NMRSIM_F.LIB
Version 1.1, July 1993
A Library of Fortran Routines for Simulating Multivariate Normal Rectangle Probabilities and Their Derivatives

by

©Vassilis Argyrou Hajivassiliou

(with the aid of Yoosoon Chang)

Department of Economics, London School of Economics

1 Introduction

A companion to this code is the paper **Simulation of Multivariate Normal Rectangle Probabilities and Their Derivatives: Theoretical and Computational Results** by V.A. Hajivassiliou, London School of Economics, D.L. McFadden, University of California at Berkeley, and P.A. Ruud, University of California at Berkeley, Journal of Econometrics, 1997.

The simulation methods presented in this paper have been coded in GAUSS and in FORTRAN. Both versions of the programs are available from the author's Web page through the URL: `econ.lse.ac.uk/~vassilis/pub/simulation/fortran`, directly from the author upon request, or by anonymous `FTP` from the site `ariadne.lse.ac.uk:pub/simulation/f` Instructions are given in Section 3.

Each simulator procedure requires the following standard inputs; the interpretation of some inputs may vary from routine to routine, and not all are used in all routines:

`M`	Dimension of the multivariate normal
`VMU`	Mean of multivariate normal, an $\mathtt{M} \times 1$ vector
`W`	Covariance matrix of multivariate normal, an $\mathtt{M} \times \mathtt{M}$ array
`WI`	Inverse of `W`
`C`	Lower triangular Choleski factor of `W`, an $\mathtt{M} \times \mathtt{M}$ array
`A`	Lower bound of rectangle, an $\mathtt{M} \times 1$ vector. {When the lower bound is $-\infty$, set $\mathtt{A} = (-\mathtt{1.0E10}) * \mathtt{ONES(M,1)}$.}
`B`	Upper bound of rectangle, an $\mathtt{M} \times 1$ vector. {When the upper bound is ∞, set $\mathtt{B} = (\mathtt{1.0E10}) * \mathtt{ONES(M,1)}$.}
`NR`	Number of repetitions
`U`	Random variates, an $\mathtt{M} \times \mathtt{R}$ array
`PARM`	Parameters and constants for the simulation

All the simulators return {`P`,`HU`,`HC`}, where `P` is the scalar rectangle probability, `HU` is the $(\mathtt{M}+1) \times (\mathtt{M}+1)$ array of unconditional partial moments (6), and `HC` is the $(\mathtt{M}+1) \times (\mathtt{M}+1)$ array of conditional moments (7). Parts of the output not provided by a simulator are set to -999.

In the FORTRAN implementation, the two additional inputs `MMAX` and `NRMAX` specify the maximum values of `M` and `NR` allocated at compilation time.

The Fortran call is of the form

```
CALL XXX(MMAX,NRMAX,M,VMU,W,WI,C,A,B,NR,U,PARM,P,HU,HC)
```

where `XXX` stands for the 3 or 4 letter mnemonic for the simulator in question, as follows:

AUS Approximate Unbiased Simulator

ARSE Acceptance/Rejection Simulator, Exponential comparison distribution

ARSR Acceptance/Rejection Simulator, Recursive truncated normal comparison distribution

CFS Crude Frequency Simulator

DCS Deak chi-square Simulator

GHK Geweke-Hajivassiliou-Keane Simulator

GSS Gibbs Sampler Simulator

NISE Normal Importance Sampling Simulator, Exponential comparison distribution

NIST Normal Importance Sampling Simulator, Truncated normal comparison distribution

PCF Parabolic Cylinder Function Simulator

PKFS Polynomial Kernel Smoothed Frequency Simulator

SDS Stern Decomposition Simulator

SUS Sequential Unbiased Simulator

The programs include code for all statistical functions, spherical transformations, and antithetics routines that are required by the simulation algorithms, and hence are self-contained.

2 Installation

2.1 Description of Files

6 files constitute the main program:

1. `har_main.f` A harness program to set up monte carlo test runs.
2. `har_libd.f` Machine-dependent routines, necessary for timing the results. Two versions are given, one for UNIX, and one for PC/Lahey Fortran.
3. `har_libm.f` Matrix operation routines.
4. `har_libo.f` Other utility routines.
5. `har_libs.f` The basic Simulation routines, following the GAUSS implementation in terms of calling arguments, etc.
6. `har_libt.c` (required for UNIX) This is a C routine that interfaces with the system clock, which is necessary for the timings.
   A 7th file is required, defining the maximum dimensions:
7. `hdim.inc` Before compiling, you must change this dimensioning file for the maximum number of elements in the multivariate normal (`mmaxim`), and for the maximum number of simulations allowed (`nrmaxim`).

NB: The program will search automatically for the environment variable '`machine`' and will make a note in the output of the computer used. (On UNIX, the environment variable '`machine`' must be lower case – on DOS, the case is not relevant.)

2.2 Instructions for UNIX users

To prepare the executable files "harness" and "asd2asc", run the script file "har_make" by issuing the command "source har_make".

2.3 Results of a Run

The results will be placed in files `xxMMMM.asd`, where `xx` is a two-letter name specified by the user at run time, and `MMMM` is the 3 or 4 letter identifier for the simulation algorithm (e.g., `CFS`, `ARSE`, etc.). The `*.asd` files are "direct-access" files, which are not easily readable. To change them into usual ASCII files, run the program "asd2asc". Each `*.asc` file contains the "true" (usually quadrature) values for the quantities to be simulated, and the results of each Monte Carlo repetition on a new line.

2.4 Analyzing Restults in GAUSS:

The program will create automatically the file '`xxmkgau.bat`', where `xx` is the two-letter name specified by the user at run time. When run on a DOS machine with the GAUSS utility ATOG on the path, this batch file will automatically translate the `*.asc` result files created as described above to GAUSS `.DAT/.DHT` data sets. This will facilitate the analysis of the Monte Carlo results of a particular run using GAUSS.

3 Obtaining the Simulation Code by Anonymous FTP

```
\centerline{\Large \bf Obtaining Simulation Code by Anonymous FTP}
\begin{verbatim}
                                                V.A. Hajivassiliou
                                                LSE
                                                email: vassilis@lse.ac.uk
                                                January 1998

ftp site: ariadne.lse.ac.uk
userid: anonymous
password: your email address
directory: pub/simulation

There are three subdirectories under this:
fortran -- contains the fortran code for the 13 simulation routines described
           in Hajivassiliou, McFadden, and Ruud, 1997, "Simulating Normal
           Rectangle Probabilities and their Derivatives: Theoretical and
           Computational Results," Journal of Econometrics
gauss   -- contains the gauss code for the 13 simulation routines described
           in Hajivassiliou, McFadden, and Ruud, 1997, op.cit.
ssml    -- contains actual estimation code using smoothly simulated maximum
           likelihood for various models (see Boersch-Supan and
           Hajivassiliou, 1993, "Smooth Simulators for Normal MNP Choice
           Probabilities" Journal of Econometrics, August)
           Code for specific models is kept under separate subdirectories.
           Currently, only code for the Multinomial Probit model is available,
           which can be found in directory pub/simulation/ssml/mnp.
           This program was written by Axel Boersch-Supan of the University
           of Mannheim.  Vassilis Hajivassiliou ported it to UNIX and added
           several enhancements.

For example, to get the file "simprog.gcf" containing the gauss code,
you should issue the following:

ftp ariadne.lse.ac.uk
```

```
anonymous
whoever@whereever
cd pub/simulation/gauss
get simprog.gcf
quit
```

