

TABLE OF CONTENTS

CHAPTER 1: TIME SERIES	2
CHAPTER 2: FREQUENCY DOMAIN	7
CHAPTER 3: RANDOM VARIABLES.....	15
SECTION 1: BERNOULLI RANDOM DEVIATES	15
SECTION 2: BETA RANDOM DEVIATES.....	16
SECTION 3 : CHI-SQUARE RANDOM DEVIATES	18
SECTION 4: GAUSSIAN RANDOM DEVIATES	23
SECTION 5: STUDENT-T RANDOM DEVIATES.....	40
SECTION 6: WISHART RANDOM DEVIATES	43
SECTION 7: GENERAL RANDOM DEVIATES.....	47
SECTION 8: GENERAL ROUTINES	49
CHAPTER 4: SIMULATION ROUTINES	51
SECTION 1: INPUT/OUTPUT ROUTINES	51
SECTION 2: POSTERIOR SIMULATOR ROUTINES.....	63
CHAPTER 5: MATRIX/VECTOR OPERATIONS	71
SECTION 1: MATRIX OPERATIONS	71
SECTION 2: VECTOR OPERATIONS	89
CHAPTER 6: PARAMETER TRANSFORMATIONS.....	93
CHAPTER 7: AUXILIARY ROUTINES	96
INDEX OF ROUTINES.....	110

Chapter 1: Time Series

This chapter contains functions and routines that are used for applications pertaining to regression models in the time domain.

ACFAR(NP,RX,BETA,SIGMA)

This double precision subroutine finds the AR(p) representation corresponding to a positive definite autocovariance function for lags 0 through p, by solving the Yule-Walker equations. It is the inverse of the subroutine **ARACF**. Note that this routine does not check for the positive definiteness of the autocovariance function.

INPUTS

NP : Parameter p
RX : Autocovariance function.

OUTPUTS

BETA : $NP \times 1$ vector where BETA(j) is the coefficient on lag j in the autoregression.
SIGMA : Standard deviation of the innovation.

USAGE

call ACFAR(NP,RX,BETA,SIGMA)

Last Updated: 29 January, 1997

ARACF(NP,BETA,SIGMA,RX)

This double precision subroutine finds the autocovariance function for lags 0 through p, corresponding to an AR(p) representation, by solving the system of p+1

linear equations. It is the inverse of **ACFAR**. This routine does not check the invertibility of the AR(p) representation.

INPUTS

NP: Parameter p

BETA: $NP \times 1$ vector of coefficients. BETA(j) is the coefficient on the jth lag of the autoregression.

OUTPUTS

RX: $NP \times 1$ vector of coefficients. RX(j) represents the autocovariance at lag j.

SIGMA: Standard deviation of innovation.

USAGE

call ARACF(NP,BETA,SIGMA,RX)

Last Updated: 29 January, 1997

ARPACF(NP,BETA,R)

This double precision subroutine computes the p partial autocorrelation function coefficients, given an autoregression of order p,

$$y_t = \beta_1 y_{t-1} + \dots + \beta_p y_{t-p} + \varepsilon_t .$$

It is the inverse of **PACFAR**.

INPUTS

NP: Order of autoregression, p

BETA: AR coefficients, BETA(j) = β_j

OUTPUT

R: $NP \times 1$ vector of partial autocorrelation coefficients, in order.

USAGE

call ARPACF(NP,BETA,R)

Last Updated: 29 January, 1997

ARPALJ(NP,R)

This double precision real function finds the log determinant Jacobian of transformation from p partial autocorrelation function coefficients to the p coefficients in an AR(p). i.e. $\log\left(\det\left[\frac{\partial \text{BETA}}{\partial \text{R}}\right]\right)$.

INPUTS

NP: Order of autoregression, p

R: $NP \times 1$ vector of partial autocorrelation coefficients, in order.

OUTPUT

ARPALJ: Log determinant Jacobian of transformation.

USAGE

user_var=ARPALJ(NP,R)

Last Updated: 29 January, 1997

LINOP(NP,BETA)

This logical function determines whether an autoregressive lag operator is invertible, by seeing if the partial autocorrelation coefficients of the corresponding autoregression are all less than one in absolute value. The lag operator is

$$1 - \beta_1 z - \beta_2 z^2 \dots - \beta_p z^p.$$

INPUTS

NP : Order of autoregression

BETA : AR coefficients. $BETA(j) = \beta_j$

OUTPUT

LINOP : .TRUE. if invertible; .FALSE. if not

USAGE

user_var=LINOP(NP,BETA)

Last Updated: 29th January, 1997

PACFAR(NP,R,BETA)

This double precision subroutine computes the autoregression of order p corresponding to the first p autocorrelation function coefficients. It is the inverse of **ARPACF**.

INPUTS

NP: Order of autoregression, p

R: $NP \times 1$ vector of partial autocorrelation coefficients, in order

OUTPUT

BETA: $NP \times 1$ vector of AR coefficients, in order

USAGE

call PACFAR(NP,R,BETA)

Last Updated: 29 January, 1997

CHAPTER 2: FREQUENCY DOMAIN

This chapter contains all of the routines and functions that use the frequency domain approach.

ARSDI1(ALAM1,ALAM2)

This double precision real function integrates the product of the spectral density function of a univariate autoregressive process, and the function $\text{Cos}(\lambda\tau)$, between the frequencies λ_1 and λ_2 , and $-\lambda_2$ and λ_1 .

INPUTS

ALAM1: λ_1

ALAM2: λ_2

INPUTS VIA COMMON BLOCK /AUAR/ B, H, TAU, NP

B: $\text{NP} \times 1$ vector of autoregression parameters

H: precision of shock to autoregression

TAU: τ

NP: number of coefficients in autoregression

INPUTS VIA COMMON BLOCK /CONST/ DLOG2, DLGPI, HLG2PI, PI,

PII

DLOG2: $\log(2)$

DLGPI: $\log(2\pi)$

HLG2PI: $\frac{1}{2}\log(2\pi)$

PI: π

PII: π^{-1}

NOTE: The common block /CONST/ is initialized by the subroutine INIT.

OUTPUT

ARSDI1: integrated value of product described above

COMMENTS

If $\lambda_1=0$ and $\lambda_2=\pi$ then the function ARSDI1(0, π) provides the autocovariance at lag τ . If $\tau=0$, then the function ARSDI1(λ_1,λ_2) provides the power of the spectrum over the frequencies (λ_1,λ_2).

USAGE

user_var = ARSDI1(λ_1,λ_2)

Last Updated: 29 January, 1997

FARSDI1(ALAM)

This double precision real function evaluates the product of the spectral density function of a univariate autoregressive process, and the function $\text{Cos}(\lambda\tau)$, at the frequency λ .

INPUTS

ALAM: λ

INPUTS VIA COMMON BLOCK /AUAR/ B, H, TAU, NP

B: $\text{NP} \times 1$ vector of autoregression parameters

H: precision of shock to autoregression

TAU: τ

NP: number of coefficients in autoregression

**INPUTS VIA COMMON BLOCK /CONST/ DLOG2, DLGPI, HLG2PI, PI,
PII**

DLOG2: $\log(2)$

DLGPI: $\log(2\pi)$

HLG2PI: $\frac{1}{2}\log(2\pi)$

PI: π

PII: π^{-1}

NOTE: The common block /CONST/ is initialized by the subroutine INIT.

