

Q U I C K R E F E R E N C E

CONVEX SCILIB
Quick Reference

Order No. DSW-361

Second Edition
February 1993



CONVEX

CONVEX COMPUTER CORPORATION

CONVEX SCILIB

Quick Reference

Order No. DSW-361

Document No. 710-014330-003

Second Edition

Copyright © 1991, 1993 CONVEX Computer Corporation
All rights reserved.

This document is copyrighted. This document may not, in whole or part, be copied, duplicated, reproduced, translated, electronically stored, or reduced to machine readable form without prior written consent from CONVEX Computer Corporation.

Although the material contained herein has been carefully reviewed, CONVEX Computer Corporation (CONVEX) does not warrant it to be free of errors or omissions. CONVEX reserves the right to make corrections, updates, revisions or changes to the information contained herein. CONVEX does not warrant the material described herein to be free of patent infringement.

CONVEX and the CONVEX logo "C" are registered trademarks of CONVEX Computer Corporation.

VECLIB is a trademark of CONVEX Computer Corporation.

Cray and UNICOS are registered trademarks of Cray Research, Inc.

Printed in the United States of America.

606 607

Contents

1 Quick Reference Explanation	1
2 Basic Vector Operations	3
3 Basic Matrix Operations	8
4 Linear Equations	14
5 Eigenvalues and Eigenvectors	18
6 Fast Fourier Transforms	19
7 Correlation and Convolution	21
8 Linear Recurrences	22
9 Miscellaneous Routines	24

The *CONVEX SCILIB Quick Reference* provides a quick and efficient method for obtaining basic subroutine usage information about CONVEX SCILIB. This reference lists subprogram names, purposes, and usage examples for all the subprograms in CONVEX SCILIB.

Entries in this quick reference have the following format:

Subprogram Names
Function
Usage Example

Consider the following example:

MINV
Invert Matrix or Solve Linear Equations
INTEGER*8 n, ldab, m, job
REAL*8 ab(ldab, n+m), work(2*n), det, tol
CALL MINV (ab, n, ldab, work, det, tol, m, job)

Please note:

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

SCILIB PRODUCT

The CONVEX SCILIB library is a collection of FORTRAN-callable subprograms that provides a look-alike implementation of Cray Research Incorporated's UNICOS Math and Scientific Library. SCILIB contains all of the public domain routines found in the UNICOS library and all of the CRAY-specific routines.

Introduction

ACCESSING SCILIB

You can incorporate the SCILIB library of subprograms into your programs with a compiler. To incorporate a subprogram, include the appropriate declarations and CALL statements in your FORTRAN source program and specify that SCILIB be used as an object library at link time.

To access SCILIB subprograms, use the `-l` option on the `fc` command line:

```
fc [options] file [linker-options]
```

where `[linker-options]` includes the string:

```
-lscilib
```

If you use any VECLIB routines you'll need to include VECLIB8 also. In this case `-lveclib8` must *follow* `-lscilib` in the list of linker options. A typical `fc` command line might look like:

```
fc -cfc progr1.f -lscilib -lveclib8
```

The `-cfc` option flag specifies Cray FORTRAN mode. For more information, see the *CONVEX FORTRAN Guide*. For information on CONVEX VECLIB routines, see the *CONVEX VECLIB User's Guide*.

DEFAULT DATA TYPES

In Cray FORTRAN, the single precision REAL and INTEGER data types are 64 bits in length; single precision COMPLEX and double precision REAL types are 128 bits in length. In CONVEX SCILIB, it is important to note that *only default single-precision Cray data type lengths are used*. Any REAL or INTEGER variables passed to SCILIB subprgrams must be of length 8 bytes; any COMPLEX variables passed must be of length 16 bytes. For more information, see Table 1-1 in the *CONVEX SCILIB User's Guide*.