The PostScript code for some of my papers is kept in the directory

`pub/papers/ps`

and the Adobe Acrobat (PDF) code for them in directory

`pub/papers/hplj`

4 Description of Procedures in Library

```
      program harness
c================================================================c
c
c   A. Overview
c      This section contains a series of FORTRAN procedures  for
c simulation  of multivariate normal rectangle probabilities, and  the
c derivatives  of  these probabilities with respect to the mean and
c covariances of the normal  distribution.  Table 2 indexes the
c procedures in this section.
c
c                           TABLE 2.  FORTRAN PROCEDURES
c
c      SECTION   MNEUMONIC*             DESCRIPTION
c
c         H        CFS          Crude Frequency Simulator
c         I        NIS          Normal Importance Sampling Simulator
c                               Procs NISE and NIST use exponential and
c                               truncated normal comparison
c                               distributions, respectively
c         J        KFS          Polynomial Kernel Smoothed Frequency
c                               Simulator.  The proc is PKFS
c         K        SDS          Stern Decomposition Simulator
c         L        GHK          Geweke-Hajivassiliou-Keane Simulator
c         M        PCF          Parabolic Cylinder Function Simulator
c         N        DCS          Deak chi-square Simulator
c         O        ARS          Acceptance/Rejection Simulator
c                               Procs ARSE and ARSR use exponential and
c                               recursive truncated normal comparison
c                               distributions, respectively
c         P        GSS          Gibbs Sampler Simulator
c         Q        AUS          Approximate (OR Sequential) Unbiased
c                               Simulator
c         R        SUS          Sequential Unbiased Simulator
c         S        LIB_A        Statistical Functions
c         T        LIB_B        Spherical Transformations and
```

```
c                               Antithetics
c         U        LIB_C        Miscellaneous Utilities
c
c * The procs are located in libraries named *.G, where '*' denotes the
c mneumonic.
c
c      Each of the simulator procedures requires the following standard
c inputs; the interpretation of some inputs may vary from routine to
c routine, and not all are used in some routines:
c
c      M         Dimension of the multivariate normal
c      MU        Mean of multivariate normal, a M x 1 vector
c      W         Covariance matrix of multivariate normal, a M x M
c                array
c      WI        Inverse of W
c      C         Lower triangular Cholesky factor of W, a M x M array
c      A         Lower bound of rectangle, a M x 1 vector [When the
c                lower bound is -infinity, set A = (-1.0E10)*ONES(M,1)]
c      B         Upper bound of rectangle, a M x 1 vector [When the
c                upper bound is +infinity, set B = (+1.0E10)*ONES(M,1)]
c      R         Number of repetitions
c      U         Random variates, a M x R array
c      PARM      Parameters and constants for the simulation
c
c The simulators all return {P,HU,HC}, where P is the scalar rectangle
c probability, HU is the (M+1) x (M+1) array of unconditional partial
c moments (6), and HC is the (M+1) x (M+1) array of conditional moments
c (7).  Parts of the output not provided by a simulator are set to
c -999.
c
c Interpretations of input parameters
c -----------------------------------
c ARSR,ARSE
c ---------
c These procs assume that PARM[.,1] contains seeds for the uniform
c random number generator RNDUS called by the routines; there is one
c seed for each repetition R.  PARM[R+1,1] is assumed to contain a
c censoring limit for rejection.
```

```
c ARSR and ARSE use  U = UUBIG;                              (uniforms)
c
c AUS
c ---
c The proc assumes that PARM(.,1) contains a vector of R seeds for the
c normal random number generator RNDNS, PARM(1,2) contains the number
c of initial GHK simulators of P used in the simulation of 1/P,
c PARM(2,2) contains the maximum number of trials using the crude
c frequency simulator that are made before the process is censored.  If
c PARM(2,2) is made large (say 1.e50), then this is computationally the
c same as the Sequential Unbiased Simulator (SUS).  PARM(3,2) contains
c a seed for RNDUS.
c AUS uses  U = UUBIG[.,1:R];                                (uniforms)
c
c CFS
c ---
c This proc does not use PARM.
c CFS uses  U = UNBIG[.,1:R]                         (standard normals)
c
c DCS
c ---
c The routine assumes that PARM(1,1) = DET(W).
c DCS uses  U = USBIG[.,1:R];       (random normal grid on unit sphere)
c
c GHK
c ---
c This proc does not use PARM.
c GHK uses  U = UUBIG[.,1:R];                                (uniforms)
c
c GSS
c ---
c It assumes that PARM[1,1] gives the number of rounds to be used in
c the construction of the random draws.
c GSS uses  U = UUBIG[.,1:R];                                (uniforms)
c
c NISE,NIST
c ---------
c These procs do not use PARM.
```

```
c NISE and NIST use  U = UUBIG[.,1:R];                        (uniforms)
c
c PCF
c ---
c The routine assumes that PARM(1,1) = DET(W).
c PCF uses  U = USSBIG[.,1:R]; (random normal grid on unit sphere and
c                              orthant)
c
c PKF
c ---
c This proc does not use W.
c It assumes that PARM[1,1] contains a window width parameter.
c KFS uses  U = UNBIG[.,1:R];                        (standard normals)
c
c SDS
c ---
c The SDS proc assumes that PARM[1,1] contains a scalar l > 0 such that
c W-lI is positive definite.
c SDS uses  U = UNBIG[.,1:R];                        (standard normals)
c Non-standard feature:
c --------------------
c SDS assumes that C is a Cholesky factor of W-lI, not of W.
c
c SUS
c ---
c This procedure is the same the Asymptotically Unbiased Simulator
c (AUS) with PARM(2,2) made large (1.e50).
c The proc assumes that PARM(.,1) contains a vector of R seeds for the
c normal random number generator RNDNS, PARM(1,2) contains the number
c of initial GHK simulators of P used in the simulation of 1/P,
c PARM(2,2) contains the maximum number of trials using the crude
c frequency simulator that are made before the process is censored.
c This is made very large.
c PARM(3,2) contains a seed for RNDUS.
c SUS uses  U = UUBIG[.,1:R];                               (uniforms)
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c
```

```
c      Test Harness
c      ------------
c      The following program provides a simple harness for testing the
c procs above.  A one-factor model is used to generate probabilities,
c with gaussian quadrature used for evaluation.  The accuracy of this
c quadrature can be checked against symmetric cases where the
c probabilities are known.
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine timdat(itim,idat)
c================================================================c
c
c       set of routines to do timing on a unix machine
c
c       vassilis hajivassiliou 14:00:48.40, sun, jun 3, 1990
c
c      1.subroutine timdat(itim,idat)
c         itim(1)=hours, itim(2)=minutes, itim(3)=secs,
c         itim(4)=1/100th secs, itim(5)=1/100th-secs-since-jan1,1989
c         idat(1)=day, idat(2)=month, idat(3)=year (two digits)
c      2.double precision function sec()
c         number of seconds since a fixed point, typically midnight
c      3.double precision function cpusec()
c         number of seconds of cpu since previous invocation
c  1,2, and 3 are system dependent routines, called by others
c
c      4.double precision function seed(iseed)
c         calls the system clock to obtain a seed; returned also as i*4
c      5.subroutine times(nfile)
c         prints timing information (since last invocation if re-called)
c      6.subroutine when(nfunit)
c         prints date and time
c      7.double precision function hsec()
c         number of 1/100th secs since fixed point (typically midnight)
c************************************************************************
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c================================================================c
c sec.unx
```

```
c
c for c-interfaced programs, one for sec and one for secmic,
c see file csecond.unx
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

      subroutine getevar(cvname,cvvalue)
c==================================================================c
c
c     vah 8:34:21.59, Tue, Mar 24, 1992
c     UNIX version
c
c     cvvalue must be a character variable declared in the calling
