# 3.2 Gaussian (Normal) Random Numbers and Vectors

## A. Purpose

Generate pseudorandom numbers or vectors from the Gaussian (normal) distribution.

## B. Usage

### B.1 Generating Gaussian (normal) pseudorandom numbers

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

**REAL SRANG, X**

$$\boxed{\mathbf{X = SRANG()}}$$

#### B.1.b Argument Definitions

**SRANG** [out] The function returns a pseudorandom number from the Gaussian (normal) distribution with mean zero and unit standard deviation.

### B.2 Generating Gaussian (normal) pseudorandom vectors

Given an N-vector, $\mu$, and a symmetric positive-definite N×N matrix, $A$, the objective is to compute pseudorandom N-vectors, **x**, from the N-dimensional Gaussian (normal) distribution having mean vector, $\mu$, and covariance matrix, $A$.

On the first call to SRANGV with a new covariance matrix, $A$, the user must set HAVEC = .false. to indicate that the Cholesky factor of $A$ has not yet been computed. SRANGV will replace $A$ in storage by its Cholesky factor, $C$, and set HAVEC = .true. to indicate that the array A() now contains $C$ rather than $A$.

On each call to SRANGV an N-vector, **x**, is returned as a pseudorandom sample from the specified distribution.

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

**INTEGER NDIM, N, IERR**

**REAL A**(NDIM≥N, ≥N)**, U**(≥N)**, X**(≥N)

**LOGICAL HAVEC**

Assign values to A(,), NDIM, N, U(), and HAVEC.

**CALL SRANGV(A, NDIM, N, U, X, HAVEC, IERR)**

Computed values will be returned in X() and IERR, and the contents of A(,) and HAVEC may be changed.

©1997 Calif. Inst. of Technology, 2015 Math à la Carte, Inc.

#### B.2.b Argument Definitions

**A(,)** [inout] When HAVEC is .false., A(,) contains a user-supplied N×N symmetric positive-definite covariance matrix, $A$. Only the diagonal and subdiagonal elements of $A$ need be given. This subroutine will not access the super-diagonal locations in the array, A(,). This subroutine will replace $A$ by its lower-triangular Cholesky factor, $C$, and set HAVEC = .true.

When HAVEC is .true., A(,) is assumed to contain the Cholesky factor, $C$.

**NDIM** [in] First dimension of the array A(,). Require NDIM ≥ N.

**N** [in] Order of the covariance matrix $A$ and dimension of the vectors U and X. Require N ≥ 1.

**U()** [in] Contains the N-dimensional mean vector, $\mu$.

**X()** [out] On return will contain the N-dimensional generated random vector.

**HAVEC** [inout] See description above for A(,).

**IERR** [out] IERR will only be referenced when the subroutine is entered with HAVEC = .false. If the Cholesky factorization is successful, the subroutine will set HAVEC = .true. and IERR = 0. Otherwise, it will leave HAVEC = .false. and set IERR to the index of the row of $A$ in which failure was noted. In this latter case the results returned in A(,) and X() will not be useful.

### B.3 Modifications for Double Precision

For double-precision usage, change the REAL statements to DOUBLE PRECISION and change the initial "S" of the function and subroutine names to "D." Note particularly that if the function name, DRANG, is used it must be typed DOUBLE PRECISION either explicitly or via an IMPLICIT statement.

## C. Examples and Remarks

The program DRSRANG demonstrates the use of SRANG to compute Gaussian random numbers and uses SSTAT1 and SSTAT2 to compute and print statistics and a histogram based on a sample of 10000 numbers delivered by SRANG.

To compute Gaussian random numbers with mean XMEAN and standard deviation STDDEV, one can use the statement

X = XMEAN + STDDEV ∗ SRANG()

The program DRSRANGV demonstrates the use of SRANGV to compute pseudorandom vectors from a 3-dimensional Gaussian distribution with mean vector and covariance matrix specified as

$$\mu = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad \text{and} \quad A = \begin{bmatrix} 0.05 & 0.02 & 0.01 \\ 0.02 & 0.07 & -0.03 \\ 0.01 & -0.03 & 0.06 \end{bmatrix}.$$

To fetch or set the seed used in the underlying pseudorandom integer sequence use the subroutines described in Chapter 3.1.