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

CLUSEQ/CLUSNE/CLUSFxx/CLUSIxx

(xx = GE, GT, LE, or LT)

Find Clusters of Selected Vector Elements

INTEGER*8 n, incx, a, indx, nindx

REAL*8 x(lenx), a

CALL CLUSEQ (n, x, incx, a, indx, nindx)

GATHER

Gather Sparse Vector

INTEGER*8 m, indx(m)

REAL*8 y(n), x(m)

CALL GATHER (m, x, y, indx)

ILLZ

Count Initial Zero Elements

INTEGER*8 i, ILLZ, n, x(lenx), incx

i = ILLZ (n, x, incx)

ILLZ

Count Initial Positive Elements

INTEGER*8 i, ILLZ, n, x(lenx), incx

i = ILLZ (n, x, incx)

ILSUM

Count TRUE Vector Elements

INTEGER*8 i, ILSUM, n, incx

LOGICAL*8 x(lenx)

i = ILSUM (n, x, incx)

INFLMAX

Index of Maximum Element of Vector

INTEGER*8 i, INFLMAX, n, x(lenx), incx,

mask, rshift

i = INFLMAX (n, x, incx, mask, rshift)

Basic Vector Operations

INFLMIN

Index of Minimum Element of Vector

INTEGER*8 i, INFLMIN, n, x(lenx), incx,
mask, rshift

i = INFLMIN (n, x, incx, mask, rshift)

ISAMAX/ICAMAX

Index of Maximum of Magnitudes

INTEGER*8 i, ISAMAX, n, incx
REAL*8 x(lenx)

i = ISAMAX (n, x, incx)

ISAMIN

Index of Minimum of Magnitudes

INTEGER*8 i, ISAMIN, n, incx
REAL*8 x(lenx)

i = ISAMIN (n, x, incx)

ISMAX/INTMAX

Index of Maximum Element of Vector

INTEGER*8 i, ISMAX, n, incx
REAL*8 x(lenx)

i = ISMAX (n, x, incx)

ISMIN/INTMIN

Index of Minimum Element of Vector

INTEGER*8 i, ISMIN, n, incx
REAL*8 x(lenx)

i = ISMIN (n, x, incx)

ISRCHEQ/ISRCHNE/ISEARCH/ISRCHFxx/ISRCHIxx

(xx = GE, GT, LE, or LT)

Search Vector for Element

INTEGER*8 i, ISRCHEQ, n, incx
REAL*8 x(lenx), a

i = ISRCHEQ (n, x, incx, a)

ISRCHM_{xx}

(xx = EQ, GE, GT, LE, LT, or NE)

Search Vector for Element

INTEGER*8 i, ISRCHM_{xx}, n, x(lenx), incx, a,
mask, rshift
i = ISRCHM_{xx}(n, x, incx, a, mask, rshift)

OSRCHF/OSRCHI

Search Ordered Vector for Element

INTEGER*8 n, incx, indx, ifound, iwould, icount
REAL*8 x(lenx), a
CALL OSRCHF (n, x, incx, a ifound, iwould, icount)

OSRCHM

Search Ordered Vector for Element

INTEGER*8 n, x(lenx), incx, a, mask, rshift,
ifound, iwould, icount
CALL OSRCHM (n, x, incx, a, mask, rshift,
ifound, iwould, icount)

SASUM/SCASUM

Sum of Magnitudes of the Elements of a Vector

INTEGER*8 n, incx
REAL*8 s, SASUM, x(lenx)
s = SASUM (n, x, incx)

SAXPY/CAXPY

Elementary Vector Operation

INTEGER*8 n, incx, incy
REAL*8 a, x(lenx), y(leny)
CALL SAXPY (n, a, x, incx, y, incy)

SCATTER

Scatter Sparse Vector

INTEGER*8 m, indx(m)
REAL*8 y(n), x(m)
CALL SCATTER (m, y, indx, x)

Basic Vector Operations

SCOPY/CCOPY

Copy Vector

```
INTEGER*8 n, incx, incy
REAL*8 x(lenx), y(leny)
CALL SCOPY (n, x, incx, y, incy)
```

SDOT/CDOTC/CDOTU

Dot Product of Two Vectors

```
INTEGER*8 n, incx, incy
REAL*8 s, SDOT, x(lenx), y(leny)
s = SDOT (n, x, incx, y, incy)
```

SNRM2/SCNM2

Euclidean Norm of a Vector

```
INTEGER*8 n, incx
REAL*8 s, SNRM2, x(lenx)
s = SNRM2 (n, x, incx)
```

SPAXPY

Sparse Elementary Vector Operation

```
INTEGER*8 m, indx(m)
REAL*8 a, x(m), y(n)
CALL SPAXPY (m, a, x, y, indx)
```