c     program.
c
c     if cvname is not set, cvvalue will be '?' upon return.
c
c     requires subprograms system, icoccfst, and ltrim
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine fsize(fname,nbytes)
c     Returns size of file fname in bytes.
c     Returns -1 if file does not exist
c
c     unix version
c
c     vah 18:33:12.14, Sat, May 12, 1990
c
c     calls ltrim, cexist
c     and unix function stat(fname,istat13)  ??? sun-specific ???
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine cbind(x,ldx,m,nx,y,ldy,ny,res,ldres)
c==================================================================c
c
c-----This routine takes two input matrices x and y with the following
c     dimensions and performs a horizontal concatenation on them.
c
c      x(ldx,m,nx), where ldx is the leading dimension
c      y(ldy,m,ny), where ldy is the leading dimension
c
```

```
c     The input matrices x and y must have the same number of rows.
c     The number of columns in the resulting matrix res is the sum
c     of nx and ny.
c     The horizontally concatenated matrix res(ldres,m,nx+ny) is
c     then returned.
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine chol(x,ldx,n,chx,ldchx)
c==================================================================c
c
c     This routine is based on the (double precision) subroutine DCHDC
c     in DLINP.
c
c-----chol computes the decomposition of a positive matrix x with ldx
c     leading dimension and n effective dimension, and returns the
c     lower triagular cholesky factor of x.
c
c     on entry ------ x(ldx, n), a positive matrix
c     on exit  ------ x:   the upper cholesky factor of x
c                     chx: the lower cholesky factor of x
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine diag(x,ldx,iorder,res,ldres)
c==================================================================c
c
c-----diag extracts diagnol elements from a (m by n) matrix x, and put
c     it into a vector res, which has iorder elements.
c     iorder = min(m,n)
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine diagrv(x,ldx,iorder,nx,vec,ldvec)
c==================================================================c
c
c-----diagrv replaces the diagnol elements of a (m by n) matrix x with
c     leading dimension ldx with a vector, vec(ldvec).
c     iorder = min(m,n)
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine eye(eyemat,ldx,iorder)
c==================================================================c
c
```

```
c-----eye creates "iorder" dimensional identity matrix with a leading
c     dimension ldx.
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine hadcomp(x,ldx,y,ldy,res,ldres,mx,nx,coper)
c==============================================================c
c
c-----hadcomp does element by element comparision of the matrices x and
c     y with the same dimensions (mx by nx) according to the coper code,
c     and stores the resulting values into the res matrix of the same
c     dimesion.
c     ldx,ldy,and ldres are the leading dimensions for x,y, and res
c     respectively.
c
c          coper         logical operation
c          -----         ------------------
c          'gt'               >
c          'ge'               >=
c          'eq'               =
c          'lt'               <
c          'le'               <=
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

      subroutine hadoper(x,ldx,y,ldy,res,ldres,mx,nx,ioper)
c==============================================================c
c
c-----hadoper does element by element arithmetic operations of the
c     matrices x and y with the same dimensions (mx by nx) according to
c     the "ioper" code,and stores the resulting values into the res
c     matrix of the same dimesion.
c     ldx,ldy,and ldres are the leading dimensions for x,y, and res
c     respectively.
c
c     ioper         operation
c     -----------------------
c       1             +
c       2             -
c       3             *
c       4             /
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
      function intfloor(x)
      integer*4 intfloor
      double precision x
c=================================================================c
c
c     intfloor takes double precision number x and truncates fractional
c     part down toward negative infinity. So, on exit, x has a
c     integer value.
c
c     e.g.) intfloor(-14.10d0) = -15
c           intfloor(4.99d0) = 4
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine matcopy(y,ldy,x,ldx,mstart,mend,nstart,nend)
c=================================================================c
c
c-----matcopy copies a matrix x[mstart=mend,nstart=nend] with ldx
c     leading dimesion to a matrix y with the dimensions (mend-mstart+1)
c     by (nend-nstart+1) and leading dimension ldy.
c
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine maxc(x,ldx,mx,nx,res,ldres)
c=================================================================c
c
c     maxc is based on the double precision function FMAX in VFLIB.
c
c-----maxc takes the maximal value of each column of a matirx x(mx,nx)
c     with a leading dimension ldx, and stores them in res(nx).
c     The resulting vector res(nx) with a leading dimension ldres
c     is then returned.
c
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine mcdfn(x,ldx,mx,nx,res,ldres)
c=================================================================c
c
c-----mcdfn computes the cumulative distribution function(cdf) at each
c     element of the input matrix x(mx,nx) with a leading dimension ldx,
c     using the double precision function PHEEV from VFLIB.
c     Res(ldres,nx) stores the resulting values and is returned.
```

```
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine mcdfni(x,ldx,mx,nx,res,ldres)
c==================================================================c
c
c     This routine is based on the double precision function PHINV
c     in the VFLIB.
c
c-----mcdfn takes a (mx by nx) probability matrix x with a leading
c     dimension ldx and calculates, for each element of x,
c     the normal inverse cdf(quantile). Res(ldres,nx) stores the
c     resulting values and is returned.
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine mdscal(x,ldx,res,ldres,mx,nx,scale)
c==================================================================c
c
c-----mdscal scales a matrix x(mx,nx) with a leading dimesion ldx,
c     by the value of the given scale.
c     Res(ldres,nx) stores the scaled values of x, and is returned.
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine meanc(x,ldx,mx,nx,res,ldres)
c==================================================================c
c
c-----meanc takes a matrix x(mx,nx) with a leading dimension ldx, and
c     computes the mean of every column of the matrix.
c     Res(nx) with a leading dimensin ldres stores the mean of each
c     column and is returned.
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine minc(x,ldx,mx,nx,res,ldres)
c==================================================================c
c
c     minc is based on the double precision function FMIN from VFLIB.
c
c-----minc takes the minimum value of each column of a matrix x(mx,nx),
c     with a leading dimension ldx, and stores them in res(nx) with a
```

```
c     leading dimension ldres. The resulting vector res(ldres) is then
c     returned.
c
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine mrnorm(x,ldx,mx,nx,iseed)
c================================================================c
c
c     This routine is based on the double precision function RANORM in
c     the VFLIB.
c
c-----mrnorm creates and returns a (mx by nx) matrix of standard normal
c     random numbers with a leading dimension ldx, using a seed given
c     by the argument iseed.
c
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine mrunif(x,ldx,mx,nx,iseed)
c================================================================c
c
c-----mrunif generates and returns a matrix x(mx,nx) with a leading
c     dimension ldx, of uniform random numbers on the interval [0,1].
c     The argument iseed is given to the function RANDU from VFLIB.
c     The argument iseed is the current random number generator seed
c     and RANDU updates it.
c
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine packr(x,ldx,m,n,codemis,icount)
c================================================================c
c
c     packr takes m by n matrix with ldx leading dimension and checks
c     each element of x for codemis. If any element of a row matches
c     with the codemis, the row is deleted.
