, ICLIP

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

Left-Sided Vector Clip
SCLIPL, DCLIPL, ICLIPL

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

Right-Sided Vector Clip
SCLIPR, DCLIPR, ICLIPR

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

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

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

Dot Product of Two Vectors
SDOT, DDOT, CDOTC, CDOTU, ZDOTC, ZDOTU

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

Sparse Dot Product of Two Vectors
SDOTI, DDOTI, CDOTCI, CDOTUI, ZDOTCI, ZDOTUI

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

Extract Fractional Parts of the Elements of a Vector
SFRAC, DFRAC

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

Gather a Sparse Vector
SGTHR, DGTHR, IGTHR, CGTHR, ZGTHR

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

Gather and Zero a Sparse Vector
SGTHRZ, DGTHRZ, IGTHRZ, CGTHRZ, ZGTHRZ

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

Build a List of Indices of Selected Vector Elements
SLSTxx, DLSTxx, ILSTxx, CLSTxx, ZLSTxx
(xx = EQ, GE, GT, LE, LT, or NE)

```
INTEGER*4 n,incx,nindx,indx(n)
REAL*4 a,x(lenx)
CALL SLSTxx (n, x, incx, a, nindx, indx)
```

Value of the Maximum Element of a Vector
SMAX, DMAX, IMAX

```
INTEGER*4 n,incx
REAL*4 s,SMAX,x(lenx)
s = SMAX (n, x, incx)
```

Value of the Minimum Element of a Vector
SMIN, DMIN, IMIN

```
INTEGER*4 n,incx
REAL*4 s,SMIN,x(lenx)
s = SMIN (n, x, incx)
```

Euclidean Norm of a Vector
SNRM2, DNRM2

```
INTEGER*4 n,incx
REAL*4 s,SNRM2,x(lenx)
s = SNRM2 (n, x, incx)
```

Euclidean Norm of a Vector
SCNRM2, DZNRM2

```
INTEGER*4 n,incx
REAL*4 s,SCNRM2
COMPLEX*8 x(lenx)
s = SCNRM2 (n, x, incx)
```

Square of the Euclidean Norm of a Vector
SNRSQ, DNRSQ

```
INTEGER*4 n,incx
REAL*4 s,SNRSQ,x(lenx)
s = SNRSQ (n, x, incx)
```

Square of the Euclidean Norm of a Vector
SCNRSQ, DZNRSQ

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

Generate a Linear Ramp
SRAMP, DRAMP, IRAMP

```
INTEGER*4 n,incx
REAL*4 a,h,x(lenx)
CALL SRAMP (n, a, h, x, incx)
```

Apply a Givens Rotation to Two Vectors
SROT, DROT

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

Apply a Givens Rotation to Two Vectors
CROT, ZROT

```
INTEGER*4 n,incx,incy
REAL*4 c
COMPLEX*8 x(lenx),y(leny),s
CALL CROT (n, x, incx, y, incy, c, s)
```

Apply a Givens Rotation to Two Vectors
CSROT, ZDROT

```
INTEGER*4 n,incx,incy
REAL*4 c,s
COMPLEX*8 x(lenx),y(leny)
CALL CSROT (n, x, incx, y, incy, c, s)
```

Construct a Givens Rotation
SROTG, DRTOG

```
REAL*4 a,b,c,s
CALL SROTG (a, b, c, s)
```

Construct a Givens Rotation
CROTG, ZROTG

```
REAL*4 c
COMPLEX*8 a,b,s
CALL CROTG (a, b, c, s)
```

Apply a Sparse Givens Rotation to Two Vectors
SROTI, DROTI

```
INTEGER*4 m,indx(m)
REAL*4 x(m),y(n),c,s
CALL SROTI (m, x, indx, y, c, s)
```

Apply a Modified Givens Rotation
SROTM, DROTM

```
INTEGER*4 n,incx,incy
REAL*4 x(lenx),y(leny),param(5)
CALL SROTM (n, x, incx, y, incy, param)
```

Construct a Modified Givens Rotation
SROTMG, DROTMG

```
REAL*4 d1,d2,x1,y1,param(5)
CALL SROTMG (d1, d2, x1, y1, param)
```