OUTPUT

FARSD1: value of product above

USAGE

user_var=FARSD1(ALAM)

Last Updated: 29 January, 1997

ZI1(NA,A,NB,B)

This double precision subroutine calculates the inverse of a z-transform,
 $B(Z) = 1 / A(Z)$.

INPUTS

NA: Highest power of A

A: Coefficients A(0) through A(NA)

NB: Highest power of B to compute

OUTPUT

B: Coefficients B(0) through B(NB) , the inverse of A(z).

USAGE

call ZI1(NA,A,NB,B)

Last Updated: 29 January, 1997

ZI2(M,NA,A,LDRA,LDCA,NB,B,LDRB,LDCB)

This double precision subroutine inverts a matrix z -transform,

$$\mathbf{B}(z) = \mathbf{A}(z)^{-1}.$$

INPUTS

NA: Highest power of **A**

A: An $LDRA \times LDCA \times NA$ matrix containing coefficients **A**(0) through **A**(NA)

LDRA: Leading row dimension of **A**

LDCA: Leading column dimension of **A**

NB: Highest power of **B** to compute

LDRB: Leading row dimension of **B**

LDCB: Leading column dimension of **B**

OUTPUT

B: An $LDRB \times LDCB \times NB$ matrix containing coefficients **B**(0) through **B**(NB), the inverse of **A**(z).

Last Updated: 29 January, 1997

ZM1(NA,A,NB,B,NC,C)

This double precision subroutine calculates the product of two z-transforms, $C(z) = A(z) * B(z)$.

INPUTS

NA: Highest power of A
A: Coefficients A(0) through A(NA)
NB: Highest power of B
B: Coefficients B(0) through B(NB)
NC: Highest power of C to compute

OUTPUTS

C: Coefficients C(0) through C(NC)

USAGE

call ZM1(NA,A,NB,B,NC,C)

Last Updated: 29 January, 1997

ZM2(NA,NRA,NCA,A,LDRA,LDCA,NB,NRB,NCB,B,LDRB,LDCB,NC,NRC,NCC,C,LDRC,LDCC)

This double precision subroutine calculates the product of two matrix z-transforms, $C(z) = A(z) * B(z)$

INPUTS

NA: Highest power of A
NRA: Number of rows in A
NCA: Number of columns in A
A: An LDRA×LDCA×NA matrix of coefficients A(0) through A(NA).
LDRA: Leading row dimension of A

LDCA: Leading column dimension of **A**
NB: Highest power of **B**
NRB: Number of rows in **B**
NCB: Number of columns in **B**
B: An LDRB×LDCB×NB matrix of coefficients **B**(0) through **B**(NB).
LDRB: Leading row dimension of **B**
LDCB: Leading column dimension of **B**
NC: Highest power of **C** to compute
NRC: Number of rows in **C**
NCC: Number of columns in **C**
LDRC: Leading row dimension of **C**
LDCC: Leading column dimension of **C**

OUTPUT

C:- An LDRC×LDCC×NC matrix of coefficients **C**(0) through **C**(NC), the product of matrix z-transforms **A**(z) and **B**(z).

USAGE

call ZI2(NA,NRA,NCA,A,LDRA,LDCA,NB,NRB,NCB,B,LDRB,
LDCB,NC,NRC, NCC,C,LDRC,LDCC)

Last Updated: 29 January, 1997

ZQ1(NA,A,NB,B,NC,C)

This double precision subroutine forms the quotient of two z-transforms,

$$C(z) = \frac{A(z)}{B(z)}$$

INPUTS

NA: Highest power of A
A: Coefficients A(0) through A(NA)
NB: Highest power of B
B: Coefficients B(0) through B(NB)
NC: Highest power of C to compute

OUTPUT

C: Coefficients C(0) through C(NC), the quotient of z-transforms A(z) and B(z)

USAGE

call ZQ1(NA,A,NB,B,NC,C)

Last Updated: 29 January, 1997

**ZQ2(NA,NRA,NCA,A,LDRA,LDCA,NB,NRB,NCB,B,LDRB,LDCB,NC,NRC,
NCC,C,LDRC,LDCC)**

This double precision subroutine forms the quotient of two matrix z-transforms,
 $C(z) = B(z)^{-1} * A(z)$.

INPUTS

NA: Highest power of A
NRA: Number of rows in A
NCA: Number of columns in A
A: An NRA×NCA×NA matrix of coefficients A(0) through A(NA)
LDRA: The leading row dimension of A as defined in the calling program
LDCA: The leading column dimension of A as defined in the calling program

NB: The highest power of **B**
NRB: The number of rows in **B**
NCB: The number of columns in **B**
B: An $\text{NRB} \times \text{NRC} \times \text{NB}$ matrix of coefficients **B**(0) through **B**(NB)
LDRB: The leading row dimension of **B**
LDCB: The leading column dimension of **B**
NC: The highest power of **C**
NRC: Number of rows in **C**
NCC: Number of columns in **C**
LDRC: The leading row dimension of **C** as defined in the calling program
LDCC: The leading column dimension of **C** as defined in the calling program

OUTPUT

C: An $\text{NRC} \times \text{NCC} \times \text{NC}$ matrix of coefficients **C**(0) through **C**(NC), the quotient of the matrix z-transforms **A**(z) and **B**(z).

USAGE

call ZQ1(NA,NRA,NCA,A,LDRA,LDCA,NB,NRB,NCB,B,LDRB,
LDCB,NC,NRC,NCC,LDRC,LDCC)

Last Updated: 29 January, 1997

CHAPTER 3: RANDOM VARIABLES

This chapter contains all of the routines and functions that are used in drawing random deviates.

Section 1: Bernoulli Random Deviates

BERN(P)

This double precision real function generates a Bernoulli random variable.

INPUT

P: Probability parameter

OUTPUT

BERN: 1 with probability **P** and 0 with probability 1-**P**

USAGE

user_var = BERN(P)

Last Updated: 4th June, 1997

LBERN(P)

This logical function returns the value **.TRUE.** with probability **P** and **.FALSE.** with probability 1-**P**

INPUT

P: Probability parameter

OUTPUT

LBERN: .TRUE. with probability **P** and .FALSE. with probability 1-**P**

USAGE

user_var = LBERN(P)

Last Updated: 4th June, 1997

Section 2: Beta Random Deviates

These functions and routines are used to draw BETA random deviates and constants of integration.

BET(N,R,P)

This double precision subroutine generates one realization from a multivariate beta , or Dirichlet, distribution. The kernel density of this distribution is

$$p_1^{r_1-1} \dots p_n^{r_n-1}.$$

INPUTS

N: Order of the distribution

R: An N×1 vector where R(i) = r_i

OUTPUT

P: An N×1 vector containing one realization from a Beta(N,R) distribution , where P(i) = p_i.