c     On exit, the matrix x is free of elements=codemis)
c
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine prodc(x,ldx,mx,nx,res,ldres)
c================================================================c
c
c returns products of columns.
```

```
c
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine rbind(x,ldx,mx,n,y,ldy,my,res,ldres)
c================================================================c
c
c     rbind takes two input matrices of the following dimension and
c     performs a vertical concatenation on them.
c
c      x(ldx,mx,n), where ldx is the leading dimension
c      y(ldy,my,n), where ldy is the leading dimension
c
c     The input matrices x and y must have the same number of columns.
c     The number of rows in the resulting matrix res is the sum
c     of mx and my.
c     The vertically concatenated matrix res(ldres,mx+my,n)is then
c     returned.
c
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine stdc(x,ldx,mx,nx,res,ldres)
c================================================================c
c
c-----stdc takes a matrix x(ldx,mx,nx) where ldx is the leading
c     dimension of x and computes the standard deviation of the
c     elements in each column of x.
c     The vector res(nx) with a leading dimension ldres stores the
c     resulting standard deviations and is returned.
c
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

      subroutine sumc(x,ldx,mx,nx,res,ldres)
c================================================================c
c
c-----sumc takes a matrix x(mx,nx) with a leading dimension ldx, and
c     computes the sum of every column of the matrix.
c     The resulting vector res(nx) with a leading dimension ldres
c     stores the sums of each column and is returned.
c
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine transps(x,ldx,mx,nx,res,ldres)
```

```
c================================================================c
c
c-----transps transposes a (mx by nx) matrix with leading dimension ldx.
c     It returns the transposed(nx by mx) matrix res with leading
c     dimension ldres.
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine cexist(cfile,nrc)
c================================================================c
c     vah 3:29 am, monday, october 27, 1986
c
c     query the existence of a file using the inquire statement
c     nrc=0 if file exists, = -1 if not
c
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      double precision function chicdf(cs,df,ier)
c================================================================c
c
c   constructed from the
c
c   imsl routine name   - mdch
c
c   by vah, 16:28:44.31, tue, jul 30, 1991
c
c   changed into full double precision.  calls to mdnor replaced by
c   calls to pheev
c
c------------------------------------------------------------------------
c
c   computer            - ibm77: double
c
c   latest revision     -  16:30:02.36, tue, jul 30, 1991
c
c   purpose             - chi-squared probability distribution function
c
c   usage               - p=chicdf(cs,df,ier)
c
c   arguments    cs     - input value for which the probability is
c                           computed. cs must be greater than or equal
```

```
c                            to zero.
c                df     - input number of degrees of freedom of the
c                            chi-squared distribution. df must be greater
c                            than or equal to .5 and less than or equal
c                            to 200,000.
c                ier    - error parameter. (output)
c                          terminal error
c                            ier = 129 indicates that cs or df was
c                              specified incorrectly.
c                          warning error
c                            ier = 34 indicates that the normal pdf
c                              would have produced an underflow.
c
c   returns:     p      - output probability that a random variable
c                            which follows the chi-squared distribution
c                            with df degrees of freedom is less than or
c                            equal to cs.
c   precision/hardware  - double/all
c
c   reqd. routines      - pheev, dgamma
c
c-----------------------------------------------------------------------
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine daxpy(n,da,dx,incx,dy,incy)
c================================================================c
c
c     constant times a vector plus a vector.
c     uses unrolled loops for increments equal to one.
c     jack dongarra, linpack, 3/11/78.
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine dchdc(a,lda,p,work,jpvt,job,info)
c================================================================c
      integer lda,p,jpvt(1),job,info
      double precision a(lda,1),work(1)
c
c     dchdc computes the cholesky decomposition of a positive definite
```

```
c     matrix.  a pivoting option allows the user to estimate the
c     condition of a positive definite matrix or determine the rank
c     of a positive semidefinite matrix.
c
c     on entry
c
c         a       double precision(lda,p).
c                 a contains the matrix whose decomposition is to
c                 be computed.  onlt the upper half of a need be stored.
c                 the lower part of the array a is not referenced.
c
c         lda     integer.
c                 lda is the leading dimension of the array a.
c
c         p       integer.
c                 p is the order of the matrix.
c
c         work    double precision.
c                 work is a work array.
c
c         jpvt    integer(p).
c                 jpvt contains integers that control the selection
c                 of the pivot elements, if pivoting has been requested.
c                 each diagonal element a(k,k)
c                 is placed in one of three classes according to the
c                 value of jpvt(k).
c
c                    if jpvt(k) .gt. 0, then x(k) is an initial
c                                       element.
c
c                    if jpvt(k) .eq. 0, then x(k) is a free element.
c
c                    if jpvt(k) .lt. 0, then x(k) is a final element.
c
c                 before the decomposition is computed, initial elements
c                 are moved by symmetric row and column interchanges to
c                 the beginning of the array a and final
c                 elements to the end.  both initial and final elements
```

```
c                are frozen in place during the computation and only
c                free elements are moved.  at the k-th stage of the
c                reduction, if a(k,k) is occupied by a free element
c                it is interchanged with the largest free element
c                a(l,l) with l .ge. k.  jpvt is not referenced if
c                job .eq. 0.
c
c        job     integer.
c                job is an integer that initiates column pivoting.
c                if job .eq. 0, no pivoting is done.
c                if job .ne. 0, pivoting is done.
c
c     on return
c
c         a      a contains in its upper half the cholesky factor
c                of the matrix a as it has been permuted by pivoting.
c
c         jpvt   jpvt(j) contains the index of the diagonal element
c                of a that was moved into the j-th position,
c                provided pivoting was requested.
c
c         info   contains the index of the last positive diagonal
c                element of the cholesky factor.
c
c     for positive definite matrices info = p is the normal return.
c     for pivoting with positive semidefinite matrices info will
c     in general be less than p.  however, info may be greater than
c     the rank of a, since rounding error can cause an otherwise zero
c     element to be positive. indefinite systems will always cause
c     info to be less than p.
c
c     linpack. this version dated 08/14/78 .
c     j.j. dongarra and g.w. stewart, argonne national laboratory and
c     university of maryland.
c
c
c     blas daxpy,dswap
c     fortran dsqrt
```

```
c
c     internal variables
c
      subroutine  dcopy(n,dx,incx,dy,incy)
c================================================================c
c
c     copies a vector, x, to a vector, y.
c     uses unrolled loops for increments equal to one.
c     jack dongarra, linpack, 3/11/78.
c
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      double precision function ddot(n,dx,incx,dy,incy)
c================================================================c
c
c     forms the dot product of two vectors.
