4.1 Square Nonsingular Systems of Equations

A. Purpose

For a square nonsingular matrix, A, of order N, subroutines are provided for solving systems of equations involving the matrix A or A^t(A^h in the complex case), and for computing the inverse, determinant, or (reciprocal) condition number of A. Versions are provided for REAL, DOUBLE PRECISION and COMPLEX data types.

B. Usage

The individual subroutines are described as follows:

B.1 SGEFS Factor and solve.

B.2 SGEFSC Factor, solve, and reciprocal condition number.

B.3 SGEFA Factor.

B.4 SGECO Factor and reciprocal condition number.

B.5 SGESLD Solve $A\mathbf{x} = \mathbf{c}$.

B.6 SGESLT Solve $A^t\mathbf{x} = \mathbf{c}$. ($A^h\mathbf{x} = \mathbf{c}$ for the complex case.)

B.7 SGED Determinant.

B.8 SGEI Inverse.

B.9 Modifications for double-precision and complex.

Subroutine SGEFA is the fundamental subroutine of this set. It does the forward pass of Gaussian elimination with partial pivoting, replacing the matrix A, given in the array A(,), with two triangular matrices representing its LU factorization and storing a record in IPVT() of the row interchanges done during the factorization. The other subroutines of this set either call SGEFA to accomplish this factorization, or else must be used after the factorization has already been computed by SGEFA.

This code is all derived from LINPACK [1], but the organization of the code into separate subroutines differs from LINPACK.

B.1 Usage to factor and solve

Use SGEFS to factor A and solve the system, $AX = B$, where A is an N×N square nonsingular matrix and B is an N×NB matrix. The N×NB solution matrix, X, will overwrite the given matrix, B, and the factors of A will overwrite A. SGEFS calls SGEFA to factor A, and calls SGESLD, NB times, to compute the columns of X.

B.1.a Program Prototype, Single Precision

INTEGER LDA, N, LDB, NB, IPVT(≥N)**, INFO**

REAL A(LDA, ≥N)**, B**(LDB, ≥NB)

Assign values to A(,), LDA, N, B(,), LDB, and NB.

CALL SGEFS (A, LDA, N, B, LDB, NB, IPVT, INFO)

Computed quantities are returned in A(,), B(,), IPVT(), and INFO.

B.1.b Argument Definitions

A(,) [inout] On entry contains the N×N matrix A. On return contains the LU factorization of A as computed by SGEFA.

LDA [in] Leading dimensioning parameter for the array A(,). Require LDA ≥ N.

N [in] The order of the matrix A and number of rows in the matrices B and X.

B(,) [inout] On entry contains the N×NB matrix, B. On return contains the N×NB solution matrix, X. The array B(,) can be declared as singly subscripted if NB = 1. No solution will be computed if the returned value of INFO is nonzero.

LDB [in] Leading dimensioning parameter for the array B(,). Require LDB ≥ N.

NB [in] Number of columns in the matrices B and X. If NB < 1, the matrix, A, will be factored but no reference will be made to the array B(,).

IPVT() [out] Integer array of length at least N. On return will contain a record of the row interchanges done during factorization of A.

INFO [out] Set to zero if all diagonal elements in the U matrix of the LU factorization are nonzero. If nonzero, INFO is the index of the first diagonal element of U that is zero. In this latter case the solution X will not be computed.

B.2 Usage to factor, solve, and compute reciprocal condition number

Use SGEFSC to factor A and solve the system, $AX = B$, where A is an N×N nonsingular matrix and B is an N×NB matrix. The N×NB solution matrix, X, will overwrite the given matrix, B, and the factors of A will overwrite A. In addition SGEFSC sets RCOND to the reciprocal condition number of A.

SGEFSC calls SGECO to factor A and compute RCOND, and then calls SGESLD, NB times, to compute the columns of X.

B.2.a Program Prototype, Single Precision

INTEGER LDA, N, LDB, NB, IPVT(≥N)

REAL A(LDA, ≥N)**, B**(LDB, ≥NB)**, RCOND, Z**(≥N)

Assign values to A(,), LDA, N, B(,), LDB, and NB.

CALL SGEFSC(A, LDA, N, B, LDB, NB, IPVT, RCOND, Z)

Computed quantities are returned in A(,), B(,), IPVT(), RCOND, and Z().

