# 4.2 Linear Least-Squares and Covariance Matrix

## A. Purpose

The principal expected application of this set of subroutines is the solution of the linear least-squares problem, $A\mathbf{x} \simeq \mathbf{b}$, where $A$ is an $m \times n$ matrix with $m > n$, and $\mathbf{b}$ is an $m$-vector. The solution vector, $\mathbf{x}$, is an $n$-vector. Computation of the covariance matrix of the solution vector, $\mathbf{x}$, is also supported.

This software is not limited to the usual case of $m > n$ and $A$ being of full rank. A pseudoinverse solution is provided for all of the cases, $m > n$, $m = n$, and $m < n$, including the case of $A$ being rank-deficient. Furthermore, the right-side of the problem may be an $m \times k$ matrix, $B$, in which case the solution will be an $n \times k$ matrix, $X$.

## B. Usage

### B.1 Solving a Least-Squares Problem

#### B.1.a Program Prototype, Single Precision

**INTEGER LDA, M, N, LDB, NB, KRANK, IP**(≥N)

**REAL A**(LDA, ≥N)**, TAU, RNORM**(≥NB)**, WORK**(≥N)**, B**(LDB) or **B**(LDB, ≥ NB)

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

**CALL SHFTI(A, LDA, M, N, B, LDB, NB, TAU, KRANK, RNORM, WORK, IP)**

Computed quantities are returned in A(,), B(,), KRANK, RNORM(), and IP().

#### B.1.b Argument Definitions

**A(,)** [inout] On entry contains the M×N matrix, $A$, of the least-squares problem. Permit M > N, M = N, or M < N. On return contains an upper triangular matrix that can be used by subroutine, SCOV2, to compute the covariance matrix.

**LDA** [in] First dimensioning parameter for the array, A(,). Require LDA ≥ M.

**M** [in] Number of rows of data in A(,) and B(,) on entry. Require M ≥ 1.

**N** [in] Number of columns of data in A(,) on entry. Require N ≥ 0.

**B(,)** [inout] May be a singly or doubly subscripted array. On entry contains the right-side M-vector, $\mathbf{b}$, or (M×NB)-matrix, $B$, for the least-squares problem. On return contains the solution N-vector, $\mathbf{x}$, or (N×NB)-matrix, $X$.

**LDB** [in] First dimensioning parameter for the array B(,). Require LDB ≥ max(M, N) when NB ≥ 1, and LDB ≥ 1 when NB = 0.

**NB** [in] Number of right-side vectors for the least-squares problem. Require NB ≥ 0. If NB = 1, the array, B(,), may be either singly or doubly subscripted. If NB > 1, the array B(,) must be doubly subscripted. If NB = 0, this subroutine will not access B(,).

**TAU** [in] Absolute tolerance parameter provided by the user. Ideally should indicate the noise level of the data in the given matrix, $A$. The value, zero, is acceptable. Will be used in estimating the rank of $A$. Larger values of TAU will lead to a smaller estimated rank for $A$.

**KRANK** [out] Rank of $A$ estimated by the subroutine. Will be in the range, 0 ≤ KRANK ≤ min(M, N).

**RNORM()** [out] On return, RNORM(i) will contain the square root of sum of squares of residuals for the $i^{th}$ right-side vector.

**WORK()** [scratch] This array, of length at least N, is used internally by the subroutine as working space.

**IP()** [out] On return contains a record of column interchanges.

### B.2 Computing the Covariance Matrix

Subroutine SCOV2 is designed to be used following SHFTI; but only in cases in which KRANK determined by SHFTI is N.

#### B.2.a Program Prototype, Single Precision

**INTEGER LDA, N, IP**(≥N)**, IERR**

**REAL A**(LDA, ≥N)**, VAR**

On entry the values in A(,), LDA, N, and IP() should be the same as on return from a previous call to SHFTI. Also, assign a value to VAR.

**CALL SCOV2(A, LDA, N, IP, VAR, IERR)**

Computed quantities are returned in A(,) and IERR.