USAGE

call BET(N,R,P)

Last Updated: 29 January, 1997

BETK(N,R,P)

This double precision function evaluates the log density kernel of a vector **P** with a multivariate Beta(N,R) distribution. The density kernel is

$$p_1^{r_1-1} \cdots p_n^{r_n-1}.$$

INPUTS

N: The order of the distribution

R: An N×1 vector such that R(i) = r_i

P: An N×1 vector such that P(i) = p_i

OUTPUT

BETK: The value of the log density kernel

USAGE

user_var=BETK(N,R,P)

Last Updated: 29 January, 1997

BETN(N,R)

This double precision function evaluates the log normalizing constant for the log density kernel of a random variate computed in the routine **BET**. This normalizing constant, when multiplied by the kernel in **BETK**, yields the p.d.f. of a random vector $p \sim \text{Beta}(N,R)$.

INPUTS

N: Order of the distribution

R: An $N \times 1$ vector such that $R(i) = r_i$

OUTPUT

BETN: The value of the log density kernel

USAGE

user_var = BETN(N,R)

Last Updated: 29 January, 1997

Section 3 : Chi-Square Random Deviates

CHI(S2,ANU)

This double precision real function generates a random variable z, where

$$s^2 z \sim \chi^2(v).$$

INPUTS

S2:- Numerator parameter s^2

ANU :- degrees of freedom parameter v

OUTPUT

CHI :- random variate z

USAGE

user_var=CHI(S2,ANU)

Last Updated: 29th January, 1997

CHIK(S2,ANU,X)

This double precision real function evaluates the log density kernel of a random variable, z , where $s^2z \sim \chi^2(v)$, at the point x .

INPUTS

S2: Numerator parameter s^2

ANU: Degrees of freedom parameter, v

X: Point of evaluation

OUTPUT

CHIK: Value of log density kernel

USAGE

user_var=CHIK(S2,ANU,X)

Last Updated: 29th January

CHIN(S2,ANU)

This double precision real function evaluates the log normalizing constant for the log density kernel of a random variate computed in CHI. This normalizing constant, when multiplied by the kernel in CHIK, yields the p.d.f. of a random variate z such that $s^2z \sim \chi^2(v)$.

INPUTS

S2: Numerator parameter s^2

ANU: Degree of freedom parameter v

**INPUTS VIA COMMON BLOCK /CONST/ DLOG2, DLGPI ,HLG2PI,
PI,PII**

DLOG2: $\log(2)$

DLGPI: $\log(2\pi)$

HLG2PI: $\frac{1}{2}\log(2\pi)$

PI: π

PII: π^{-1}

NB: The common block /CONST/ is initialized by the routine INIT.

OUTPUT

CHIN: The value of the log normalizing constant of the log density of z

USAGE

user_var=CHIN(S2,ANU)

Last updated: 29th January, 1997

CHIO(N,S2,ANU,NITER,CHI)

This double precision subroutine draws, using simple rejection, from N independent χ^2 distributions, subject to the restriction that the realizations are monotonically nondescending.

INPUTS

N : Number of χ^2 distributions

S2: An $N \times 1$ vector containing the constants for the $N \chi^2$ distributions

ANU: An $N \times 1$ vector containing the degrees-of-freedom parameters for the N χ^2 distributions

NITER: Number of Gibbs iterations to perform

INPUT/OUTPUT

CHI: An $N \times 1$ vector of realizations from the N χ^2 distributions in nondecreasing order

USAGE

call CHIO(N,S2,ANU,NITER,CHI)

Last Updated: 29th January, 1997

CHIO1(N,S2,ANU,NITER,CHI)

This double precision subroutine draws from N independent χ^2 distributions, subject to the restriction that the realizations are monotonically nondecreasing.

INPUTS

N: Number of distributions

S2: $N \times 1$ vector of constant parameters for the distributions

ANU: $N \times 1$ vector of degrees-of-freedom parameters for the distributions

NITER: Number of Gibbs iterations to perform

INPUT/OUTPUT

CHI: $N \times 1$ random vector containing drawings from χ^2 distributions

USAGE

call CHIO1(N,S2,ANU,NITER,CHI)

Last Updated: 4th June , 1997

CHIOP(N,S2,ANU,NITER,JFIX,CHI)

This double precision subroutine draws from N independent χ^2 distributions, subject to the restriction that the realizations are monotonically nondecreasing. Also, one of the “chi-squares” is fixed at a preset value.

INPUTS

N: Number of distributions

S2: N×1 vector of constant parameters for the distributions

ANU: N×1 vector of degree-of-freedom parameters for the distributions

NITER: Number of Gibbs iterations to perform

JFIX: Value of (pre-) fixed precision

INPUT/OUTPUT

CHI: N×1 random vector containing the drawings from the N χ^2 distributions

USAGE

call CHIOP(N,S2,ANU,NITER,JFIX,CHI)

Last Updated: 4th June , 1997

XCHIT(S2,ANU,A,B,LB)

This double precision real function generates a random deviate from a doubly truncated χ^2 distribution where $s^2h \sim \chi^2(v)$, $a < h < b$.

INPUTS

S2: The distribution parameter, s^2
ANU: Degrees of freedom parameter, ν
A: Lower bound
B: Upper bound if **LB**=**FALSE**.
LB: If **LB**=**TRUE**., then there is no upper bound

OUTPUT

XCHIT: Random variate h such that $s^2 h \sim \chi^2(\nu)$, $a < h < b$

USAGE

user_var= XCHIT(S2,ANU,A,B,LB)

Last Updated: 29th January, 1997

Section 4: Gaussian Random Deviates

GAU(K,AMU,H,LDH,X1,X2)

This double precision subroutine generates two antithetic pseudo-random vectors x_1 and x_2 from a Gaussian distribution with mean vector μ and precision matrix **H**.

INPUTS

K : Dimension of distribution
AMU : A $k \times 1$ mean vector μ
H : A $k \times k$ precision matrix **H**
LDH : Leading dimension of **H** as defined in calling routine

OUTPUTS

\mathbf{X}_1 : A random $k \times 1$ vector drawn from $N(\mu, \mathbf{H}^{-1})$

\mathbf{X}_2 : A random $k \times 1$ vector drawn from $N(\mu, \mathbf{H}^{-1})$

USAGE

call GAU(K,AMU,H,LDH,X1,X2)

Last Updated: 29th January, 1997

GAU1MX(NMIX,AMU,SIG,P)

This double precision real function randomly draws from one of a number of Normal distributions, where the probabilities are proportional to the vector p .

INPUTS

NMIX : Number of Normal distributions to draw from

AMU : An $\text{NMIX} \times 1$ vector of means $\mu = (\mu_1, \dots, \mu_{\text{NMIX}})'$

SIG : An $\text{NMIX} \times 1$ vector of variances $\sigma^2 = (\sigma_1^2, \dots, \sigma_{\text{NMIX}}^2)$

P : An $\text{NMIX} \times 1$ vector of probabilities

OUTPUT

GAU1MX : A draw from $N(\mu_j, \sigma_j^2)$ for some j

USAGE

user_var=GAU1MX(NMIX,AMU,SIG,P)

Last Updated: 29th January, 1997

GAUA(K,AMU,H,LDH,X1,X2)

This double precision subroutine uses the subroutine URNMVN to draw two antithetic vectors from a normal distribution with mean μ and variance \mathbf{HH}' .