B.2.b Argument Definitions

The first seven arguments for SGEFSC have the same meaning as for SGEFS. The final two arguments are defined as follows:

RCOND [out] An estimate for the reciprocal condition number of A. Will satisfy $0.0 \leq \text{RCOND} \leq 1.0$. A zero value indicates a singular matrix. Larger values indicate a better conditioned matrix.

Z() [scratch] An array of length N used as working space in SGEFSC. The contents on return will generally not be of interest to the user. It will be an N-vector, $\mathbf{z}$, that satisfies

$$\|A\mathbf{z}\| = RCOND \cdot \|A\| \cdot \|\mathbf{z}\|$$

If A is nearly singular then $\mathbf{z}$ will be an approximate null vector.

B.3 Usage to factor only

Use SGEFA to factor a given N×N matrix, A, obtaining matrices, L and U, satisfying $LU = A$, where U is upper triangular and L is a permutation of a lower triangular matrix. U and a representation of L^{-1} will be returned in the array A(,), and a record of the permutations will be returned in IPVT().

B.3.a Program Prototype, Single Precision

INTEGER LDA, N, IPVT(≥N)**, INFO**

REAL A(LDA, ≥N)

Assign values to A(,), LDA, and N.

CALL SGEFA(A, LDA, N, IPVT, INFO)

Computed quantities are returned in A(,), IPVT(), and INFO.

B.3.b Argument Definitions

The arguments have the same meaning as the arguments of the same names for the subroutine SGEFS described above in Section B.1.

B.4 Usage to factor and compute reciprocal condition number

SGECO calls SGEFA to factor the given matrix, A, and then computes RCOND as an estimate of the reciprocal condition number of A.

B.4.a Program Prototype, Single Precision

INTEGER LDA, N, IPVT(≥N)

REAL A(LDA, ≥N)**, RCOND, Z**(≥N)

Assign values to A(,), LDA, and N.

CALL SGECO(A, LDA, N, IPVT, RCOND, Z)

Computed quantities are returned in A(,), IPVT(), RCOND, and Z().

B.4.b Argument Definitions

A(,), LDA, N, IPVT() See description in Section B.1 above.

RCOND, Z() See description in Section B.2 above.

B.5 Usage to solve the direct system, $A\mathbf{x} = \mathbf{c}$

Given A(,) and IPVT() containing factorization results produced by SGEFA for a matrix, A, use SGESLD to solve the direct system, $A\mathbf{x} = \mathbf{c}$.

B.5.a Program Prototype, Single Precision

INTEGER LDA, N, IPVT(≥N)

REAL A(LDA, ≥N)**, C**(≥N)

Values must initially be present in A(,), LDA, N, IPVT(), and C().

CALL SGESLD(A, LDA, N, IPVT, C)

Computed quantities are returned in C().

B.5.b Argument Definitions

A(,) [in] On entry contains the N×N set of numbers representing the LU decomposition of a matrix A as computed by SGEFA. The contents of A(,) are not altered by this subroutine.

LDA [in] Leading dimensioning parameter for the array A(,). Require LDA ≥ N.

N [in] The order of the original matrix A.

IPVT() [in] Integer array of length at least N. On entry contains a record of the row interchanges done during factorization of A.

C() [inout] On entry must contain the right-side N-vector for the problem, $A\mathbf{x} = \mathbf{c}$. On return, if A is nonsingular, C() will contain the solution N-vector, $\mathbf{x}$. See Section E for discussion of singularity.

B.6 Usage to solve the transposed system, $A^t\mathbf{x} = \mathbf{c}$

Given A(,) and IPVT() containing factorization results produced by SGEFA for a matrix, A, use SGESLT to solve the transposed system, $A^t\mathbf{x} = \mathbf{c}$. In the complex case, *i.e.* using CGEFA rather than SGEFA, the problem solved is $A^h\mathbf{x} = \mathbf{c}$, where A^h denotes the conjugate transpose of A.

B.6.a Program Prototype, Single Precision

INTEGER LDA, N, IPVT(≥N)

REAL A(LDA, ≥N)**, C**(≥N)

Values must initially be present in A(,), LDA, N, IPVT(), and C().

```
CALL SGESLT(A, LDA, N, IPVT, C)
```

Computed quantities are returned in C().