## D. Functional Description

### Method

The algorithm for generation of Gaussian random numbers is based on [1]–[3]. This method draws pairs of uniform random numbers $x$ from [0, 1] and $y$ from [−1, 1] until a pair is obtained that satisfies $x^2 + y^2 \le 1$. The probability of satisfying this constraint is $\pi/4 \approx 0.785$. It then draws another uniform random number $u$ from [0, 1] and computes

$$\begin{aligned} s &= x^2 + y^2 \\ t &= \sqrt{-2\ \log u}\,/s \\ g_1 &= t(x^2 - y^2) \\ g_2 &= 2xyt \end{aligned}$$

The numbers $g_1$ and $g_2$ are independent samples from the Gaussian distribution with mean zero and unit standard deviation. The number $g_1$ is returned when it is computed and $g_2$ is saved and returned the next time a Gaussian random number is requested. The saved value will be discarded if the underlying uniform sequence is reinitialized by a call to RAN1 or RANPUT of Chapter 3.1.

This method is a mathematically exact transformation from the uniform distribution to the Gaussian distribution so the statistical quality of the delivered numbers depends entirely on the quality of the uniform pseudorandom numbers used. The uniform numbers are obtained by calling SRANUA or DRANUA, using the array in common block /RANCMS/ or /RANCMD/ as a buffer as described in Chapter 3.1.

For the generation of Gaussian random vectors we are given a mean vector $\mu$ and a symmetric positive-definite covariance matrix $A$. The Cholesky method is used to factor the given covariance matrix $A$ as $A = CC^t$ where $C$ is a lower triangular matrix. Then for each vector to be delivered the method first constructs a vector $\mathbf{g}$ whose components are independent samples from the Gaussian distribution with mean zero and unit standard deviation. It then computes

$$\mathbf{x} = \mu + C\mathbf{g}$$

which is a sample from the N-dimensional Gaussian distribution with mean $\mu$ and covariance matrix $A$.

Values returned as double-precision random numbers will have random bits throughout the word, however the quality of randomness should not be expected to be as good in a low-order segment of the word as in a high-order part.

### References


1. James R. Bell, *Algorithm 334 — Normal random deviates*, **Comm. ACM 11**, 7 (July 1968) 498.
2. G. Box and M. Muller, *Note on the generation of normal deviates*, **Ann. Math. Stat. 28** (1958) 610.
3. John von Neumann, **Various techniques used in connection with random digits in the National Bureau of Standards**, *Applied Mathematics Series 12*, National Bureau of Standards (1959) 36 pages.


## E. Error Procedures and Restrictions

While computing the Cholesky factorization of $A$, subroutine SRANGV or DRANGV may determine that the matrix is not positive-definite. In that case it will return with IERR set to the index of the row of the matrix at which the problem was detected. In this non-positive-definite case the contents of the array A(,) will have been altered, HAVEC will still have the value .false., and X() will not contain useful results.

When the Cholesky factorization is successful, IERR will be set to zero.

The conditions that N ≥ 1, and NDIM ≥ N are required but not checked.

## F. Supporting Information

The source language is ANSI Fortran 77.

| Entry | Required Files |
|---|---|
| **DRANG** | DRANG, ERFIN, ERMSG, RANPK1, RANPK2 |
| **DRANGV** | DRANG, DRANGV, ERFIN, ERMSG, RANPK1, RANPK2 |
| **SRANG** | ERFIN, ERMSG, RANPK1, RANPK2, SRANG |
| **SRANGV** | ERFIN, ERMSG, RANPK1, RANPK2, SRANG, SRANGV |

Based on subprograms written for JPL by Carl Pitts, Heliodyne Corp., April, 1969. Adapted to Fortran 77 for the JPL MATH77 library by C. L. Lawson and S. Y. Chiu, JPL, April 1987. November 1991: Lawson reorganized and renamed common blocks. Improved coordination between Gaussian and uniform sequences on reinitializations.