INPUTS

K : Order of distribution

AMU : The $k \times 1$ mean vector of the distribution, μ

H : Upper triangular $k \times k$ matrix such that $\mathbf{x} \sim N(\mu, \mathbf{HH}')$

LDH : Leading dimension of **H** as defined in the calling program

OUTPUT

X1 and X2 : Two antithetic $k \times 1$ vectors drawn from $N(\mu, \mathbf{HH}')$

USAGE

call GAUA(K,AMU,H,LDH,X1,X2)

Last Updated: 29th January, 1997

GAUB(K,AMU,R,LDR,X1,X2)

This double precision subroutine uses URNMVN to draw two antithetic vectors from a normal distribution with mean μ and variance-covariance matrix \mathbf{RR}' .

INPUTS

K : Order of distribution

AMU : The $k \times 1$ mean vector of the distribution, μ

R : Upper triangular $k \times k$ matrix such that $\mathbf{x} \sim N(\mu, \mathbf{RR}')$

LDR : Leading dimension of **R** as defined in the calling program

OUTPUT

X1 and X2 : Two antithetic $k \times 1$ vectors drawn from $N(\mu, \mathbf{RR}')$

USAGE

call GAUA(K,AMU,R,LDR,X1,X2)

Last Updated: 4th June, 1997

GAUI(NP,AMU,H,LDH,BETA)

This double precision subroutine generates a random vector from a Gaussian distribution with mean vector μ and precision matrix \mathbf{H} , subject to the restriction that the vector corresponds to an invertable lag operator.

INPUTS

NP : The Dimension of the distribution

AMU : The $NP \times 1$ mean vector μ

H : The $NP \times NP$ precision matrix \mathbf{H} for the distribution

LDH : Leading dimension of \mathbf{H} as defined in the calling program

OUTPUT

BETA : An $NP \times 1$ random vector drawn from $N(\mu, \mathbf{H}^{-1})$

USAGE

call GAUI(NP,AMU,H,LDH,BETA)

Last Updated: 29th January, 1997

GAUI1(NP,AMU,H,LDH,BETA)

This double precision subroutine makes draws from $N(\mu, \mathbf{H}^{-1})$ using independence Metropolis with uniform p.a.c.f. source.

INPUTS

NP : Order of distribution

AMU : The $NP \times 1$ mean vector, μ

H : The $NP \times NP$ precision matrix \mathbf{H}

LDH : The leading dimension of \mathbf{H} as defined in the calling program

OUTPUT

BETA : An $NP \times 1$ random vector drawn from $N(\mu, \mathbf{H}^{-1})$

USAGE

call GAUI1(NP,AMU,H,LDH,BETA)

Last Update: 4th June, 1997

GAUI2(NP,AMU,H,LDH,BETA,WTL)

This double precision subroutine makes draws from $N(\mu, \mathbf{H}^{-1})$ using independence Metropolis with uniform p.a.c.f. source and provides importance sampling weights.

INPUTS

NP : Order of distribution

AMU : The $NP \times 1$ mean vector, μ

H : The $NP \times NP$ precision matrix \mathbf{H}

LDH : The leading dimension of \mathbf{H} as defined in the calling program

OUTPUT

BETA : An $NP \times 1$ random vector drawn from $N(\mu, \mathbf{H}^{-1})$

WTL : Log weight, for importance sampling

USAGE

call GAUI1(NP,AMU,H,LDH,BETA,WTL)

Last Updated: 4th June, 1997

GAUIN(NP,AMU,H,LDH,SD,TIME)

This double precision real function evaluates the log normalizing constant for the log density kernel of a random vector computed in GAUK, given an invertibility condition. This normalizing constant, when multiplied by the kernel in GAUK, yields the p.d.f. of the random vector.

INPUTS

NP : Order of distribution

AMU : The $NP \times 1$ mean vector, μ

H : The $NP \times NP$ precision matrix \mathbf{H}

LDH : The leading dimension of \mathbf{H} as defined in the calling program

SD: Routine stops when the standard deviation of the approximation error of log kernel reaches this point

TIME : Routine stops when this much CPU time has been used

OUTPUT

GAUIN : The log normalizing constant (approximate)

SD : Standard error of the approximation

TIME : CPU time used

USAGE

user_var= GAUIN(NP,AMU,H,LDH,SD,TIME)

Last Updated: 4th June, 1997

GAUK(K,AMU,H,LDH,X)

This double precision real function evaluates the log density kernel of a multivariate normal random vector $\mathbf{x} \sim N(\boldsymbol{\mu}, \mathbf{H}^{-1})$. The kernel density is

$$\exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})' \mathbf{H}(\mathbf{x} - \boldsymbol{\mu})\right)$$

INPUTS

K: Dimension of vector \mathbf{x}

AMU : The $k \times 1$ mean vector $\boldsymbol{\mu}$

H : The $k \times k$ precision matrix \mathbf{H}

LDH : The leading dimension of \mathbf{H} as defined in the calling program

X : Point of evaluation

OUTPUT

GAUK : The log density kernel of $N(\boldsymbol{\mu}, \mathbf{H}^{-1})$ evaluated at \mathbf{x}

USAGE

user_var=GAUK(K,AMU,H,LDH,X)

Last Updated: 29th January, 1997

GAUN(K,H,LDH)

This double precision real function evaluates the log normalizing constant for the log density kernel of a random vector computed in GAUK. This normalizing constant, when multiplied by the kernel in GAUK, yields the p.d.f of the random vector.

INPUTS

K : Dimension of the random vector \mathbf{x} whose log density kernel is evaluated in GAUK

H : The $k \times k$ precision matrix

LDH : The leading dimension of **H**

OUTPUT

GAUN : The log normalizing factor

USAGE

```
user_var = GAUN(K,H,LDH)
```

Comments:

This function uses constants declared in the common block /CONST/ which is initialized by the subroutine INIT.

Last Updated: 29th January, 1997

GAUT(N,AMU,H,LDX,D,DINV,LDD,A,B,LA,LB,LE,NITER,X)

This double precision subroutine draws from a truncated multivariate normal distribution without rejection. The truncation is formed by linear restrictions equal in number to the dimension of the distribution. The bounds $+\infty$ and $-\infty$ are allowed. The truncated distribution is

$$\mathbf{x} \sim \mathbf{N}(\boldsymbol{\mu}, \mathbf{H}^{-1}), \quad \mathbf{A} < \mathbf{D}\mathbf{x} < \mathbf{B}.$$

INPUTS

N: Order of the multivariate normal distribution

AMU: Mean of the distribution, $\boldsymbol{\mu}$

H: The $N \times N$ precision matrix for the distribution

LDX: The leading dimension for the matrices **H** as defined in the calling program

D: The $N \times N$ matrix of linear constraints for \mathbf{x}

DINV: The inverse of **D**

LDD: Leading dimension of **D** and **DINV** as defined in the calling program

A: The $N \times 1$ vector of lower bounds for the linear combinations

B: The $N \times 1$ vector of upper bounds for the linear combinations

LA: An $N \times 1$ logical vector where $LA(j) = .TRUE.$ implies no lower bound for x_j

LB: An $N \times 1$ logical vector where $LB(j) = .TRUE.$ implies no upper bound for x_j

LE: An $N \times 1$ logical vector where $LE(j) = .TRUE.$ implies an equality constraint for x_j

NITER: Number of Gibbs iterations to perform

OUTPUT

X: An $N \times 1$ random vector drawn from $\mathbf{N}(\boldsymbol{\mu}, \mathbf{H}^{-1})$ subject to

$\mathbf{A} < \mathbf{D}\mathbf{x} < \mathbf{B}$

USAGE

call GAUT(N,AMU,H,LDX,D,DINV,LDD,A,B,LA,LE,NITER,X)

Reference:

J. Geweke, 1991. "Efficient Simulation from the Multivariate Normal and Student-t Distribution Subject to Linear Constraints," in E. M. Keramidas (ed.), *Computing Science and Statistics: Proceedings of the 23rd Symposium on the Interface*, pp. 571-578. Fairfax, VA: Interface Foundation of North America, Inc.