B.6.b Argument Definitions

A(,), LDA, N, IPVT() [in] See description in Section B.5 above.

C() [inout] On entry must contain the right-side N-vector for the problem, $A^t\mathbf{x} = \mathbf{c}$. On return, if A is nonsingular, C() will contain the solution N-vector, $\mathbf{x}$. See Section E for discussion of singularity.

B.7 Usage to compute the determinant of A

Given A(,) and IPVT() containing factorization results produced by SGEFA for a matrix, A, use SGED to compute the determinant of A.

B.7.a Program Prototype, Single Precision

INTEGER LDA, N, IPVT(≥N)

REAL A(LDA, ≥N)**, DET**(2)

Values must initially be present in A(,), LDA, N, and IPVT().

```
CALL SGED(A, LDA, N, IPVT, DET)
```

Computed quantities are returned in DET(1) and DET(2).

B.7.b Argument Definitions

A(,), LDA, N, IPVT() [in] See description in Section B.5 above.

DET() [out] DET(1) and DET(2) will be set to values representing the determinant of A in the form:

$$\text{Determinant} = \text{DET}(1) \times 10^{\text{DET}(2)}$$

DET(2) will contain an integer value, which may be positive, negative, or zero.

If the determinant is zero, both DET(1) and DET(2) will be zero.

If the determinant is nonzero, DET(1) and DET(2) will be some choice among the many pairs of values that could be used to represent the value of the determinant. The algorithm tends to produce a representation with DET(2) = 0 if the magnitude of the determinant is not extremely large or small, however it is not designed to select this representation in all possible cases. See Section D for further explanation.

B.8 Usage to compute the inverse matrix of A

Given A(,) and IPVT() containing factorization results produced by SGEFA for a matrix, A, use SGEI to compute the inverse matrix of A.

B.8.a Program Prototype, Single Precision

INTEGER LDA, N, IPVT(≥N)

REAL A(LDA, ≥N)**, WORK**(≥N)

Values must initially be present in A(,), LDA, N, and IPVT().

```
CALL SGEI(A, LDA, N, IPVT, WORK)
```

Computed quantities are returned in A(,).

B.8.b Argument Definitions

A(,) [inout] On entry contains an N×N set of numbers representing the LU decomposition of a matrix A as computed by SGEFA. On return, if A is nonsingular, contains the inverse matrix of A. See Section E for discussion of singularity.

LDA, N, IPVT() [in] See description in Section B.5 above.

WORK() [scratch] An array of at least N locations used as internal work space.

B.9 Modifications for Double Precision or Complex

For double-precision usage change the initial letter of each subroutine name from S to D, and change the REAL declarations to DOUBLE PRECISION.

For complex usage change the initial letter of each subroutine name from S to C, and change the REAL declarations to COMPLEX, with the exception that the argument, RCOND, must remain REAL.

Note that the COMPLEX subroutine CGESLT solves $A^h\mathbf{x} = \mathbf{c}$, where A^h denotes the conjugate transpose of A.

C. Examples and Remarks

Program DRSGEFSC illustrates the use of subroutine SGEFSC to solve a system of linear equations and compute the reciprocal condition number of the matrix of the system. Output is shown in ODSGEFSC. The data for this problem were chosen so the exact solution has components 2, −5, and 3.

Avoiding computation of the inverse

If A is a nonsingular matrix, the relations $AX = B$ and $X = A^{-1}B$ are mathematically equivalent. Thus, given A and B, where B and thus X may be either a matrix or a vector, one could either solve the system $AX = B$ for X, or one could compute A^{-1} and then multiply A^{-1} times B to obtain X. The former approach will be faster as it requires fewer arithmetic operations. It will also generally be slightly more accurate. Thus wherever an inverse matrix appears in an expression to be computed, it is preferable to formulate the computation in terms of solving systems rather than computing inverses, unless the inverse matrix is needed for another reason.

D. Functional Description

This code is all derived from LINPACK [1]. Reference [1] gives complete descriptions of the algorithms implemented.