SPDOT

Sparse Dot Product

```
INTEGER*8 m, indx(m)
REAL*8 s, SPDOT, y(n), x(m)
s = SPDOT (m, y, indx, x)
```

SROT/CRO/

Apply a Givens Rotation to Two Vectors

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

SROTG/CROTG

Construct a Givens Rotation

```
REAL*8 a, b, c, s
CALL SROTG (a, b, c, s)
```

Basic Vector Operations

SROTM

Apply a Modified Givens Rotation

```
INTEGER*8 n, incx, incy
REAL*8 x(lenx), y(leny), param(5)
CALL SROTM (n, x, incx, y, incy, param)
```

SROTMG

Construct a Modified Givens Rotation

```
REAL*8 d1, d2, x1, y1, param(5)
CALL SROTMG (d1, d2, x1, y1, param)
```

SSCAL/CSCAL/CSSCAL

Scale a Vector

```
INTEGER*8 n, incx
REAL*8 a, x(lenx)
CALL SSCAL (n, a, x, incx)
```

SSUM/CSUM

Sum of the Elements of a Vector

```
INTEGER*8 n, incx
REAL*8 s, SSUM, x(lenx)
s = SSUM (n, x, incx)
```

SSWAP/CSWAP

Swap Two Vectors

```
INTEGER*8 n, incx, incy
REAL*8 x(lenx), y(leny)
CALL SSWAP (n, x, incx, y, incy)
```

WHENEQ/WHENNE/WHENFxx/WHENLxx

(xx = GE, GT, LE, or LT)

Find Selected Vector Elements

```
INTEGER*8 n, incx, indx(n), nindx
REAL*8 x(lenx), a
CALL WHENEQ (n, x, incx, a, indx, nindx)
```

WHENMxx

(xx = EQ, GE, GT, LE, LT, or NE)

Find Selected Vector Elements

```
INTEGER*8 n, x(lenx), incx, a, indx(n), nind,
mask, rshift
CALL WHENMxx(n, x, incx, a, indx, nindx,
mask, rshift)
```

This section lists the subprograms for performing basic matrix operations. They are functionally equivalent to the Extended BLAS subprograms found in Chapter 3 of the *CONVEX VECLIB User's Guide* in that they produce equivalent answers for the same problems, but with different usage.

MXM

Specialized Matrix-Matrix Multiply

INTEGER*8 m, k, n

REAL*8 a(m, k), b(k, n), c(m, n)

CALL MXM (a, m, b, k, c, n)

MXMA

Generalized Matrix-Matrix Multiply

INTEGER*8 ia, ja, ib, jb, ic, jc, m, k, n

REAL*8 a(lena), b(lenb), c(lenc)

CALL MXMA (a, ia, ja, b, ib, jb, c, ic, jc, m, k, n)

MXV

Specialized Matrix-Vector Multiply

INTEGER*8 m, n

REAL*8 a(m, n), x(n), y(m)

CALL MXV (a, m, x, n, y)

MXVA

Generalized Matrix-Vector Multiply

INTEGER*8 ia, ja, incx, incy, m, n

REAL*8 a(lena), x(lenx), y(leny)

CALL MXVA (a, ia, ja, x, incx, y, incy, m, n)

SGBMV/CGBMV

General Band Matrix-Vector Multiply

CHARACTER*1 trans

INTEGER*8 m, n, kl, ku, ldab, incx, incy

REAL*8 alpha, beta, ab(ldab, n), x(lenx), y(leny)

CALL SGBMV (trans, m, n, kl, ku, alpha, ab, ldab,
x, incx, beta, y, incy)

SGEMM/CGEMM

General Matrix-Matrix Multiply

CHARACTER*1 transa, transb
 INTEGER*8 m, n, k, lda, ldb, ldc
 REAL*8 alpha, beta, a(lda, *), b(ldb, *), c(ldc, n)
 CALL SGEMMS (transa, transb, m, n, k, alpha, a, lda,
 b, ldb, beta, c, ldc)

SGEMMS/CGEMMS

Strassen Matrix-Matrix Multiply

CHARACTER*1 transa, transb
 INTEGER*8 m, n, k, lda, ldb, ldc
 REAL*8 alpha, beta, a(lda, *), b(ldb, *),
 c(ldc, n), work(lwork)
 CALL SGEMMS (transa, transb, m, n, k, alpha, a, lda,
 b, ldb, beta, c, ldc, work)