Last Updated: 4th June, 1997

GAUTK(N,AMU,H,LDX,D,DINV,LDD,A,B,LA,LE,NITER,X)

This double precision real function returns the log kernel of a truncated normal distribution. If the constraints are satisfied then the kernel is the same as for the untruncated distribution. If they are violated, a small negative number is returned.

INPUTS

N: Order of the multivariate normal distribution

AMU: Mean of the distribution, μ

H: The $N \times N$ precision matrix for the distribution

LDX: The leading dimension for the matrices **H** as defined in the calling program

D: The $N \times N$ matrix of linear constraints for **x**

DINV: The inverse of **D**

LDD: Leading dimension of **D** and **DINV** as defined in the calling routine

A: The $N \times 1$ vector of lower bounds for the linear combinations

B: The $N \times 1$ vector of upper bounds for the linear combinations

LA: An $N \times 1$ logical vector where $LA(j) = .TRUE.$ implies no lower bound for x_j

LB: An $N \times 1$ logical vector where $LB(j) = .TRUE.$ implies no upper bound for x_i

LE: An $N \times 1$ logical vector where $LE(j) = .TRUE.$ implies an equality constraint for x_i

NITER: Number of Gibbs iterations to perform

X: An $N \times 1$ vector where the log kernel is to be evaluated

OUTPUT

GAUTK : The log kernel evaluated at \mathbf{x}

USAGE

```
user_var= GAUTK(N,AMU,H,LDX,D,DINV,LDD,A,B,LA,  
LB,LE,NITER,X)
```

Last Updated: 4th June, 1997.

GAUTN(N,AMU,H,LDX,D,DINV,LDD,A,B,LA,LB,LE,NITER)

This double precision real function returns the log normalizing factor of a truncated normal distribution. The normalizing factor, when multiplied by the kernel evaluated in GAUTK, yields the properly normalized probability density of the truncated normal distribution. The normalizing factor is the one for the untruncated normal, divided by the probability that the untruncated model would satisfy all of the inequalities in the truncated normal distribution. This probability is approximated using the GHK simulation algorithm.

INPUTS

N: Order of the multivariate normal distribution

AMU: Mean of the distribution, μ

H: The $N \times N$ precision matrix for the distribution

LDX: The leading dimension for the matrices **H** as defined in the calling program

D: The $N \times N$ matrix of linear constraints for \mathbf{x}

DINV: The inverse of **D**

LDD: Leading dimension of **D** and **DINV** as defined in the calling routine

A: The $N \times 1$ vector of lower bounds for the linear combinations

B: The $N \times 1$ vector of upper bounds for the linear combinations

LA: An $N \times 1$ logical vector where $LA(j) = .TRUE.$ implies no lower bound for x_j

LB: An $N \times 1$ logical vector where $LB(j) = .TRUE.$ implies no upper bound for x_j

LE: An $N \times 1$ logical vector where $LE(j) = .TRUE.$ implies an equality constraint for x_j

NITER: Number of Monte Carlo draws to use in the GHK algorithm

OUTPUT

GAUTN: The log normalizing factor of the truncated normal distribution

USAGE

```
user_var=GAUTN(N,AMU,H,LDX,D,DINV,LDD,A,B,LA,LE,
NITER)
```

Last Updated: 4th June, 1997

GHK(N,AMU,H,LDX,D,DINV,LDD,A,B,LA,LE,NITER,SE)

This double precision real function uses the GHK algorithm to approximate the probability that a random variable with the distribution $\mathbf{x} \sim N(\boldsymbol{\mu}, \mathbf{H}^{-1})$ satisfies the inequality constraints $\mathbf{A} \leq \mathbf{D}\mathbf{x} \leq \mathbf{B}$.

INPUTS

N: Order of the multivariate normal distribution

AMU: Mean of the distribution, $\boldsymbol{\mu}$

H: The $N \times N$ precision matrix for the distribution

LDX: The leading dimension for the matrices **H** as defined in the calling program

D: The $N \times N$ matrix of linear constraints for \mathbf{x}

DINV: The inverse of **D**

LDD: Leading dimension of **D** and **DINV** as defined in the calling routine

A: The $N \times 1$ vector of lower bounds for the linear combinations

B: The $N \times 1$ vector of upper bounds for the linear combinations

LA: An $N \times 1$ logical vector where $LA(j) = .TRUE.$ implies no lower bound for x_j

LB: An $N \times 1$ logical vector where $LB(j) = .TRUE.$ implies no upper bound for x_j

LE: An $N \times 1$ logical vector where $LE(j) = .TRUE.$ implies an equality constraint for x_j

NITER: Number of Monte Carlo draws to use in the GHK algorithm

OUTPUT

GHK: The probability approximation

SE: The standard error of the probability approximation

USAGE

`user_var=GHK(N,AMU,H,LDX,D,DINV,LDD,A,B,LA,LB,LE,NITER,`

SE)

Last Updated: 4th June, 1997

PNORDF(AMU,H,X)

This double precision real function evaluates the c.d.f. of a univariate normal.

INPUTS

AMU: Mean of normal distribution

H: Precision of normal distribution

X: Point of evaluation

OUTPUT

PNORDF: Value of c.d.f at **X**

USAGE

user_var = PNORDF(AMU,H,X)

Last Updated: 4th June, 1997

URNMVN(K,U,LDU,X)

This double precision subroutine generates one pseudo-random vector from a multivariate normal distribution with mean **0** and variance **UU'**.

INPUTS

K: Dimension of distribution

U: Upper triangular **K** \times **K** matrix such that $\mathbf{x} \sim N(\mathbf{0}, \mathbf{UU}')$

LDU: Leading dimension of **U** as defined in the calling program

OUTPUT

X: A $K \times 1$ random vector drawn from $N(\mathbf{0}, \mathbf{UU}')$

USAGE:

call URNMVN(K,U,LDU,X)

Last Updated: 29th January, 1997

XNOT(AMU,H,A,B,LA,LB)

This double precision real function generates a univariate normal random variable subject to the constraint that it be in an interval (a,b), where the endpoints may be finite or infinite.