#### B.2.b Argument Definitions

**A(,)** [inout] On entry contains an N×N upper-triangular matrix produced by the subroutine, SHFTI. On return contains the upper-triangular part of the symmetric covariance matrix of the original least-squares problem. Elements in A(,) below the diagonal will not be referenced.

**LDA, N** [in] Must be the same as in a previous call to SHFTI.

**IP()** [in] Contains a record of column interchanges performed by a previous call to SHFTI.

**VAR** [in] Estimate of the variance of the data errors in the original right-side vector, **b**. To compute the covariance matrix for the $i^{th}$ right-side vector of the original problem, the user's code can compute VAR in terms of output quantities from SHFTI as

```
DOF = M - N
STDDEV = RNORM(i)/sqrt(DOF)
VAR = STDDEV**2
```

**IERR** [out] Error flag. Zero indicates no error was detected. A positive value indicates that A(IERR, IERR) was zero on entry. In this latter case no result will be produced.

### B.3 Modifications for Double Precision

Change the REAL type statements to DOUBLE PRECISION, and change the subroutine names from SHFTI and SCOV2 to DHFTI and DCOV2, respectively.

## C. Examples and Remarks

### C.1 A least-squares example

Data for a sample linear least-squares problem were generated by computing values of

$$y = 0.5 + 0.25\ \sin(2\pi x) + 0.125\ \exp(-x)$$

at eleven points, $x = 0.0$, 0.1, ..., 1.0, and rounding the resulting $y$-values to 4 decimal places, thus introducing errors bounded in magnitude by 0.00005. The program, DRSHFTI, uses SHFTI to compute a least-squares fit to this data using the model

$$c_1 + c_2 \sin(2\pi x) + c_3 \exp(-x)$$

and uses SCOV2 to compute the covariance matrix for the computed coefficients. Results are shown in the output file, ODSHFTI.

### C.2 Large problems

If M >> N and storage limitations make it awkward or impossible to allocate M×N locations for the array, A(,), one can use sequential accumulation of the rows of data to produce a smaller matrix to which SHFTI and SCOV2 can then be applied. See Chapter 4.4 for sequential accumulation.

### C.3 Underdetermined problems

If M < N, or, more generally, whenever Rank($A$) < N, there will be infinitely many vectors, **x**, that achieve the same minimal residual norm for the least-squares problem. For such cases SHFTI computes the pseudoinverse solution, *i.e.* the solution vector, **x**, of least Euclidean norm.

### C.4 Computing the pseudoinverse matrix

If one sets B(,) to be the M×M identity matrix, the resulting N×M solution matrix, $X$, will be the pseudoinverse matrix of $A$.

### C.5 Reliability issues regarding KRANK

KRANK must be understood as an estimate of the rank of $A$ that depends not only on $A$, but on the user's setting of TAU and the particular algorithm used in SHFTI. A small change in the value of TAU could result in a different value being assigned to KRANK, which in turn could result in a large change in the solution vector.

Although this subroutine will produce a solution whatever the value of KRANK, the occurrence of KRANK < min(M, N) should be regarded as exceptional. The user should investigate such instances as it could be due to a programming error or a very ill-conditioned model. Singular Value Analysis (see Chapter 4.3) can be useful in analyzing an ill-conditioned model.

## D. Functional Description

This software is an adaptation to Fortran 77 of the subroutine, HFTI, given and described in [1]. The name, HFTI, denotes Householder Forward Transformation with column Interchanges.

To avoid nonessential complications, we shall describe the algorithm only for the case of M ≥ N and NB = 1. A sequence of up to N Householder orthogonal transformations is applied from the left, with column interchanges, the total effect of which may be summarized by the equation

$$Q\,[A : \mathbf{b}] \begin{bmatrix} P & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R & \mathbf{g} \\ 0 & \mathbf{h} \end{bmatrix}$$

where Q is the M×M product of the Householder matrices, $P$ is an N×N permutation matrix accounting for the column interchanges, $R$ is an N×N upper-triangular matrix with diagonal elements in order of decreasing magnitudes, **g** is an N-vector, and **h** is an (M − N)-vector.