SGEMV/CGEMV

General Matrix-Vector Multiply

CHARACTER*1 trans
 INTEGER*8 m, n, lda, incx, incy
 REAL*8 alpha, beta, a(lda, n), x(lenx), y(leny)
 CALL SGEMV (trans, m, n, alpha, a, lda, x, incx,
 beta, y, incy)

SGER/CGERC/CGERU

General Rank-1 Update

INTEGER*8 m, n, lda, incx, incy
 REAL*8 alpha, a(lda, n), x(lenx), y(leny)
 CALL SGER (m, n, alpha, x, incx, y, incy, a, lda)

SMXPY

Matrix-Vector Multiply and Add

INTEGER*8 m, n, lda
 REAL*8 a(lda, n), x(n), y(m)
 CALL SMXPY (m, y, n, lda, x, a)

SSBMV/CHBMV

Symmetric or Hermitian Band Matrix-Vector Multiply

CHARACTER*1 uplo
 INTEGER*8 n, kd, ldab, incx, incy
 REAL*8 alpha, beta, ab(ldab, n), x(lenx), y(leny)
 CALL SSBMV (uplo, n, kd, alpha, ab, ldab, x, incx,
 beta, y, incy)

Basic Matrix Operations

SSPMV/CHPMV

Symmetric or Hermitian Matrix-Vector Multiply

CHARACTER*1 uplo

INTEGER*8 n, incx, incy

REAL*8 alpha, beta, ap(lenap), x(lenx), y(leny)

CALL SSPMV (uplo, n, alpha, ap, x, incx,
beta, y, incy)

SSPR/CHPR

Symmetric Rank-1 Update

CHARACTER*1 uplo

INTEGER*8 n, incx

REAL*8 alpha, ap(lenap), x(lenx)

CALL SSPR (uplo, n, alpha, x, incx, ap)

SSPR2/CHPR2

Symmetric or Hermitian Rank-2 Update

CHARACTER*1 uplo

INTEGER*8 n, incx, incy

REAL*8 alpha, ap(lenap), x(lenx), y(leny)

CALL SSPR2 (uplo, n, alpha, x, incx, y, incy, ap)

SSYMM/CHEMM/CSYM/

Symmetric or Hermitian Matrix-Matrix Multiply

CHARACTER*1 side, uplo

INTEGER*8 m, n, lda, ldb, ldc

REAL*8 alpha, beta, a(lda, *), b(ldb, *), c(ldc, *)

CALL SSYMM (side, uplo, m, n, alpha, a, lda, b, ldb,
beta, c, ldc)

SSYMV/CHEMV

Symmetric or Hermitian Matrix-Vector Multiply

CHARACTER*1 uplo

INTEGER*8 n, lda, incx, incy

REAL*8 alpha, beta, a(lda, n), x(lenx), y(leny)

CALL SSYMV (uplo, n, alpha, a, lda, x, incx, beta, y, incy)

SSYR/CHER

Symmetric or Hermitian Rank-1 Update

CHARACTER*1 uplo

INTEGER*8 n, lda, incx

REAL*8 alpha, a(lda, n), x(lenx)

CALL SSYR (uplo, n, alpha, x, incx, a, lda)

SSYR2/CHER2

Symmetric or Hermitian Rank-2 Update

CHARACTER*1 uplo
 INTEGER*8 n, lda, incx, incy
 REAL*8 alpha, a(lda, n), x(lenx), y(leny)
 CALL SSYR2 (uplo, n, alpha, x, incx, y, incy,
 a, lda)

SSYR2K/CSYR2K

Symmetric or Hermitian Rank-2k Update

CHARACTER*1 uplo, trans
 INTEGER*8 n, k, lda, ldb, ldc
 REAL*8 alpha, beta, a(lda, *), b(ldb, *), c(ldc, *)
 CALL SSYR2K (uplo, trans, n, k, alpha, a, lda,
 b, ldb, beta, c, ldc)

SSYRK/CHERK/CSYRK

Symmetric or Hermitian Rank-k Update

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

STBMV/CTBMV

Triangular Band Matrix-Vector Multiply

CHARACTER*1 uplo, trans, diag
 INTEGER*8 n, kd, ldab, incx
 REAL*8 ab(ldab, n), x(lenx)
 CALL STBMV (uplo, trans, diag, n, kd, ab, ldab, x, incx)