Scale a Vector
SSCAL, DSCAL, CSCAL, ZSCAL, CSCALC, ZSCALC

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

Scale a Vector
CSSCAL, ZDSCAL

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

Scatter a Sparse Vector
SSCTR, DSCTR, ISCTR, CSCTR, ZSCTR

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

Sum of the Elements of a Vector
SSUM, DSUM, ISUM, CSUM, ZSUM

```
INTEGER*4 n,incx
REAL*4 s,SSUM,x(lenx)
s = SSUM (n, x, incx)
```

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

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

Weighted Dot Product of Two Vectors
SWDOT, DWDOT

```
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)
```

Weighted Dot Product of Two Vectors
CWDOTC, CWDOTU, ZWDOTC, ZWDOTU

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

3 Basic Matrix Operations

This section lists subprograms for performing matrix operations. These subprograms include Level 2 (two-loop) BLAS and Level 3 (three-loop) BLAS.

General Matrix-Matrix Multiply
SGEMM, DGEMM, CGEMM, ZGEMM

```
CHARACTER*1 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)"
```

General Matrix-Vector Multiply
SGEMV, DGEMV, CGEMV, ZGEMV

```
CHARACTER*1 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)
```

General Rank-1 Update
SGER, DGER, CGERC, CGERU, ZGERC, ZGERU

```
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)
```

General Rank-2 Update
SGER2, DGER2

```
INTEGER*4 m,n,lda,incx,incy
REAL*4 alpha1,alpha2,a(lda,n),x1(lenx),x2(lenx),y1(leny),y2(leny)
CALL SGER2 (m,n,alpha1,alpha2,x1,x2,incx,y1,y2,incy,a,lda)
```

Symmetric or Hermitian Matrix-Matrix Multiply
SSYMM, DSYMM, CHEMM, CSYMM, CHEMM, ZSYMM

```
CHARACTER*1 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)
```

Symmetric or Hermitian Matrix-Vector Multiply
SSYMV, DSYMV, CHEMV, ZHEMV

```
CHARACTER*1 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)
```

Symmetric or Hermitian Rank-1 Update
SSYR, DSYR

```
CHARACTER*1 uplo
INTEGER*4 n,lda,incx
REAL*4 alpha,a(lda,n),x(lenx)
CALL SSYR (uplo, n, alpha, x, incx, a, lda)
```

Symmetric or Hermitian Rank-1 Update
CHER, ZHER

```
CHARACTER*1 uplo
INTEGER*4 n,lda,incx
REAL*4 alpha
COMPLEX*8 a(lda,n),x(lenx)
CALL CHER (uplo, n, alpha, x, incx, a, lda)
```

Symmetric or Hermitian Rank-2 Update
SSYR2, DSYR2, CHER2, ZHER2

```
CHARACTER*1 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)
```

Symmetric or Hermitian Rank-k Update
SSYRK, DSYRK, CSYRK, ZSYRK

```
CHARACTER*1 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)
```

Symmetric or Hermitian Rank-k Update
CHERK, ZHERK

```
CHARACTER*1 uplo,trans
INTEGER*4 n,k,lda,ldc
REAL*4 alpha,beta
COMPLEX*8 a(lda,*),c(ldc,*)
CALL CHERK (uplo, trans, n, k, alpha, a, lda, beta, c, ldc)
```

Symmetric or Hermitian Rank-2k Update
SSYR2K, DSYR2K, CSYR2K, ZSYR2K

```
CHARACTER*1 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)
```

Symmetric or Hermitian Rank-2k Update
CHER2K, ZHER2K

```
CHARACTER*1 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)
```

Solve Triangular Band System
STBSV, DTBSV, CTBSV, ZTBSV

```
CHARACTER*1 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)
```

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

```
CHARACTER*1 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)
```

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

```
CHARACTER*1 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)
```

Solve Simultaneous Triangular Systems
STRSM, DTRSM, CTRSM, ZTRSM

```
CHARACTER*1 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)
```

Solve One Triangular System
STRSV, DTRSV, CTRSV, ZTRSV

```
CHARACTER*1 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)
```

BLAS Error Handler
XERBLA

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

4 Linear Equations

This section lists the optimized LINPACK subprograms included in VECLIB.