SGEFA and SGECO are the same in name, argument list, and functionality in Fortran 77 as the LINPACK subroutines of the same name. SGESLD and SGESLT provide the two distinct functionalities provided by the LINPACK subroutine, SGESL. SGED and SGEI provide the two distinct functionalities provided by the LINPACK subroutine, SGEDI. SGEFS and SGEFSC are shell subroutines added to package certain convenient sequences of calls to the other subroutines.

D.1 Factorization

The fundamental subroutine of this set is SGEFA. SGEFA performs the forward pass of Gaussian elimination with partial pivoting. Partial pivoting means row interchanges are used to bring the element of largest magnitude, at or below the diagonal position in the pivot column, to the diagonal position at each stage of the elimination process. A record of the row interchanges is kept in IPVT(). This algorithm has very good numerical stability and efficiency, the count of arithmetic operations being $n^3/3 + O(n^2)$ additions and multiplications.

The result of this process is a factorization of the given matrix, A, of the form

$$A = LU = P_1K_1P_2K_2 \cdots P_{n-1}K_{n-1}U$$

where U is an upper triangular matrix, each P_i is a permutation matrix representing the permutation of the index i with an index $\geq i$, and each K_i differs from the identity only by having nonzero elements below the diagonal in column i. SGEFA stores a convenient representation of this factorization in the arrays A(,) and IPVT(). Each of the subroutines SGESLD, SGESLT, SGED, and SGEI is designed to use this factorization as the starting point for its computation.

D.2 Scaling

Even with partial pivoting, the accuracy of the solution can be adversely affected if the scaling is extremely bad. There is no universally applicable algorithmic criterion for determining a "good" scaling, and consequently the LINPACK subroutines do not introduce any scaling. The user should assure that the scaling is reasonable for his or her application. One reasonable approach is to scale the rows and columns of A so that, if possible, the absolute size of the uncertainty in each element of the matrix, A, is nearly the same for all elements.

D.3 Solving equations using SGESLD or SGESLT

SGESLD completes the solution of the problem, $A\mathbf{x} = \mathbf{b}$. Given the factorization, $A = LU$, this is done by first solving $L\mathbf{y} = \mathbf{b}$, and then solving $U\mathbf{x} = \mathbf{y}$.

SGESLT completes the solution of the problem, $A^t\mathbf{x} = \mathbf{b}$. Given the factorization, $A = LU$, this is done by first solving $U^t\mathbf{y} = \mathbf{b}$, and then solving $L^t\mathbf{x} = \mathbf{y}$.

Each of these two subroutines has an operation count of $n^2 + O(n)$ additions and multiplications.

D.4 Computing A^{-1}

SGEI completes the computation of A^{-1}. Given the factorization, $A = LU$, A^{-1} can be expressed as $A^{-1} = U^{-1}L^{-1}$. SGEI first computes the triangular matrix U^{-1}, overwriting U. It then does the two steps of forming L^{-1} and multiplying U^{-1} times L^{-1} in an interleaved manner to conserve storage. The algorithm uses n^2 locations in A(,) and n locations in WORK(). Operations required are $(2/3)n^3 + O(n^2)$ additions and multiplications.

D.5 Estimating the reciprocal condition number

The condition number of an $n \times n$ nonsingular matrix, A, is defined as $\kappa = \|A\| \times \|A^{-1}\|$, a quantity that is never less than unity. This is the largest factor by which the relative error in a vector, $\mathbf{y}$, can be amplified as a result of multiplication by A or by A^{-1}. Roughly speaking, if $\kappa = 10^k$ and the components of the vector $\mathbf{b}$ in the problem, $A\mathbf{x} = \mathbf{b}$, are known to d decimal digits, and the components of A are known to more than d decimal

digits, and the problem is solved using precision greater than d decimal digits, then the solution will be known to about $d-k$ decimal digits. In particular, if $k \geq d$, the solution may have no reliable digits at all. (See [1] for a more precise discussion of the relation between κ and the accuracy of $\mathbf{x}$.)

Any method of computing κ beginning with A appears to require at least n^3 additions and multiplications. The authors of LINPACK, in collaboration with J. H. Wilkinson, developed an algorithm for estimating κ (actually κ^{-1}) that requires only $3n^2 + O(n)$ additions and multiplications following the factorization produced by SGEFA.