STBSV/CTBSV

Solve Triangular Band System

CHARACTER*1 uplo, trans, diag
 INTEGER*8 n, kd, ldab, incx
 REAL*8 ab(ldab, n), x(lenx)
 CALL STBSV (uplo, trans, diag, n, kd, ab, ldab, x, incx)

STPMV/CTPMV

Triangular Matrix-Vector Multiply

CHARACTER*1 uplo, trans, diag
 INTEGER*8 n, incx
 REAL*8 ap(lenap), x(lenx)
 CALL STPMV (uplo, trans, diag, n, ap, x, incx)

Basic Matrix Operations

STPSV/CTPSV

Solve One Triangular System

```
CHARACTER*1 uplo, trans, diag  
INTEGER*8 n, incx  
REAL*8 ap(lenap), x(lenx)  
CALL STPSV (uplo, trans, diag, n, ap, x, incx)
```

STRMM/CTRMM

Triangular Matrix-Matrix Multiply

```
CHARACTER*1 side, uplo, transa, diag  
INTEGER*8 m, n, lda, ldb  
REAL*8 alpha, a(lda, *), b(ldb, *)  
CALL STRMM (side, uplo, transa, diag, m, n,  
alpha, a, lda, b, ldb)
```

STRMV/CTRMV

Triangular Matrix-Vector Multiply

```
CHARACTER*1 uplo, trans, diag  
INTEGER*8 n, lda, incx  
REAL*8 a(lda, n), x(lenx)  
CALL STRMV (uplo, trans, diag, n, a, lda, x, incx)
```

STRSM/CTRSM

Solve Simultaneous Triangular Systems

```
CHARACTER*1 side, uplo, transa, diag  
INTEGER*8 m, n, lda, ldb  
REAL*8 alpha, a(lda, *), b(ldb, *)  
CALL STRSM (side, uplo, transa, diag, m, n,  
alpha, a, lda, b, ldb)
```

STRSV/CTRSV

Solve One Triangular System

```
CHARACTER*1 uplo, trans, diag  
INTEGER*8 n, lda, incx  
REAL*8 a(lda, n), x(lenx)  
CALL STRSV (uplo, trans, diag, n, a, lda, x, incx)
```

SXMPY

Matrix-Vector Multiply and Add

```
INTEGER*8 n, m, lda, incx, incy  
REAL*8 a(lda, n), x(lenx), y(leny)  
CALL SXMPY (n, incy, y, m, incx, x, lda, a)
```

Basic Matrix Operations

XERBLA

BLAS Error Handler

CHARACTER*6 name

INTEGER*8 larg

CALL XERBLA (name, larg)

This section lists the subprograms for solving linear equations documented in the *CONVEX SCILIB User's Guide*.

MINV

Invert Matrix or Solve Linear Equations

INTEGER*8 n, ldab, m, job
REAL*8 ab(ldab, n+m), work(2*n), det, tol
CALL MINV (ab, n, ldab, work, det, tol, m, job)

OPFILT

Solve Symmetric Toeplitz Linear Equations

INTEGER*8 n
REAL*8 x(n), b(n), work(2*n), q(n)
CALL OPFILT (n, x, b, work, q)

SGBCO/CGBCO

Factor a General Band Matrix and Estimate its
Condition Number

INTEGER*8 ldab, n, kl, ku, ipvt(n)
REAL*8 ab(ldab, n), rcond, work(n)
CALL SGBCO (ab, ldab, n, kl, ku, ipvt, rcond, work)

SGBDI/CGBDI

Determinant of a General Band Matrix

INTEGER*8 ldab, n, kl, ku, ipvt(n)
REAL*8 ab(ldab, n), det(2)
CALL SGBDI (ab, ldab, n, kl, ku, ipvt, det)

SGBFA/CGBFA

Factor a General Band Matrix

INTEGER*8 ldab, n, kl, ku, ipvt(n), ier
REAL*8 ab(ldab, n)
CALL SGBFA (ab, ldab, n, kl, ku, ipvt, ier)

SGBSL/CGBSL

Solve Linear Equations with a General Band Matrix

INTEGER*8 ldab, n, kl, ku, ipvt(n), job

REAL*8 ab(ldab, n), b(n)

CALL SGBSL (ab, ldab, n, kl, ku, ipvt, b, job)