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

```
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)
```

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

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

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

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

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

```
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)
```

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

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

Determinant and Inverse of a General Matrix
SGEDI, DGEDI, CGEDI, ZGEDI

```
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)
```

Factor a General Matrix
SGEFA, DGEFA, CGEFA, ZGEFA

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

Solve Linear Equations with a General Matrix
SGESL, DGESL, CGESL, ZGESL

```
INTEGER*4 lda,n,ipvt(n),job
REAL*4 a(lda,n),b(n)
CALL SGESL (a, lda, n, ipvt, b, job)
```

Factor Positive Definite Band Matrix, Estimate Condition Number
SPBCO, DPBCO, CPBCO, ZPBCO

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

Determinant of a Positive Definite Band Matrix
SPBDI, DPBDI, CPBDI, ZPBDI

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

Cholesky Factorization of a Positive Definite Band Matrix
SPBFA, DPBFA, CPBFA, ZPBFA

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

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

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

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

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

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

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

Cholesky Factorization of a Positive Definite Matrix
SPOFA, DPOFA, CPOFA, ZPOFA

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

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

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

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

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

Solve Linear Equations with a Tridiagonal Matrix
STRI, DTRI

```
INTEGER*4 n,ier
REAL*4 c(n–1),d(n),e(n–1),b(n)
CALL STRI (n, c, d, e, b, ier)
```

5 Eigenvalues and Eigenvectors

This section lists optimized EISPACK library subprograms included in VECLIB

Eigenvalues and Eigenvectors of a Real Symmetric Matrix
RS

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

Eigenvalues and Eigenvectors of a Real Symmetric Tridiagonal Matrix
TQL2

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

Eigenvalues of a Real Symmetric Tridiagonal Matrix
TQLRAT

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

Reduce a Real Symmetric Matrix to Real Symmetric Tridiagonal Form
TRED1

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

Reduce a Real Symmetric Matrix to Real Symmetric Tridiagonal Form
TRED2

```
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 subprograms that solve sparse symmetric systems of linear equations by direct methods.

One-Call Usage
DSLEFS

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

Initialize Sparse Linear Equations
DSLEIN

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

Matrix Structure Input by Single Entry
DSLEI1

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

Matrix Structure Input by Column
DSLEIC

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

Matrix Structure Input by Finite Element
DSLEIE

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

Matrix Structure Input by Matrix
DSLEIM

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

End of Matrix Structure Input
DSLEIF

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

Reordering and Symbolic Factorization
DSLEOR

```
INTEGER*4 maxzer,ier
REAL*8 global(150)
CALL DSLEOR (maxzer, global, ier)
```

Matrix Value Input by Single Entry
DSLEV1

```
INTEGER*4 irow,jcol,ier
REAL*8 value,global(150)
CALL DSLEV1 (irow, jcol, value, global, ier)
```

Matrix Value Input by Column
DSLEVC

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

Matrix Value Input by Finite Element
DSLEVE

```
INTEGER*4 nnode,ldelmx,nodlst(nnode),ier
REAL*8 elmmtx(ldelmx,nnode),global(150)
CALL DSLEVE (nnode, nodlst, elmmtx, ldelmx, global, ier)
```

Matrix Value Input by Matrix
DSLEVM

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

Numeric Factorization and Condition Number Estimation
DSLECO

```
INTEGER*4 inrtia(3),ier
REAL*8 pvttol,cond,global(150)
CALL DSLECO (pvttol, cond, inrtia, global, ier)
```

Numeric Factorization
DSLEFA

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

Solve
DSLESL

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

Deallocate Working Storage
DSLEDA

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

Output Control
DSLEOC

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

Print Statistics
DSLEPS

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

Restore Problem State from a Savefile
DSLERS

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

Retrieve Runtime Statistics
DSLESR

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

Save Problem State to a Savefile
DSLESV

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

7 Sparse Eigenvalues/Eigenvectors

This section lists subprograms that solve sparse symmetric ordinary and generalized eigenvalue problems.

One Call Usage
DSEVE1

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

Initialize Sparse Eigenvalues/Eigenvectors
DSEVIN

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

Matrix Structure Input by Single Entry
DSEVI1

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

Matrix Structure Input by Column
DSEVIC

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

Matrix Structure Input by Finite Element
DSEVIE