If all diagonal elements of $R$ exceed TAU in magnitude, the solution, **x**, is computed by solving $R\mathbf{x} = \mathbf{g}$. The subroutine sets KRANK = N and RNORM(1) = $\|\mathbf{h}\|$.

Alternatively, if the diagonal elements of $R$ beyond position $i$ are less than TAU, the subroutine sets KRANK

$= i$. Let $[R_1 : \mathbf{g}_1]$ denote the first KRANK rows of $[R : \mathbf{g}]$, and let $\mathbf{g}_2$ denote the last N − KRANK components of $\mathbf{g}$. The subroutine applies up to KRANK Householder transformations to $R_1$ from the right, effecting the transformation

$$R_1 K = [W : 0]$$

where $K$ is the N×N product of Householder transformations and $W$ is a KRANK×KRANK non-singular upper-triangular matrix. A KRANK-vector, $\mathbf{y}_1$, is computed by solving $W\mathbf{y}_1 = \mathbf{g}_1$. An N-vector, $\mathbf{y}$, is formed by appending zeros to the end of $\mathbf{y}_1$, and the minimal length solution vector, $\mathbf{x}$, is computed as $\mathbf{x} = \mathrm{PK}\mathbf{y}$. RNORM(1) is computed as the Euclidean norm of the (M − KRANK)-vector formed by concatenating $\mathbf{g}_2$ and $\mathbf{h}$.

Subroutine, SCOV2, is intended for use only when Rank($A$) = N. Algorithm, COV, from [1] is used. The covariance matrix is traditionally defined as $C = \sigma^2(A^tA)^{-1}$, where $\sigma^2$ is the variance of the data error. Using the triangular matrix, $R$, and the (orthogonal) permutation matrix, $P$, defined above, one can also write $C = \sigma^2(PR^tRP^t)^{-1} = \sigma^2PR^{-1}R^{-t}P^t$, where $R^{-t}$ denotes the transpose of $R^{-1}$. Thus, SCOV2 computes

$$E = R^{-1}$$
$$F = EE^t$$
$$C = \sigma^2PFP^t$$

**References**


1. Charles L. Lawson and Richard J. Hanson, **Solving Least-Squares Problems**, Prentice-Hall, Englewood Cliffs, N. J. (1974) 340 pages.


## E. Error Procedures and Restrictions

In SHFTI (or DHFTI) an error message will be issued and an immediate return will be made, setting KRANK = 0, if any of the following conditions are noted:

$$\mathrm{M} < 1,\ \mathrm{N} < 0,\ \mathrm{LDA} < \mathrm{M},\ \text{or LDB} < \max(\mathrm{M},\ \mathrm{N})$$

Most commonly, on return KRANK will have the value min(M, N). A smaller value of KRANK may be valid, but unless the user has reason to expect this possibility it is likely to be due to a usage error.

In SCOV2 (or DCOV2) the N diagonal elements of A(,) must be nonzero on entry. If so, IERR is set to zero. If not, IERR is set to the index of the first zero element, an error message will be issued, and a return will be made with the computation being incomplete.

## F. Supporting Information

The source language is ANSI Fortran 77.

| Entry | Required Files |
|---|---|
| **DCOV2** | DCOV2, DDOT, DSWAP, ERFIN, ERMSG, IERM1, IERV1 |
| **DHFTI** | AMACH, DAXPY, DDOT, DHFTI, DHTCC, DHTGEN, DNRM2, ERFIN, ERMOR, ERMSG, IERM1, IERV1 |
| **SCOV2** | ERFIN, ERMSG, IERM1, IERV1, SCOV2, SDOT, SSWAP |
| **SHFTI** | AMACH, ERFIN, ERMOR, ERMSG, IERM1, IERV1, SAXPY, SDOT, SHFTI, SHTCC, SHTGEN, SNRM2 |

Adapted to Fortran 77 from [1] by C. L. Lawson and S. Y. Chiu, JPL, May 1986, June 1987.