SGECO/CGECO

Factor a General Matrix and Estimate its Condition Number

INTEGER*8 lda, n, ipvt(n)

REAL*8 a(lda, n), rcond, work(n)

CALL SGECO (a, lda, n, ipvt, rcond, work)

SGEDI/CGEDI

Determinant and Inverse of a General Matrix

INTEGER*8 lda, n, ipvt(n), job

REAL*8 a(lda, n), det(2), work(n)

CALL SGEDI (a, lda, n, ipvt, det, work, job)

SGEFA/CGEFA

Factor a General Matrix

INTEGER*8 lda, n, ipvt(n), ier

REAL*8 a(lda, n)

CALL SGEFA (a, lda, n, ipvt, ier)

SGESL/CGESL

Solve Linear Equations with a General Matrix

INTEGER*8 lda, n, ipvt(n), job

REAL*8 a(lda, n), b(n)

CALL SGESL (a, lda, n, ipvt, b, job)

SGTSL/CGTSL

Solve Linear Equations with a Tridiagonal Matrix

INTEGER*8 n, ier

REAL*8 dl(n), d(n), du(n), b(n)

CALL SGTSL (n, dl, d, du, b, ier)

Linear Equations

SPBCO/CPBCO

Factor a Positive Definite Band Matrix and Estimate its Condition Number

```
INTEGER*8 ldab, n, kd, ier  
REAL*8 ab(ldab, n), rcond, work(n)  
CALL SPBCO (ab, ldab, n, kd, rcond, work, ier)
```

SPBDI/CPBDI

Determinant of a Positive Definite Band Matrix

```
INTEGER*8 ldab, n, kd  
REAL*8 ab(ldab, n), det(2)  
CALL SPBDI (ab, ldab, n, kd, det)
```

SPBFA/CPBFA

Cholesky Factorization of a Positive Definite Band Matrix

```
INTEGER*8 ldab, n, kd, ier  
REAL*8 ab(ldab, n)  
CALL SPBFA (ab, ldab, n, kd, ier)
```

SPBSL/CPBSL

Solve Linear Equations with a Positive Definite Band Matrix

```
INTEGER*8 ldab, n, kd  
REAL*8 a(ldab, n), b(n)  
CALL SPBSL (a, ldab, n, kd, b)
```

SPOCO/CPOCO

Factor a Positive Definite Matrix and Estimate its Condition Number

```
INTEGER*8 lda, n, ier  
REAL*8 a(lda, n), rcond, work(n)  
CALL SPOCO (a, lda, n, rcond, work, ier)
```

SPODI/CPODI

Determinant and Inverse of a Positive Definite Matrix

```
INTEGER*8 lda, n, job  
REAL*8 a(lda, n), det(2)  
CALL SPODI (a, lda, n, det, job)
```

SPOFA/CPOFA

Cholesky Factorization of a Positive Definite Matrix

INTEGER*8 lda, n, ier
REAL*8 a(lda, n)
CALL SPOFA (a, lda, n, ier)

SPOSL/CPOSL

Solve Linear Equations with a Positive Definite Matrix

INTEGER*8 lda, n
REAL*8 a(lda, n), b(n)
CALL SPOSL (a, lda, n, b)

SPTSL/CPTSL

**Solve Linear Equations with a Positive Definite
Tridiagonal Matrix**

INTEGER*8 n
REAL*8 d(n), e(n-1), b(n)
CALL SPTSL (n, d, e, b)

This section lists the SCILIB subprograms to compute eigenvalues or eigenvalues and eigenvectors of matrices.

RS

Eigenvalues and Eigenvectors of a Real Symmetric Matrix

```
INTEGER*8 ldx, 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*8 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 Tridiagonal Matrix

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

TRED1

Reduce a Real Symmetric Matrix to Tridiagonal Form

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

TRED2

Reduce a Real Symmetric Matrix to Tridiagonal Form

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

This section lists the SCILIB Fast Fourier transform (FFT) subprograms.

CFFT2

One-Dimensional Complex-to-Complex FFT

```
INTEGER*8 init, sign, l
COMPLEX*16 zin(l), zout(l)
REAL*8 work(5*l)
CALL CFFT2 (init, isgn, l, zin, work, zout)
```

CFTFAX/CFFTMLT

Simultaneous One-Dimensional FFT