```
CHARACTER*(*) matrix
INTEGER*4 nnode,nodlst(nnode),ier
REAL*8 global(150)
CALL DSEVIE (matrix, nnode, nodlst, global, ier)
```

Matrix Structure Input by Matrix
DSEVIM

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

End of Matrix Structure Input
DSEVIF

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

Reordering and Symbolic Factorization
DSEVOR

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

Matrix Value Input by Single Entry
DSEVV1

```
CHARACTER*(*) matrix
INTEGER*4 irow, jcol, ier
REAL*8 value, global(150)
CALL DSEVV1 (matrix, irow, jcol, value, global, ier)
```

Matrix Value Input to Main Diagonal
DSEVVD

```
CHARACTER*(*) matrix
INTEGER*4 norder,ier
REAL*8 values(norder),global(150)
CALL DSEVVD (matrix, values, global, ier)
```

Matrix Value Input by Finite Element
DSEVVE

```
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)
```

Matrix Value Input by Matrix
DSEVVM

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

Eigenextraction
DSEVES

```
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)
```

Eigenextraction
DSEVEX

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

Return Eigenvalue/Eigenvector Results
DSEVRC

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

Return Eigenvalue Results
DSEVRL

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

Check Accuracy of Results
DSEVCK

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

Output Control
DSEVOC

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

Restore Problem State from a Savefile
DSEVRS

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

Save Problem State to a Savefile
DSEVSV

```
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.

One-Dimensional FFT — Complex Storage Mode
C1DFFT, Z1DFFT

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

One-Dimensional FFT — Real Storage Mode
S1DFFT, D1DFFT

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

Two-Dimensional FFT — Complex Storage Mode
C2DFFT, Z2DFFT

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

Two-Dimensional FFT — Real Storage Mode
S2DFFT, D2DFFT

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

Three-Dimensional FFT — Complex Storage Mode
C3DFFT, Z3DFFT

```
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)
```

Three-Dimensional FFT — Real Storage Mode
S3DFFT, D3DFFT

```
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)
```

Simultaneous One-Dimensional FFT — Complex Storage Mode
CFFTS, ZFFTS

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

Simultaneous One-Dimensional FFT — Real Storage Mode
SFFTS, DFFTS

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

Real-to-Complex One-Dimensional FFT — Full Storage Mode
CRC1FT, ZRC1FT

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

Real-to-Complex One-Dimensional FFT — Storage Conserving Mode
SRC1FT, DRC1FT

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

Real-to-Complex Two-Dimensional FFT — Full Storage Mode
CRC2FT, ZRC2FT

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

Real-to-Complex Two-Dimensional FFT — Storage Conserving Mode
SRC2FT, DRC2FT

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

Real-to-Complex Three-Dimensional FFT — Full Storage Mode
CRC3FT, ZRC3FT

```
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)
```

Real-to-Complex Three-Dimensional FFT — Storage Conserving Mode
SRC3FT, DRC3FT

```
INTEGER*4 l1,l2,l3,ldx,mdx,iopt,ier
REAL*4 x(ldx,mdx,l2)
CALL SRC3FT (x, l1, l2, l3, ldx, mdx, iopt, ier)
```

Simultaneous Real-to-Complex 1-D FFT — Full Storage Mode
CRCFTS, ZRCFTS

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

Simultaneous Real-to-Complex 1-D FFT — Storage Conserving Mode
SRCFTS, DRCFTS

```
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 subprogram available for correlation and convolution.

Discrete Correlation and Convolution
SCONV, DCONV

```
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 Miscellaneous Routines

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

Measure CPU Time Used
CPUTIME

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

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

To read elapsed time at the end of the code segment:

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

Deallocate Nonvolatile Dynamic Memory
DALLOC

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

Allocate Dynamic Memory
DYNAMIC

```
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
```

Allocate Dynamic Memory
MALLOC

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

Allocate Nonvolatile Dynamic Memory
NALLOC

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

Reallocate Nonvolatile Dynamic Memory
RALLOC

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

Scalar VAX-Compatible Random Numbers
RAN

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

Vector VAX-Compatible Random Numbers
RANV

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

CONVEX to IBM Floating-Point Conversion
SC2IBM

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

Sort Array
SSORT,DSORT,ISORT

```
CHARACTER*1 order
INTEGER*4 n,incx
REAL*4 x(lenx)
CALL SSORT (order, n, x, incx)
```