Subroutine SGECO first computes $\|A\|$ as a maximum of the absolute sum norms of the columns of A. It then calls SGEFA to factorize A, and finally executes the LINPACK algorithm for condition number estimation, returning RCOND as its estimate of κ^{-1}. Let $C = RCOND^{-1}$. The hope is that C/κ will always be close to unity. It can be shown that with exact arithmetic one would always have $C/\kappa \leq 1$, but no theoretical lower bound is known. In a test with 1250 cases reported in [1], the lowest value that C/κ attained was 0.062, *i.e.* the condition number was slightly more than 16 times what was estimated.

D.6 Computing the determinant of A

SGED completes the computation of the determinant of A. In the factorization, $A = LU$, we have $Det(L) = \pm 1$ and U is triangular. Thus SGED computes $Det(A)$ by forming the product of the diagonal elements of U, and attaching a sign determined by analysis of the permutation record in IPVT(). This process may be represented as

$$d_0 = 1$$
$$d_i = \pm u_{i,i} d_{i-1}, \quad i = 1, \ \ldots, \ n$$

determinant $= d_n$

For a matrix of large order it is not uncommon for the determinant to be of extremely large or small magnitude. Consequently the LINPACK approach computes and stores the determinant as a pair of numbers, permitting a very large exponent range.

SGED differs, however, from the LINPACK subroutine SGEDI in the two-number representation selected. If all of the quantities, $|u_{i,i}|$ and $|d_i|$, in the above equations lie between the square roots of the underflow and overflow limits of the host computer system, the returned value of DET(2) will be zero. Thus in many problems of moderate order and having moderate scaling, DET(1) by itself will be the determinant value. Besides increasing the likelihood of DET(2) being zero, this modification reduces significantly the number of multiplications used for scaling.

D.7 Other types of matrices and problems handled in LINPACK

The subroutines described here represent only a small part of the full LINPACK collection. It is thus desirable to have LINPACK as well as the MATH 77 library. LINPACK provides functionalities analogous to those described here for the following special types of matrices:

General band	Hermitian indefinite
Positive definite	Hermitian indefinite packed
Positive definite packed	Triangular
Positive definite band	General tridiagonal
Symmetric indefinite	Positive definite tridiagonal
Symmetric indefinite packed	

LINPACK also provides subroutines for least-squares problems.

References

1. J. J. Dongarra, C. B. Moler, J. R. Bunch, and G. W. Stewart, **LINPACK Users' Guide**, Society for Industrial and Applied Mathematics, Philadelphia (1979) 320 pages.

E. Error Procedures and Restrictions

The LU factorization can be completed for a singular matrix, however the subroutines of this set are not capable of solving equations using a singular matrix. The inverse matrix does not exist for a singular matrix.

A singular matrix is indicated by the returned value of INFO being nonzero or RCOND being zero. In either of these cases the subroutines SGESLD, SGESLT, and SGEI cannot compute their usual outputs. If any of these latter three subroutines encounters a zero diagonal element in the U matrix, it will issue an error message using the subroutines of Chapter 19.2 with an error level of zero and return with incomplete results.

These subroutines all require N $\geq$ 1 and LDA $\geq$ N. Subroutines SGEFS and SGEFSC also require LDB $\geq$ N. These conditions are not checked and their violation causes unpredictable effects.

F. Supporting Information

The source language is ANSI Fortran 77.

LINPACK was developed as an NSF funded project from 1974 through 1976. This library of high-quality public-domain portable Fortran linear algebra software is widely used in the U.S. and throughout the world.

The subroutines described here were adapted from LINPACK to Fortran 77 for the JPL MATH77 library by C. L. Lawson and S. Y. Chiu, JPL, August 1987.