c     uses unrolled loops for increments equal to one.
c     jack dongarra, linpack, 3/11/78.
c
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      function dgamma (x)
c================================================================c
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine dsidi(a,lda,n,kpvt,det,inert,work,job)
c================================================================c
      integer lda,n,job
      double precision a(lda,1),work(1)
      double precision det(2)
      integer kpvt(1),inert(3)
c
c     dsidi computes the determinant, inertia and inverse
c     of a double precision symmetric matrix using the factors from
c     dsifa.
c
c     on entry
c
c        a       double precision(lda,n)
c                the output from dsifa.
c
```

```
c        lda     integer
c                the leading dimension of the array a.
c
c        n       integer
c                the order of the matrix a.
c
c        kpvt    integer(n)
c                the pivot vector from dsifa.
c
c        work    double precision(n)
c                work vector.  contents destroyed.
c
c        job     integer
c                job has the decimal expansion  abc  where
c                   if  c .ne. 0, the inverse is computed,
c                   if  b .ne. 0, the determinant is computed,
c                   if  a .ne. 0, the inertia is computed.
c
c                for example, job = 111  gives all three.
c
c     on return
c
c        variables not requested by job are not used.
c
c        a      contains the upper triangle of the inverse of
c               the original matrix.  the strict lower triangle
c               is never referenced.
c
c        det    double precision(2)
c               determinant of original matrix.
c               determinant = det(1) * 10.0**det(2)
c               with 1.0 .le. dabs(det(1)) .lt. 10.0
c               or det(1) = 0.0.
c
c        inert  integer(3)
c               the inertia of the original matrix.
c               inert(1)  =  number of positive eigenvalues.
c               inert(2)  =  number of negative eigenvalues.
```

```
c                inert(3)  =  number of zero eigenvalues.
c
c     error condition
c
c        a division by zero may occur if the inverse is requested
c        and  dsico  has set rcond .eq. 0.0
c        or  dsifa  has set  info .ne. 0 .
c
c     linpack. this version dated 08/14/78 .
c     james bunch, univ. calif. san diego, argonne nat. lab
c
c     subroutines and functions
c
c     blas daxpy,dcopy,ddot,dswap
c     fortran dabs,iabs,mod
c
c     internal variables.
c
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine dsifa(a,lda,n,kpvt,info)
c================================================================c
      integer lda,n,kpvt(1),info
      double precision a(lda,1)
c
c     dsifa factors a double precision symmetric matrix by elimination
c     with symmetric pivoting.
c
c     to solve  a*x = b , follow dsifa by dsisl.
c     to compute  inverse(a)*c , follow dsifa by dsisl.
c     to compute  determinant(a) , follow dsifa by dsidi.
c     to compute  inertia(a) , follow dsifa by dsidi.
c     to compute  inverse(a) , follow dsifa by dsidi.
c
c     on entry
c
c        a       double precision(lda,n)
c                the symmetric matrix to be factored.
c                only the diagonal and upper triangle are used.
```

```
c
c        lda     integer
c                the leading dimension of the array  a .
c
c        n       integer
c                the order of the matrix  a .
c
c     on return
c
c        a       a block diagonal matrix and the multipliers which
c                were used to obtain it.
c                the factorization can be written  a = u*d*trans(u)
c                where  u  is a product of permutation and unit
c                upper triangular matrices , trans(u) is the
c                transpose of  u , and  d  is block diagonal
c                with 1 by 1 and 2 by 2 blocks.
c
c        kpvt    integer(n)
c                an integer vector of pivot indices.
c
c        info    integer
c                = 0  normal value.
c                = k  if the k-th pivot block is singular. this is
c                     not an error condition for this subroutine,
c                     but it does indicate that dsisl or dsidi may
c                     divide by zero if called.
c
c     linpack. this version dated 08/14/78 .
c     james bunch, univ. calif. san diego, argonne nat. lab.
c
c     subroutines and functions
c
c     blas daxpy,dswap,idamax
c     fortran dabs,dmax1,dsqrt
c
c     internal variables
c
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
      subroutine  dswap (n,dx,incx,dy,incy)
c===============================================================c
c
c     interchanges two vectors.
c     uses unrolled loops for increments equal one.
c     jack dongarra, linpack, 3/11/78.
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine f3swap(x,y)
c===============================================================c
c     required for pheev3
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine fillm(vec,npi,npj,val)
c===============================================================c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine fillv(a,np,val)
c===============================================================c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      double precision function fmax(vec,n)
c===============================================================c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      double precision function fmin(vec,n)
c===============================================================c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      double precision function hsec()
c===============================================================c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      double precision function hsec1990()
c===============================================================c
c
c vah 17:25:23.43, sat, nov 24, 1990
c
c returns number of hundredth's of a second since jan 1, 1990
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

      integer function icoccfst(needle,haystack)
c===============================================================c
c ----------------------------------------------------------- C
```

```
c
c     vah 3/10/1988,10:51:56:23
c         20:40:48.97, Sat, Sep 1, 1990
c
c     Returns integer location where needle occurs first in haystack.
c     Returns 0 if no match.
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      integer function icocclst(needle,haystack)
c===============================================================c
C ----------------------------------------------------------- C
c
c     vah 14:30:53.47, Sat, Sep 1, 1990
c
c     Returns integer location where needle occurs last in haystack.
c     Returns 0 if no match.
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      integer function idamax(n,dx,incx)
c===============================================================c
c
c     finds the index of element having max. absolute value.
c     jack dongarra, linpack, 3/11/78.
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine invv(a,b,ia,ib,c,n1,npr,nzr,nnr,rcond,*)
c===============================================================c
c     vah : 6:14 pm, friday, july 11, 1986
c***********************************************************************
c
c     nb: assumes a is symmetric
c                      *********
c
c     on entry,
c        *****
c     a contains the matrix to be inverted,
c     ia is the lead dim of a, ib the lead dim of b,
c     n1 the effective dim.
```

```
c
c     on exit,
c        ****
c     a is the inverse, b contains the eigenvectors,
c     c is the vector of eigenvalues in descending order,
c     npr=# of +ve roots, nnr=# of -ve roots, nzr=# of 0 roots
c
c     error-returns
c     ************
c     case 1: eigen value calculations did not converge(matev2 int=2)
c     case 2: a root is smaller or equal
c     to machine absolute precision (eps2), without attempting to
c     invert the matrix. in both cases, a will be wrong in general
c
c     written by vassilis hajivassiliou 30mar85,
c     using matev2(a,b,n1,ia,ib,int)
c     from gqopt.
c
c     eps1= machine relative accuracy (max # of significant digits)
c     eps2= machine absolute accuracy (smallest epsilon > 0)
c***********************************************************************
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      function iseednew(iseed0,icycle)
      integer*4 iseednew
c===============================================================c