INPUTS

AMU: Mean of the parent univariate normal distribution, μ

H: Precision of the parent univariate normal distribution

A,B: Endpoints of the interval; $A < B$ if $LA=LB=.FALSE.$

LA: Logical whose value is **.TRUE.** if the left endpoint is $-\infty$; in this case A is ignored

LB: Logical whose value is **.TRUE.** if the right endpoint is ∞ ; in this case B is ignored

OUTPUT

XNOT: Univariate random variate drawn from $N(\mu, h^{-1})$

USAGE

user_var= XNOT(AMU,H,A,B,LA,LB)

Reference:

J. Geweke, 1991. "Efficient Simulation from the Multivariate Normal and Student-t Distribution Subject to Linear Constraints," in E. M. Keramidas (ed.), *Computing Science and Statistics: Proceedings of the 23rd Symposium on the Interface*, pp 571-578. Fairfax, VA: Interface Foundation of North America, Inc.

Last Update: 29th January, 1997

XNOTK(AMU,H,A,B,LA,LB,X)

This double precision real function returns the log density kernel of a univariate normal distribution truncated to the interval (a,b). If x is in the interval the kernel is $-h(x-\mu)^2$; otherwise it is zero.

INPUTS

AMU: Mean of the parent univariate normal distribution, μ

H: Precision of the parent univariate normal distribution

A,B: Endpoints of the interval; $A < B$ if $LA=LB=.FALSE.$

LA: Logical whose value is **.TRUE.** if the left endpoint is $-\infty$; in this case A is ignored

LB: Logical whose value is **.TRUE.** if the right endpoint is ∞ ; in this case B is ignored

X: Point of evaluation of kernel

OUTPUT

XNOTK: Log density kernel evaluated at x

USAGE

user_var= XNOTK(AMU,H,A,B,LA,LB,X)

Last Updated: 29th January, 1997

XNOTN(AMU,H,A,B,LA,LB)

This double precision real function returns log normalizing factor of a univariate normal distribution truncated to the interval (a,b). The normalizing factor corresponds to the log kernel evaluation in XNOTK.

INPUTS

AMU: Mean of the parent univariate normal distribution, μ

H: Precision of the parent univariate normal distribution

A,B: Endpoints of the interval; $A < B$ if $LA=LB=.FALSE.$

LA: Logical whose value is `.TRUE.` if the left endpoint is $-\infty$; in this case A is ignored

LB: Logical whose value is `.TRUE.` if the right endpoint is ∞ ; in this case B is ignored

OUTPUT

XNOTN: Log normalizing factor

USAGE

user_var= XNOTN(AMU,H,A,B,LA,LB)

Last Updated: 29th January, 1997

Section 5: Student-t Random deviates

This section contains routines used to

draw random deviates from a Student-t distribution.

STT(K,AMU,H,LDH,ANU,X1,X2)

This double precision subroutine generates two, antithetic, pseudo-random vectors x_1 and x_2 from a Student-t distribution with mean vector μ and precision matrix \mathbf{H} .

INPUTS

K: Dimension of distribution

AMU: $K \times 1$ mean vector μ

H: $K \times K$ precision matrix

LDH: Leading dimension of \mathbf{H} as defined in the calling routine

ANU: Degrees-of-freedom parameter

OUTPUTS

X1,X2: Two antithetic drawings from Student-t distribution

USAGE

call STT(K,AMU,H,LDH,ANU,X1,X2)

Last Updated: 4th June, 1997

STTA(K,AMU,H,LDH,ANU,X1,X2)

This double precision subroutine generates two, antithetic, pseudo-random vectors x_1 and x_2 from a Student-t distribution with mean vector μ and precision matrix \mathbf{H} where the precision matrix \mathbf{H} is already factored.

INPUTS

K: Dimension of distribution

AMU: $K \times 1$ mean vector μ

H: factored $K \times K$ precision matrix

LDH: Leading dimension of **H** as defined in the calling routine

ANU: Degrees-of -freedom parameter

OUTPUTS

X1,X2: Two antithetic drawings from Student-t distribution

USAGE

call STTA(K,AMU,H,LDH,ANU,X1,X2)

Last Updated: 4th June, 1997

STTB(K,AMU,R,LDR,ANU,X1,X2)

This double precision subroutine generates two, antithetic, pseudo-random vectors x_1 and x_2 from a Student-t distribution with mean vector μ and variance matrix RR' where **R** is upper triangular.

INPUTS

K: Dimension of distribution

AMU: $K \times 1$ mean vector μ

R: $K \times K$ upper triangular matrix such that the variance matrix is RR'

LDR: Leading dimension of **R** as defined in the calling routine

ANU: Degrees-of -freedom parameter

OUTPUTS

X1,X2: Two antithetic drawings from Student-t distribution

USAGE

call STTB(K,AMU,B,LDB,ANU,X1,X2)

Last Updated: 4th June, 1997

STTK(K,AMU,H,LDH,ANU,X)

This double precision real valued function evaluates the log density kernel of a multivariate Student-t random vector \mathbf{X} with mean μ and precision matrix \mathbf{H} . The kernel density is

$$\left[1 + \frac{1}{2\nu} (\mathbf{X} - \mu)' \mathbf{H} (\mathbf{X} - \mu) \right]^{-\frac{(\nu+k)}{2}}$$

INPUTS

K: Dimension of distribution

AMU: K×1 mean vector μ

H: K×K precision matrix

LDH: Leading dimension of \mathbf{H} as defined in the calling routine

ANU: Degrees-of -freedom parameter

X: Point of evaluation

OUTPUT

STTK: Evaluation of log density kernel

USAGE

user_var = STTK(K,AMU,H,LDH,ANU,X)

Last Updated: 4th June ,1997

STTN(K,H,LDH,ANU)

This double precision real valued function evaluates the log normalizing constant for the log density kernel of a random vector computed in STTK. This normalizing constant, when multiplied by the kernel in STTK, yields the p.d.f. of the random vector.

INPUTS

K: Dimension of multivariate Student-t distribution

H: K×K precision matrix

LDH: Leading dimension of **H** as defined in the calling program

ANU: Degrees-of-freedom parameter

OUTPUT

STTN: Log normalizing constant

USAGE

user_var = STTN(K,H,LDH,ANU)

Last Updated: 4th June, 1997

Section 6: Wishart Random Deviates

This section contains routines used in drawing random matrices from a Wishart distribution with matrix parameter Σ and degrees of freedom parameter ν .

WISH(M,SIG,LDSIG,ANU,A,LDA)

This double precision subroutine generates a pseudo-random $M \times M$ matrix from a Wishart distribution with matrix parameter Σ and degrees of freedom parameter ν .

INPUTS

M: The order of the matrix

SIG: Matrix parameter, Σ

LDSIG: The leading dimension of Σ as defined in the calling program

ANU: The degrees of freedom parameter, ν

LDA: The leading dimension of **A** as defined in the calling program

OUTPUTS

A: Random $M \times M$ matrix drawn from $W(\Sigma, \nu)$

USAGE

call WISH(M,SIG,LDSIG,ANU,A,LDA)

Last Updated: 29th January, 1997

WISHA(M,SIG,LDSIG,ANU,A1,LD)

This double precision subroutine performs the same role as WISH except the matrix parameter is already factored.