Entry	Required Files
CGECO	CAXPY, CDOTC, CGECO, CGEFA, CSCAL, CSSCAL, ICAMAX, SCASUM
CGED	AMACH, CGED
CGEFA	CAXPY, CGEFA, CSCAL, ICAMAX
CGEFS	CAXPY, CGEFA, CGEFS, CGESLD, CSCAL, ERFIN, ERMSG, ICAMAX
CGEFSC	CAXPY, CDOTC, CGECO, CGEFA, CGEFSC, CGESLD, CSCAL, CSSCAL, ERFIN, ERMSG, ICAMAX, SCASUM
CGEI	CAXPY, CGEI, CSCAL, CSWAP, ERFIN, ERMSG
CGESLD	CAXPY, CGESLD, ERFIN, ERMSG
CGESLT	CDOTC, CGESLT, ERFIN, ERMSG
DGECO	DASUM, DAXPY, DDOT, DGECO, DGEFA, DSCAL, IDAMAX
DGED	AMACH, DGED
DGEFA	DAXPY, DGEFA, DSCAL, IDAMAX
DGEFS	DAXPY, DGEFA, DGEFS, DGESLD, DSCAL, ERFIN, ERMSG, IDAMAX
DGEFSC	DASUM, DAXPY, DDOT, DGECO, DGEFA, DGEFSC, DGESLD, DSCAL, ERFIN, ERMSG, IDAMAX
DGEI	DAXPY, DGEI, DSCAL, DSWAP, ERFIN, ERMSG
DGESLD	DAXPY, DGESLD, ERFIN, ERMSG
DGESLT	DDOT, DGESLT, ERFIN, ERMSG
SGECO	ISAMAX, SASUM, SAXPY, SDOT, SGECO, SGEFA, SSCAL
SGED	AMACH, SGED
SGEFA	ISAMAX, SAXPY, SGEFA, SSCAL
SGEFS	ERFIN, ERMSG, ISAMAX, SAXPY, SGEFA, SGEFS, SGESLD, SSCAL
SGEFSC	ERFIN, ERMSG, ISAMAX, SASUM, SAXPY, SDOT, SGECO, SGEFA, SGEFSC, SGESLD, SSCAL
SGEI	ERFIN, ERMSG, SAXPY, SGEI, SSCAL, SSWAP
SGESLD	ERFIN, ERMSG, SAXPY, SGESLD
SGESLT	ERFIN, ERMSG, SDOT, SGESLT

DRSGEFSC

```
c      program DRSGEFSC
c>> 2001-05-22 DRSGEFSC Krogh Minor change for making .f90 version.
c>> 1996-05-28 DRSGEFSC Krogh Moved format up.
c>> 1994-10-19 DRSGEFSC Krogh  Changes to use M77CON
c>> 1994-08-09 DRSGEFSC WVS removed '0' from format
c>> 1992-03-18 DRSGEFSC CLL Added "c" to "program" line above.
c>> 1987-12-09 DRSGEFSC Lawson  Initial Code.
c--S replaces "?": DR?GEFSC, ?GEFSC, ?GEFS, ?MATP
c
c      Demo driver for SGEFSC
c     ----------------------------------------------------------------
      integer NMAX
      parameter (NMAX = 3)
      real              A(3,3), B(3,1), Z(NMAX), RCOND1
      integer I, J, IPVT(NMAX)
c
      data (A(1,J),J=1,3) / 0.579E0, -.394E0, 0.915E0 /
      data (A(2,J),J=1,3) / -0.795E0, 0.226E0, -0.868E0 /
      data (A(3,J),J=1,3) / 0.141E0, -0.329E0, -0.286E0 /
c
      data (B(I,1),I=1,3) / 5.873E0, -5.324E0, 1.069E0 /
c
 100  format(/' RCOND1 =',F7.4)
c     ----------------------------------------------------------------
      call SMATP(A,NMAX,NMAX,NMAX,'0 A(,) =')
      call SMATP(B,NMAX,NMAX,1,'0 B(,) =')
c
      call SGEFSC(A,NMAX,NMAX,B,NMAX,1,IPVT,RCOND1,Z)
c     call SGEFS(A,NMAX,NMAX,B,NMAX,1,IPVT,INFO)
c
      call SMATP(B,NMAX,NMAX,1,'0 SOLN(,) =')
      print 100, RCOND1
c
      end
```

ODSGEFSC

```
 A(,) =

                COL   1         COL   2         COL   3
ROW   1      0.5790000      -0.3940000       0.9150000
ROW   2     -0.7950000       0.2260000      -0.8680000
ROW   3      0.1410000      -0.3290000      -0.2860000

 B(,) =

                COL   1
ROW   1       5.873000
ROW   2      -5.324000
ROW   3       1.069000

 SOLN(,) =

                COL   1
ROW   1       1.999999
ROW   2      -5.000001
ROW   3       3.000000
```

RCOND1 = 0.0827