## DRSRANG

```
c      program DRSRANG
c>> 2001-05-22 DRSRANG  Krogh Minor change for making .f90 version.
c>> 1996-05-28 DRSRANG  Krogh Added external statement.
c>> 1994-10-19 DRSRANG  Krogh  Changes to use M77CON
c>> 1987-12-09 DRSRANG  Lawson  Initial Code.
c--S replaces "?": DR?RANG, ?RANG, ?STAT1, ?STAT2
c      Demonstration driver for SRANG.  Generates random numbers
c      from the Gaussian distribution.
c      ------------------------------------------------------------------
      integer NCELLS
      parameter(NCELLS = 12+2)
      external SRANG
      real               SRANG, STATS(5), Y1, Y2, YTAB(1), ZERO
      integer   I, IHIST(NCELLS), N
c
      parameter(ZERO = 0.0E0)
      data       N / 10000/
      data   Y1, Y2 / -3.0E0, +3.0E0/
c      ------------------------------------------------------------------
      STATS(1) = ZERO
      do 20 I=1,N
c                            GET RANDOM NUMBER
         YTAB(1) = SRANG()
c
c                            Accumulate statistics and histogram.
c
         call SSTAT1(YTAB(1), 1, STATS, IHIST, NCELLS, Y1, Y2)
   20 continue
c                            Print the statistics and histogram.
c
      write(*,'(13x,a//)') 'Gaussian random numbers from SRANG'
      call SSTAT2(STATS, IHIST, NCELLS, Y1, Y2)
      stop
      end
```

# ODSRANG

```
              Gaussian random numbers from SRANG

BREAK PT     COUNT          PLOT OF COUNT
                13      *
 -3.00  -------------------------------------------------------
                40      0*
 -2.50  -------------------------------------------------------
               149      0 *
 -2.00  -------------------------------------------------------
               439      0     *
 -1.50  -------------------------------------------------------
               919      0            *
 -1.00  -------------------------------------------------------
              1497      0                     *
 -0.50  -------------------------------------------------------
              1950      0                           *
  0.00  -------------------------------------------------------
              1992      0                            *
  0.50  -------------------------------------------------------
              1432      0                    *
  1.00  -------------------------------------------------------
               906      0            *
  1.50  -------------------------------------------------------
               433      0     *
  2.00  -------------------------------------------------------
               172      0  *
  2.50  -------------------------------------------------------
                40      0*
  3.00  -------------------------------------------------------
                18      *

 Count    Minimum      Maximum       Mean        Std. Deviation

 10000   -4.0209       3.7531      0.32694E-03    0.99037
```

# DRSRANGV

```
c       program DRSRANGV
c>> 2001-05-22 DRSRANGV Krogh Minor change for making .f90 version.
c>> 1994-10-19 DRSRANGV Krogh  Changes to use M77CON
c>> 1987-12-09 DRSRANGV Lawson  Initial Code.
c--S replaces "?": DR?RANGV, ?RANGV
c          Demonstration driver for the Gaussian random vector generator,
c       SRANGV.
c       C. L. Lawson & S. Y. Chiu, JPL, Apr 1987.
c       ------------------------------------------------------------------
        integer I, J, N, IERR, NMAX, NSAMPL
        parameter(NMAX = 3)
        real           A(NMAX,NMAX), U(NMAX), X(NMAX)
        logical HAVEC
        data (U(  J),J=1,NMAX)/ 1.0E0,   2.0E0,   3.0E0 /
        data (A(1,J),J=1,NMAX)/ 0.05E0,  0.02E0,  0.01E0 /
        data (A(2,J),J=1,NMAX)/ 0.02E0,  0.07E0, -0.03E0 /
        data (A(3,J),J=1,NMAX)/ 0.01E0, -0.03E0,  0.06E0 /
c       ------------------------------------------------------------------
        N = NMAX
        NSAMPL = 20
        HAVEC = .false.
        print '(1x,a/1x)','Pseudorandom vectors computed by SRANGV'
        do 10 I = 1,NSAMPL
           call SRANGV(A, NMAX, N, U, X, HAVEC, IERR)
           if(IERR .ne. 0) then
              print '(1x,i3,i6)',N, IERR
           else
              print '(1x,i3,5x,3f12.3)',I,X
           endif
   10   continue
        stop
        end
```

## ODSRANGV

```
Pseudorandom vectors computed by SRANGV

  1          0.922      2.079      2.715
  2          0.868      2.135      2.802
  3          0.983      1.943      2.867
  4          1.102      1.874      2.700
  5          0.677      1.883      2.813
  6          0.763      1.981      2.640
  7          0.690      1.970      2.863
  8          0.729      2.339      2.760
  9          0.660      2.061      2.839
 10          1.246      1.847      2.721
 11          1.230      1.907      2.841
 12          1.215      2.414      2.835
 13          0.447      2.037      2.421
 14          1.126      2.210      3.175
 15          0.891      2.439      2.836
 16          0.792      2.118      2.725
 17          1.298      2.129      3.001
 18          0.999      2.092      3.112
 19          0.921      1.941      2.832
 20          0.965      2.172      3.173
```