INPUTS

M: The order of the matrix

SIG: Matrix parameter, Σ

LDSIG: The leading dimension of Σ as defined in the calling program

ANU: The degrees of freedom parameter, ν

LDA: The leading dimension of **A** as defined in the calling program

OUTPUTS

A: Random $M \times M$ matrix drawn from $W(\Sigma, \nu)$

USAGE

call WISHA(M,SIG,LDSIG,ANU,A,LDA)

Last Updated: 4th June ,1997

WISHB(M,R,LDR,ANU,A,LDA)

This double precision subroutine performs the same role as WISH except the matrix parameter is now $\mathbf{R}'\mathbf{R}$ where **R** is upper triangular with zeros set below.

INPUTS

M: The order of the matrix

R: Matrix parameter, such that $\Sigma = \mathbf{R}'\mathbf{R}$

LDR: The leading dimension of **R** as defined in the calling program

ANU: The degrees of freedom parameter, ν

LDA: The leading dimension of **A** as defined in the calling program

OUTPUTS

A: Random $M \times M$ matrix drawn from $W(\Sigma, \nu)$

USAGE

call WISHB(M,R,LDR,ANU,A,LDA)

Last Updated: 4th June ,1997

WISHK(M,P,LDP,ANU,A,LDA)

This double precision real function evaluates the log kernel of a Wishart random matrix **A** with matrix parameter Σ , and degrees of freedom parameter ν . The log density kernel is

$$\frac{1}{2}(\nu + 1 - m)\log|\mathbf{A}| - \frac{1}{2}\text{tr}(\Sigma^{-1}\mathbf{A})$$

INPUTS

M: Order of matrix, m

P: An $m \times m$ matrix such that $\mathbf{PP}' = \Sigma^{-1}$

LDP: Leading dimension of **P** as defined in the calling program

ANU: Degrees of freedom parameter, ν

A: Matrix point of evaluation of log density kernel

LDA: Leading dimension of **A** as defined in the calling program

OUTPUT

WISHK: Evaluation of the log density kernel for $\mathbf{A} \sim W(\Sigma, \nu)$

USAGE

user_var= WISHK(M,P,LDP,ANU,A,LDA)

Last Updated: 29th January, 1997

WISHN(M,SIG,LDSIG,ANU,P,LDP)

This double precision real function evaluates the log normalizing constant for the log density kernel of a random matrix computed in WISHK. This normalizing constant, when multiplied by the kernel in WISHK, yields the p.d.f. of the random matrix.

INPUTS

M: Order of Matrix, m

SIG: Matrix parameter, Σ

LDSIG: Leading dimension of Σ as defined in the calling program

ANU: Degrees of freedom parameter, ν

LDP: Leading dimension of \mathbf{P} as defined in the calling program

OUTPUTS

WISHN: Log normalizing factor

P: Upper triangular $m \times m$ matrix such that $\mathbf{PP}' = \Sigma^{-1}$

USAGE

user_var= WISHN(M,SIG,LDSIG,ANU,P,LDP)

Last Updated: 29th January, 1997

Section 7: General Random Deviates

YLC(G,G1,G2,D1,D2,LXHAT,XHAT)

This double precision real function draws from a distribution whose log density is strictly concave.

INPUTS

G: Function $G(x)$ that has input ordinate x and returns the log density kernel, G

G1: Function $G1(x)$ that has input ordinate x and returns the first derivative of the log density kernel, $G1$
G2: Function $G2(x)$ that has input ordinate x and returns the second derivative of the log density kernel, $G2$
D1: Lower end of finite support
D2: Upper end of finite support
LXHAT: Logical that has value `.TRUE.` if the mode of log support is supplied as an input

INPUT/OUTPUT

XHAT: Mode of the log density kernel

OUTPUT

YLC: Draw from distribution

USAGE

`user_var= YLC(G,G1,G2,D1,D2,LXHAT,XHAT)`

Last Updated: 4th June, 1997

YUM(F,C1,C2,XSTAR,D2LFG)

This double precision function draws from a distribution with a unimodal density on a finite support.

INPUTS

F: User defined function that has input ordinate x and returns density kernel F
C1: Lower end of finite support
C2: Upper end of finite support

XSTAR: Mode of density kernel

D2LGF: Second derivative of log density kernel at mode

OUTPUT

YUM: Drawing from unimodal distribution

USAGE

user_var= YUM(F,C1,C2,XSTAR,D2LGF)

Last Updated: 29th January, 1997

Section 8: General Routines

QUANT(N,NQ,W,G,P,Q)

This double precision subroutine computes quantiles from the sampled distribution of a random variable.

INPUTS

N : Size of sample

NQ : Number of quantiles to compute

W : An $N \times 1$ vector of log weights

G : An $N \times 1$ vector of sampled random variables

P : An $NQ \times 1$ vector of probabilities corresponding to quantiles

OUTPUTS

Q : An $NQ \times 1$ vector of quantiles

W : An $N \times 1$ vector of actual weights, unless all log weights were 0

G : An $N \times 1$ vector containing the ordered random variables

USAGE

call QUANT(N,NQ,W,G,P,Q)

Last Updated: 29th January, 1997

CHAPTER 4: SIMULATION ROUTINES

This chapter contains routines that simulate from the posteriors of some common models. It also contains input/output routines for reading in priors and writing information about the priors to a file.

Section 1: Input/Output Routines

BET0(NFILE,N,TITLE,M,R,LDR,ANORM,P,LDP)

This double precision subroutine reads in the parameters for multiple multivariate Beta distributions. The routine prints out information about the distributions, computes and returns normalizing constants for the densities, and one draw from each distribution.

INPUTS

NFILE: File unit number for reading input information

N: Number of multivariate Beta distributions to draw

TITLE: Information to be printed at start

M: Number of categories for each multivariate Beta distribution

LDR: Leading dimension of R matrix, the matrix where each row, i , contains the M parameters of distribution i .

LDP: Leading dimension of P matrix

NB: This routine reads in the parameters for the M multivariate Beta distributions from the file whose unit number is declared in **NFILE**. The file contains M records that are read into the matrix **R**.

OUTPUTS

R: An $N \times M$ matrix containing the M parameters of each of the N multivariate Beta distributions.

ANORM: The log normalizing constant for the joint multivariate Beta distribution

P: An $N \times M$ matrix where each row, i , contains the drawing from distribution i .

USAGE

call BET0(NFILE,N,TITLE,M,R,LDR,ANORM,P,LDP)

Last Updated: 29 January, 1997

CHI0(NFILE,N,TITLE,LORD,ANU0,S20,ANORM,ACHI)

This double precision subroutine reads in the constants and degrees-of-freedom parameters of multiple chi-square distributions. It prints out information about the distributions, computes and returns normalizing constants for the densities, and one draw from each distribution.

INPUTS

NFILE:- File number for reading input information

N:- Number of χ^2 distributions

TITLE:- Information printed at start

LORD:- If .TRUE., the drawings must be in ascending order

NB: This routine reads in the degree-of -freedom parameter and the numerator parameter for each distribution from the file whose unit number is declared in NFILE. The file consists of N records, corresponding to each distribution, and each record consists of the degree-of -freedom parameter and the numerator parameter for each respective distribution.