# DRSHFTI

```
c      program DRSHFTI
c>> 2001-05-22 DRSHFTI  Krogh Minor change for making .f90 version.
c>> 1996-07-03 DRSHFTI  Krogh  Special code for C conversion.
c>> 1994-10-19 DRSHFTI  Krogh  Changes to use M77CON
c>> 1987-12-09 DRSHFTI  Lawson  Initial Code.
c                       Demo driver for SHFTI and SCOV2
c--S replaces "?": DR?HFTI, ?HFTI, ?COV2
c
c      The sample data was computed as
c      y = 0.5 + 0.25 * sin(2*pi*x) + 0.125 * exp(-x)
c      rounded to four decimal places.
c      -------------------------------------------------------------
c++ Code for .C. is inactive
c%%   long int k;
c++ End
      integer MMAX, NMAX
      real            ZERO, ONE, TWO, FOUR
```

```
      parameter(MMAX=11, NMAX=3)
      parameter(ZERO = 0.0E0, ONE = 1.0E0, TWO = 2.0E0, FOUR = 4.0E0)
      real              X(MMAX),Y(MMAX),A(MMAX,NMAX),C(MMAX)
      real              WORK(NMAX), RNORM(1)
      real              DOF, PI, STDDEV, TAU, VAR
      integer           IP(NMAX), KRANK, I, IERR, J, NC, M, N
c     ------------------------------------------------------------------
      data X / 0.0E0,   0.1E0,   0.2E0,   0.3E0,   0.4E0,   0.5E0,
     *                  0.6E0,   0.7E0,   0.8E0,   0.9E0,   1.0E0/
      data Y / 0.6250E0,0.7601E0,0.8401E0,0.8304E0,0.7307E0,0.5758E0,
     *                0.4217E0,0.3243E0,0.3184E0,0.4039E0,0.5460E0/
      data M, N, NC, TAU/ MMAX, NMAX, 1, 0.0E0/
c     ------------------------------------------------------------------
      PI = FOUR * atan(ONE)
      do 20 I = 1, M
         A(I,1) = ONE
         A(I,2) = sin(TWO * PI * X(I))
         A(I,3) = exp(-X(I))
         C(I) = Y(I)
   20 continue

      call SHFTI (A,MMAX,M,N,C,MMAX,NC,TAU,KRANK,RNORM,WORK,IP)
      DOF = M - N
      STDDEV = RNORM(1) / sqrt(DOF)
      VAR = STDDEV**2
      print '(1x,''Rank of linear system   ='',i4)', KRANK
      print '(1x,''Std. Dev. of data error ='',f10.6)', STDDEV
      print '(1x,''Solution coefficients   ='',3f10.6)', (C(J),J=1,N)

      call SCOV2( A, MMAX, N, IP, VAR, IERR)
      print '(1x,''Error flag from SCOV2   ='',i4)', IERR
      print '('' Covariance matrix of computed coefficients:'')'
      print '(1X)'
c++ Code for ~.C. is active
      do 30 I = 1,N
         print '(1x,3(3x,2i3,g16.8))', (I,J,A(I,J),J=I,N)
   30 continue
c++ Code for .C. is inactive
c%%   for (i = 0; i < n; i++){
c%%      for (j = i; j < n; j+=3){
c%%         for (k = j; k < (j < n - 3 ? j+3 : n); k++)
c%%               printf( "   %3ld%3ld%16.8g", i+1, k+1, a[k][i] );
c%%         printf( "\n" );}
c%%      }
c++ End
      stop
      end
```

## ODSHFTI

```
 Rank of linear system   =    3
 Std. Dev. of data error =  0.000030
 Solution coefficients   =  0.500004  0.249999  0.125007
 Error flag from SCOV2   =    0
 Covariance matrix of computed coefficients:
```

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0.15022650E−08 | 1 | 2 | 0.42248927E−09 | 1 | 3 | −0.22285622E−08 |
| 2 | 2 | 0.30603203E−09 | 2 | 3 | −0.66292566E−09 | | | |
| 3 | 3 | 0.34968251E−08 | | | | | | |