c
c     vah 14:38:37.59, Mon, Mar 16, 1992
c
c     Starts from iseed0, and calls randu icycle times.
c     For the same iseed0 value, it will thus return a different seed
c     for every different value of icycle.
c
c     iseed0 and icycle are NOT changed.
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine lc(ch)
c===============================================================c
c     changes the character variable ch to lower case
```

```
c     vah 3:26 am, monday, october 27, 1986
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      integer function ltrim(string)
c================================================================c
c     vah 6:47 pm, saturday, february 27, 1988
      character*(*) string
c
c    return the blank-stripped length
c
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine machine(machnam)
c================================================================c
c
c     vah 8:41:23.91, Tue, Mar 24, 1992
c
c     returns the value of the environment variable 'machine'
c     requires machine-dependent subprogram GETEVAR
c
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine matptv(a,lda,b,ldb,c,ldc,n,k,l)
c================================================================c
c
c     c=a-transpose*b
c
c     a(lda,k) b(ldb,l) c(ldc,l)
c       nxk      nxl      kxl
c
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine matpv(a,lda,b,ldb,c,ldc,n,k,l)
c================================================================c
c
c     c=a*b
c
c     a(lda,k) b(ldb,l) c(ldc,l)
c       nxk      kxl      nxl
c
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine mprint(a,lda,n,m,aname,nfile,ndig)
```

```
c===============================================================c
c
c     vah : note that aname is character*(*)
c
c      dbpm   trace print of matrix
c      prints the matrix a in table of n rows and m columns
c      lda is the no. of rows in dimension of a
c      aname is the name of array
c      nfile is the file unit number where the output should go
c      ndig=6,7 or 8 for  6 per line in d16.8
c      ndig=3,4 or 5 for  8 per line in d12.4
c      ndig=0,1 or 2 for 10 per line in d9.1
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      double precision function nrmpdf(x,xm,sd)
c===============================================================c
c ----------------------------------------------------------- c
c     vah  2:48 pm, friday, april 18, 1986                    c
c         11:38 pm, wednesday, september 24, 1986             c
c     msl  4:20 pm, thursday, october 6, 1988                 c
c     vah+msl, 5:40 pm, wed, dec 7, 1988 (pareto pdf)         c
c ----------------------------------------------------------- c
c ----------------------------------------------------------- c
c                                                             c
c     1.  double precision function nrmpdf(x,xm,sd)           c
c     2.  double precision function unfpdf(x,a,b)             c
c     3.  double precision function betpdf(arg,a,d)           c
c     4.  double precision function binpdf(x,t,p)             c
c     5.  double precision function caupdf(x,xloc,scale)      c
c     6.  double precision function combnr(xn,xr)             c
c     7.  double precision function exppdf(x,theta)           c
c     8.  double precision function gampdf(x,theta,t)         c
c     9.  double precision function parpdf(x,theta)           c
c    10.  double precision function permnr(xn,xr)             c
c    11.  double precision function dgamcp(x)                 c
c    12.  double precision function xfact(x)                  c
c    13.  double precision function logpdf(arg,xm,std)       c
c    14.  double precision function dexpdf(arg,xm,std)       c
```

```
c                                                               c
c ------------------------------------------------------------- c
      implicit double precision (a-h,o-z)
      implicit integer*4 (i-n)
      external sfee
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      double precision function ph3del(x,y)
c=================================================================c
c     required for pheev3
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      double precision function ph3s(x,a,b,phix,txa)
c=================================================================c
c     required for pheev3
      implicit double precision(a-h,o-z)
c     computes steck's (1958 ann math stat) s-fn with max error e-9,
c     though computer rounding errors may reduce this accuracy. calls
c     function subprograms ph3stk(x,a,b) in which dabs(b) < 1.1,
c     pheev(x), and ph3t(x,a,phix) note that phix = pheev(x)
c     and txa = ph3t(x,a,phix).
      double precision function ph3stk(x,a,b)
c=================================================================c
c     required for pheev3
      double precision function ph3t(z,a,phix)
c=================================================================c
c     required for pheev3
c        owens t double precision function accurate to .00005
c        phix is a dummy argument
      double precision function pheev(y)
c=================================================================c
c     this essentially uses the imsl real*8 function derf, provided
c     by clint cummins for use in quail5.1 . the modification using
c     the fact that phee(z)=0.5+0.5*erf(z/sqr2) was done by vah .
c
c                                 specifications for arguments
c       sqr2, pt5 added by vah
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      double precision function pheev2(z1,z2,r)
c=================================================================c
```

```
c  program to compute bivariate normal cdf
c  cacm algorith number 462
c  based on algorithm in owen, annals of math stat, 1956:1075-90
c  accuracy is controlled by idig--number of sig digits to right of
c  decimal point in answer.  more accurate than old hausman routine
c  with idig=15 time is approximately .05 seconds on pc w/8087
c  time varies with values.  can be faster
c
c  modified by vah 6:49 pm, tuesday, december 9, 1986
c
c  routines used:
c      pheev(z) is standard cumulative normal at z
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine pheev3(z,x1,x2,x3,rho12,rho31,rho23,*)
c================================================================c
c     vah 12:55 am, sunday, may 22, 1988
c         based on farber's program tnorm
c
c     for the trivariate normal r.v. (x,y,w) with zero means, unit
c     variance and corr(x,y)=rho12,corr(y,w)rho23,corr(w,x)=rho31,tnor
c     uses the method of section 2 of steck (1958) to compute z= pr
c     (x<x1,y<x2,w<x3) accuracy of the result depends on the
c     subprograms ph3t and ph3stk. for more detail re accuracy see daley
c     (1973) anu stat dept (ias) rep. ref.: g.p. steck (1958) ann math
c     statis vol 29,pp 780-800.
c        subroutine ph3t and ph3stk use closed form approximations
c        accur to about .00025,  time is about .003
c
c     requires routines:
c      1. f3swap
c      2. ph3del
c      3. ph3t
c      4. ph3s
c      5. ph3stk
c      6. pheev
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      double precision function phinv(ppp)
```

```
c===============================================================c
c     28 mar 86
c     12:33 pm, monday, june 30, 1986
c*****inverse normal c.d.f. (pheev(ppp)=phinv) using
c*****formula 26.2.23 of abramovitz and stegun p.933 10th edition.
c*****ppp unchanged on exit. accuracy<4.5e-4
c*****
c*****written by v.a. hajivassiliou 24sep84
c*****
c*****uses inverse interpolation to 3rd order so as to
c*****raise accuracy-see a&s example 5 p.954
c*****
c*****modification done 14apr85
c*****
c*****subprogram(s) used: pheev
c*****
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      double precision function randu(iseed)
c===============================================================c
c     vah : 28 mar 86
c     1:07 pm, monday, june 30, 1986
c*****
c*****this is a pseudo random number generator,
c*****randu being the realization of a uniform on [0,1]
c*****pseudo r.v.. requires a seed (iseed) which is updated
c*****and returned. written by v.a. hajivassiliou, 24sep84,
c*****to exactly reproduce the pr1me uniform pseudo random
c*****generator rand$a(iseed).
c*****ref:
c*****lewis,goodman & miller,ibm systems journal,v.8#2 1969,pp.136-145
c*****
c*****uses multiplier number 1 for fishman and moore, "a statistical
c*****evaluation of multiplicative congruential random number generators
c*****with modulus 2**31-1", jasa 77, march 1982.  this was found to