```
INTEGER*8 work3(10), incl, incn, l, n, isgn
REAL*8 x(lenxy), y(lenxy), work1(4*l*n), work2(2*l)
```

Initialization call:

```
CALL CFTFAX (l, work3, work2)
```

Transform call:

```
CALL CFFTMLT (x, y, work1, work2, work3,
              incl, incn, l, n, isgn)
```

CRFFT2

Complex-to-Real One-Dimensional FFT

```
INTEGER*8 init, isgn, l
COMPLEX*16 zin(l/2+1)
REAL*8 work(3*l+4), xout(l)
CALL CRFFT2(init, isgn, l, zin, work, xout)
```

RCFFT2

Real-to Complex One-Dimensional FFT

```
INTEGER*8 init, isgn, l
REAL*8 xin(l) work(3*l+4)
COMPLEX*16 zout(l/2+1)
CALL RCFFT2(init, isgn, l, xin, work, zout)
```

Fast Fourier Transforms

FFTFAX/RFF TMLT

Simultaneous One-Dimensional FFT

INTEGER*8 work3(10), incl, incn, l, n, isgn
REAL*8 x(enx), work1(2*I*n), work2(2*I)

Initialization call:

CALL FFT FAX (l, work3, work2)

Transform call:

CALL RFF TMLT (x, work1, work2, work3,
incl, incn, l, n, isgn)

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

FILTERG

Discrete Correlation

INTEGER*8 m, n

REAL*8 f(m), x(n), y(n-m+1)

CALL FILTERG (f, m, x, n, y)

FILTERS

Discrete Correlation

INTEGER*8 m, n

REAL*8 f((m+1)/2), x(n), y(n-m+1)

CALL FILTERS (f, m, x, n, y)

This section lists the SCILIB subprograms available for solving a variety of linear recurrence operations.

FOLR/FOLRI

First Order Linear Recurrence

```
INTEGER *8 n, inca, incx
REAL*8 a(lena), x(lenx)
CALL FOIR (n, a, inca, x, incx)
```

FOLR2/FOLR2P

First Order Linear Recurrence

```
INTEGER *8 n, inca, incx, incy
REAL*8 a(lena), x(lenx), y(leny)
CALL FOIR2(n, a, inca, x, incx, y, incy)
```

FOLRC

First Order Linear Recurrence

```
INTEGER *8 n, incx, inca
REAL*8 x(lenx), a(lena), alpha
CALL FOIRC (n, x, incx, a, inca, alpha)
```

FOLRN/FOLRN

Last Term of First Order Linear Recurrence

```
INTEGER *8 n, inca, incx
REAL*8 y1, FOLRN, a(lena), x(lenx)
yn = FOIRN (n, a, inca, x, incx)
```

RECPP

Compute Vector of Partial Products

```
INTEGER *8 n, incx, inca
REAL*8 x(lenx), a(lena)
CALL RECPP (n, x, incx, a, inca)
```

RECPS

Compute Vector of Partial Sums

```
INTEGER *8 n, incx, inca
REAL*8 x(lenx), a(lena)
CALL RECPS (n, x, incx, a, inca)
```

SOLR

Second Order Linear Recurrence

INTEGER*8 n, inca, incb, incx

REAL*8 a(lena), b(lenb), x(lenx)

CALL SOLR (n, a, inca, b, incb, x, incx)

SOLR3

Second Order Linear Recurrence

INTEGER*8 n, inca, incb, incx

REAL*8 a(lena), b(lenb), x(lenx)

CALL SOLR3(n, a, inca, b, incb, x, incx)

SOLRN

Last Term of Second Order Linear Recurrence

INTEGER*8 n, inca, incb, incx

REAL*8 xn, SOLRN, a(lena), b(lenb), x(lenx)

xn = SOLRN (n, a, inca, b, incb, x, incx)

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

ORDERS

Sort Array into Ascending Order

```
INTEGER* 3 mode, work(lwork), indx(n), n, ldx,  
          keylen iradix  
REAL*8 x(lx, n)  
CALL ORDERS (mode, work, x, indx, n, ldx,  
            keylen iradix)
```

XERSCI

SCILIB Error Handler

```
CHARACTER*(*) name, message  
INTEGER* 3 larg  
CALL XERSCI (name, larg, messag)
```















CONVEX Computer Corporation
Richardson, Texas USA

Document No. 710-014330-003