c*****be quite acceptable in the tests, and the fastest.
c
c*****
c*****note: the imsl function ggubs is exactly the same algorithm,
```

```
c*****      except that it divides by 2147483648 , potentially
c*****      speeding the process up on 32 bit machines
c*****
ccc can't re-mention randu in rm profort:
ccc      double precision randu,xm,xb,seed,xnews
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      double precision function ranorm(xm,sd,iseed)
c================================================================c
c**********************************************************************
c*****           random number generators package
c*****           vassilis hajivassiliou october 1984
c***** last modified : 5:45 pm, wednesday, september 24, 1986
c**********************************************************************
c     this set contains:
c     *****************
c     1. ranorm(xm,sd,iseed)         : (tsp)
c     2. binrmb(x,y,xm,ym,xs,ys,rho,iseed)
c     3. birnrm(x,y,xm,ym,xs,ys,rho,iseed)
c     4. phinv(p)
c     5. randb(prob,iseed)
c     6. randc(xm,scale,iseed)
c     7. randcv(xm,scale,iseed)
c     8. rande(xm,iseed)
c     9. randl(xm,sd,iseed)
c    10. randp(xlambda,iseed)
c    11. randpa(xtheta,iseed)
c    12. randu(iseed)
c    13. ranorv(xm,sd,iseed)        : (using phinv)
c    14. ranrmb(xm,sd,iseed)
c    15. randde(xm,sd,iseed)
c
c**********************************************************************
c
c     normal(xm,sd) random number generator . vassilis hajivassiliou
c
c     october 1984
c
c     this is from cacm algorithm #488, originally implemented
```

```
c     for use in tsp4.0 by  bronwyn h. hall(oct82).
c
c**********************************************************************
c     subprogram(s) used: randu
c**********************************************************************
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

      double precision function sfee(x)
c===============================================================c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine sorts(y,n)
c===============================================================c
c     shell sort n values in t() from smallest to largest.
c     *****
c
c     note that local system sort utilities are likely to be
c     more efficient, and should be substituted whenever possible.
c
c     "abc's of eda" manual 1981 p.313
c     written 11/03/85 17:17:29
c
c     local variables
c
ccc      integer*2 i, j, j1, igap, nmg
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine srtrnk(yvec,n,ivec)
c===============================================================c
c
c     vah 8:11 pm, tuesday, october 28, 1986
c         16:50:51.97, sun, apr 9, 1989
c
c
c     shell sort n values in yvec() from smallest to largest.
c     *****
c     vah 7:29 pm, tuesday, october 28, 1986
c         modified to return also array ivec.
c     on entry, ivec(1) controls the operations to be performed:
c     if ivec(1) ge 0: yvec is sorted from min to max (call this xsort)
c                      ivec will contain the original locations of the
```

```
c                      elements of yvec (call this iloc)
c     if ivec(1) lt 0: yvec is sorted from min to max in place, i.e.
c                      the original yvec is returned
c                      ivec will contain the ranks of the yvec
c                      (call this irnk)
c-----------------------------------------------------------------------
c     the full relations are:
c
c           information returned with       information returned with
c  original        ivec(1) ge 0                 with ivec(1) lt 0
c  --------  -------------------------       -------------------------
c  yvec      yvec      ivec                 yvec      ivec
c  ----      ----      ----                 ----      ----
c  x         xsort     iloc                 x         irnk
c  -         -----     ----                 -         ----
c  2         1         4                    2         2
c  4         2         1                    4         3
c  7         4         2                    7         5
c  1         5         5                    1         1
c  5         7         3                    5         4
c
c           to obtain x and irnk            to obtain xsort and iloc
c           from xsort, iloc                from x, irnk
c           --------------------            ------------------------
c           x(iloc(j)=xsort(j)              xsort(irnk(j))=x(j)
c           irnk(iloc(j))=j                 iloc(irnk(j))=j
c
c-----------------------------------------------------------------------
c
c     note that local system sort utilities are likely to be
c     more efficient, and should be substituted whenever possible.
c
c     "abc's of eda" manual 1981 p.313
c     written 11/03/85 17:17:29
c
c     local variables
c
ccc      integer*2 i, j, j1, igap, nmg
```

```
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine uc(ch)
c===============================================================c
c     changes the character variable ch to upper case
c     vah 5:13 pm, thursday, april 3, 1986
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine when(nfunit)
c===============================================================c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine antith(kmax,k,irsimBIG,it,v0,a)
c===============================================================c
c
c ANTITH is a routine that creates a "grid" of antithetic points S on the
c unit sphere, starting from a random basis provided by RNDBASE.
c These routines will ordinarily be called for each observation.  To
c avoid "chatter" when parameter values are changed, the seed for the
c random number generator for each observation should be stored so that
c the same outputs can be regenerated in iteration to parameter estimates.
c
ccc   k=m
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine arse(mmax,irmax,m,xmu,w,wi,chw,a,b,ir,u,parm,p,hu,hc)
c===============================================================c
c
c   P. Acceptance/Rejection Simulator (ARS)
c      The ARS simulator is given in a version using the truncated
c bilateral exponential comparison distribution (ARSE), and in a
c version using a recursive truncated comparison distribution (ARSR).
c PARM[.,1] is assumed to contain seeds for the uniform random number
c generator RNDUS called by the routines; there is one seed for each
c repetition R.  PARM[R+1,1] is assumed to contain a censoring limit
c for rejection.  Should only compute HC.
c
c The ARS method provides a mechanism for drawing from a conditional
c density when transformations from uniform or standard normal variates
c are not available.  We want to sample from f(x/a).  We generate x
c from g(x).  In our case g(x) = ARSE.  Generate tsi from uniform.
```

```
c Generate unconditional f(x) and  bound alpha. We accept x if
c                tsi < f(x)/(g(x)*alpha)
c x has conditional density f(x/a) = HC.  We set x = VD.
c
c NB: Upon exit, the first element of the BLANK common block, contains
c --  the proportion of non-censored draws
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine arsr(mmax,irmax,m,xmu,w,wi,chw,a,b,ir,u,parm,p,hu,hc)
c===============================================================c
c
c see ARSE for documentation.
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine aus(mmax,irmax,m,xmu,w,wi,chw,a,b,ir,u,parm,p,hu,hc)
c===============================================================c
c
c   S. Asymptotically Unbiased Simulator (AUS)
c      The AUS simulator is obtained as the product of an unbiased
c simulator of H and an asymptotically unbiased simulator of 1/P.  The
c proc is defined with the GHK simulator used for H and for the initial
c independent simulations of P(B)/P(A), where A is the event that the
c first component is in the rectangle bounds.  The proc assumes that
c PARM(.,1) contains a vector of R seeds for the normal random number
c generator RNDNS, PARM(1,2) contains the number of initial GHK
c simulators of P used in the simulation of 1/P, PARM(2,2) contains the
c maximum number of trials using the crude frequency simulator that are
c made before the process is censored.  If PARM(2,2) is made large (say
c 1.e50), then this is computationally the same as the Sequential
c Unbiased Simulator (SUS).  PARM(3,2) contains a seed for RNDUS.
c Gives HU and HC.
c
c This does continuous parameter estimation of 1/p.
c
c NB: Upon exit, the first element of the BLANK common block, contains
c --  the maximum number of CFS draws allowed in AUS.
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
      subroutine bexp(mmax,m,v,xlam,p)
c===============================================================c
c
c bilateral exponential cdf.
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine bexpi(mmax,m,ir,p,xlam,pa,pb,x,d)
c===============================================================c
c
c inverse truncated bilateral exponential cdf.
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine cf(irmax,m,ir,alp,bet,gam,y0,y1,y2)
c===============================================================c
c
c see PCF for documentation.
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine cfs(mmax,irmax,m,xmu,w,wi,chw,a,b,ir,u,parm,p,hu,hc)
c===============================================================c
c
c   I. Crude Frequency Simulator (CFS)
c      This simulator assumes that U is an array of standard normal
c variates, independent within each column, and either independent or
c antithetic across columns.  It does not use PARM.  It returns both HU
c and HC.
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c
      subroutine dcs(mmax,irmax,m,xmu,w,wi,chw,a,b,ir,u,parm,p,hu,hc)
c===============================================================c
c
c   O. Deak Chi-Square Simulator (DCS)
c
c      The DCS assumes that U is an array whose columns are points in
c the unit sphere, antithetic across columns.
c The routine assumes that PARM(1,1) = DET(W).
c
```

```
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine exaf(mmax,m,a,b,xmu,w,xl,p0,dm,dl)
c==================================================================c
c
c approximate answers by numerical quadrature.
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine ffact(mmax,m,x,a,b,xmu,d,xl,y0,y1,y2)
c==================================================================c
c
c integrand in one-factor model.
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine fdl(mmax,m,h,xl,d)
c==================================================================c
c
c return specific elements from the H matrix.
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine fghk(ii,mmax,m,xmu,w,wi,chw,a,b,ir,iseed,pm,p,hu)
c==================================================================c
c
c see GHK for documentation.
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine fi(mmax,irmax,m,ir,a,b,dkap,fi0,fi1,fi2)
c==================================================================c
c
c FI is a proc that returns partial moments of the standard normal.
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine ghk(mmax,irmax,m,xmu,w,wi,chw,a,b,ir,u,parm,p,hu,hc)
c==================================================================c
c
c   M. Geweke-Hajivassiliou-Keane Simulator (GHK)
c      The GHK procedure assumes that U is an array of uniform [0,1]
c variates, independent within columns, and in general independent
c between columns. It does not use PARM.  It returns both HU and HC.
```

```
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine gss(mmax,irmax,m,xmu,w,wi,chw,a,b,ir,u,parm,p,hu,hc)
c===============================================================c
c
c   Q. Gibbs's Sampler Simulator (GSS)
c      The GSS simulates HC, but does not simulate P or HU.  It assumes
c that PARM[1,1] gives the number of rounds to be used in the
c construction of the random draws (JL) and that U is an MxJL array of
c uniform [0,1] random numbers.
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine nden(mmax,irmax,m,ir,vd,wi,chw,q)
c===============================================================c
c
c NDEN is the multivariate normal density.
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine ndenist(mmax,irmax,m,ir,vd,wi,chw,q)
c===============================================================c
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

      subroutine nise(mmax,irmax,m,xmu,w,wi,chw,a,b,ir,u,parm,p,hu,hc)
c===============================================================c
c
c   J. Normal Importance Sampling Simulator (NIS)
c      This simulator provides HC = HU/P.  It provides two alternative
c simulators, corresponding to the exponential (NISE) and independent
c truncated normal (NIST) comparison distributions.  These simulators
c assume U is an array of uniform random (0,1) variates.  These procs
c call the routine NDEN that returns a vector of values of the
c multivariate normal density.
c      Procedure NISE does not use C or PARM.  It calls the bilateral
c exponential CDF BEXP and its inverse BEXPI from LIB_A.G.  It
c guarantees P positive. Provides both HU and HC.
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine nist(mmax,irmax,m,xmu,w,wi,chw,a,b,ir,u,parm,p,hu,hc)
```

```
c===============================================================c
c
c see NISE for documentation.
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine pcf(mmax,irmax,m,xmu,w,wi,chw,a,b,ir,u,parm,p,hu,hc)
c===============================================================c
c
c   N. Parabolic Cylinder Function Simulator (PCF)
c      The PCF proc assumes that U is an array whose column vectors are
c points in the intersection of the unit sphere and the negative
c orthant.
c The routine assumes that PARM(1,1) = DET(W).
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine pf(mmax,m,aa,bb,xmu,w,p)
c===============================================================c
c
c used by EXAF.
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine pker(z,mmax,irmax,m,ir,prodzwgt)
c===============================================================c
c
c see PKF for documentation.
c
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
c     Polynomial kernel function; PKFS calls this routine.
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine pkfs(mmax,irmax,m,xmu,w,wi,chw,a,b,ir,u,parm,p,hu,hc)
c===============================================================c
c
c   K. Polynomial Kernel-Smoothed Frequency Simulator (PKF)
c      The KFS simulator uses the kernel (31), defined by the function
c PKER. It assumes that U is an array of standard normal variates,
c independent within rows.  It does not use W.  PARM[1,1] contains a
c window width parameter.
c
```

```
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine reswrite(cfil,line,irec)
c===============================================================c
c     direct access version
c     Reswrite writes the character string line to the end of the cfil.
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine rndbase(kmax,k,iseed1,v,t)
c===============================================================c
c
c RNDBASE is a routine that draws a random basis in K space, where K is
c the dimension of the random coefficient vector.
c RNDBASE takes as input a dimension K and returns a K by K array V that
c is column orthonormal and random, bordered by a column of random
c lengths of K-dim. normal random vectors.  SEED1 is a seed for the
c random number generator.
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine rndt(mmax,irmax,m,ir,u,w,a,b,v,y)
c===============================================================c
c
c RNDT is a proc for drawing an array from truncated independent
c normals that have the same means and variances as the multivariate
c density.
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine sds(mmax,irmax,m,xmu,w,wi,chw,a,b,ir,u,parm,p,hu,hc)
c===============================================================c
c
c   L. Stern Decomposition Simulator (SDS)
c      The SDS proc assumes that PARM[1,1] contains a scalar l > 0 such
c that W-lI is positive definite, and assumes that C is a Cholesky
c factor of W-lI. It assumes that U is an array of standard normal
c variates, independent within each column.
c
c+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine sim(mmax,m,xl,e1,p,hu,hc,ec,pp,dm,xlm,dl,xll)
c===============================================================c
c
```

```
c procedure to accumulate results.
c
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      subroutine sus(mmax,irmax,m,xmu,w,wi,chw,a,b,ir,u,parm,p,hu,hc)
c================================================================c
c
c   R. Sequentially Unbiased Simulator (SUS)
c      The SUS simulator is obtained as the product of an unbiased
c simulator of H and an unbiased simulator of 1/P.
c Computationally, if PARM(2,2) is made large (say 1.e50), SUS is the
c same as the Asymptotically Unbiased Simulator (AUS).
c
c NB: Upon exit, the first element of the BLANK common block, contains
c --  the maximum number of CFS draws allowed in AUS
c++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```