OUTPUTS

ANU0: An $N \times 1$ vector of degrees-of-freedom parameters

S20: An $N \times 1$ vector of constant parameters

ANORM: Log normalizing constant for the joint χ^2 distribution

ACHI: An $N \times 1$ vector of draws from the $N \chi^2$ distributions

USAGE

call `CHI0(NFILE,N,TITLE,ANU0,S20,ANORM,ACHI)`

Last updated: 4th June, 1997

CHI0P(NFILE,N,TITLE,LORD,JFIX,HFIX,ANU0,S20,ANORM,ACHI)

This double precision subroutine reads in the constants and degrees-of-freedom parameters for multiple χ^2 distributions. It prints out information about the distributions, computes and returns normalizing constants for the densities, and one draw from each distribution. One of the draws is fixed at a pre-set value.

INPUTS

NFILE: File number for reading input information

N: Number of χ^2 distributions

TITLE: Information printed at start

LORD: If .TRUE., drawings must be in ascending order

JFIX: Precision to be fixed

HFIX: Value of fixed precision

OUTPUTS

ANU0: Degrees-of-freedom parameters

S20: Constant parameters

ANORM: Log normalizing constant for joint χ^2 distribution

ACHI: Drawings from the distribution

USAGE

call CHI0P(NFILE,N,TITLE,LORD,JFIX,HFIX,ANU0,S20,ANORM,
ACHI)

Last Updated: 4th June, 1997

GAU0(NFILE,NK,TITLE,H0,LDH,B0,HB0,ANORM,THETA)

This double precision subroutine reads in the mean vector μ and precision matrix \mathbf{H} of a $k \times 1$ parameter vector θ . It computes and returns the product of \mathbf{H} and μ , prints information about, and makes one draw from the prior using the routine GAU.

INPUTS

NFILE: File number for reading prior

NK: Dimension of parameter vector, k

TITLE: Information to be printed at top of output file

LDH: Leading dimension of H0 as defined in the calling program

NB. This routine reads in the mean vector and precision matrix from the file assigned to unit NFILE. The first record of the file contains the vector μ and the next k records contain the k rows of \mathbf{H} .

OUTPUTS

H0: The $k \times k$ prior precision matrix \mathbf{H}

B0: The $k \times 1$ prior mean vector μ

HB0: The $k \times 1$ vector formed by the product of **H** and μ

ANORM: The log normalizing constant for the prior density

THETA: A draw from the prior

USAGE

call GAU0(NFILE,NK,TITLE,H0,LDH,B0,HB0,ANORM,THETA)

Last Updated: 4th June, 1997

GAUI0(NFILE,NK,TITLE,H0,LDH,B0,HB0,ANORM,THETA)

This double precision subroutine reads in the mean vector μ and precision matrix **H** of a $k \times 1$ parameter vector θ . It computes and returns the product of **H** and μ , prints information about, and makes one draw from the prior using the routine GAUI.

INPUTS

NFILE: File number for reading prior

NK: Dimension of parameter vector, k

TITLE: Information to be printed at top of output file

LDH: Leading dimension of H0 as defined in the calling program

NB. This routine reads in the mean vector and precision matrix from the file assigned to unit NFILE. The first record of the file contains the vector μ and the next k records contain the k rows of **H**.

OUTPUTS

H0: The $k \times k$ prior precision matrix **H**

B0: The $k \times 1$ prior mean vector μ

HB0: The $k \times 1$ vector formed by the product of **H** and μ

ANORM: The log normalizing constant for the prior density

THETA: A draw from the prior

USAGE

call GAUI0(NFILE,NK,TITLE,H0,LDH,B0,HB0,ANORM,THETA)

Last Updated: 4th June, 1997

**NMXD0(NFILE,TITLE,NMIX,IDCODE,MXCODE,AH0,LDMIX,AB0,AHB0,
ALPHA,ANU0,SONU0,H,R,P,PRI0)**

This double precision subroutine reads in parameters for priors for a discrete mixture of normals model. Normalization constants for truncated alpha and chi priors must be fixed. The routine prints out information about the prior distributions and makes initial draws from the prior distributions.

INPUTS

NFILE: File number of file to be read

TITLE: Information to be printed at start of output file

NMIX: Number of components of mixture

IDCODE: Orderings indicator:

=1: Mean vector Alpha in nondescending order

=2: Precision vector **H** in nondescending order

=3: Probability vector **P** in nondescending order

Otherwise, no identifying restrictions beyond prior distributions applied

MXCODE: Free parameter indicator:

=1: Scale mixture of normals (means fixed at 0)

=2: Mean mixture of normals (precisions all the same)

Otherwise, both means and precisions are free

LDMIX: Leading dimension of **AH0** as defined in the calling program

NB. This routine reads in data from the file that is assigned to unit number NFILE. The file should have the following structure. The routine uses the routines GAU0, CHI0 and BET0 to read in prior information depending on the type of model used. For the most general case where both means and precisions are free the first NMIX + 1 records contains the Gaussian prior for the vector of mixture means, the next NMIX records contain the NMIX χ^2 priors for the mixtures and the last record contains the Beta prior for the mixture probabilities. For the scale mixture case, i.e. MXCODE=1, there are no Gaussian priors read in. The rest is as in the general case. For the mean mixture case, i.e. MXCODE=2, the format is the same as the general case except there is only one χ^2 prior to be read in.

OUTPUTS

AH0: NMIX×NMIX prior precision matrix for coefficient Gaussian prior

AB0: NMIX×1 prior mean vector for coefficient Gaussian prior

AHB0: NMIX×1 vector of the product of the prior precision matrix and the prior mean vector

ALPHA: Initial draw of the mean vector

ANU0: NMIX×1 vector of degrees of freedom parameters for precision priors

S0NU0: NMIX×1 vector of constant parameters for precision priors

H: NMIX×1 vector. Initial draw of precision parameters

R: NMIX×1 vector of prior parameters for probability beta distribution

P: NMIX×1 vector. Initial draw of probability parameters

PRI0: Log normalization constant for prior

USAGE

call NMXD0(NFILE,TITLE,NMIX,IDCODE,MXCODE,
AH0,LDMIX,AB0,AHB0,ALPHA,ANU0,S0NU0,H,R,P,PRI0)

Last Updated: 29th January, 1997

**NMXD0P(NFILE, TITLE,NMIX,IDCODE,MXCODE,AH0,LDMIX,AB0,AHB0,
ALPHA,ANU0,S0NU0,H,R,P,JFIX,PRI0)**

This double precision subroutine reads in parameters for priors for a discrete mixture of normals distribution, for the case in which one precision is fixed.

INPUTS

NFILE: File number of file to be read

TITLE: Information to be printed at start of output file

NMIX: Number of components of mixture

IDCODE: Orderings indicator:

=1: Mean vector **Alpha** in nondescending order

=2: Precision vector **H** in nondescending order

=3: Probability vector **P** in nondescending order

Otherwise, no identifying restrictions beyond prior distributions applied

MXCODE: Free parameter indicator:

=1: Scale mixture of normals (means fixed at 0)

=2: Mean mixture of normals (precision's all the same)

Otherwise, both means and precision's are free

LDMIX: Leading dimension of **AH0** as defined in the calling program

NB. This routine reads in data from the file that is assigned to unit number NFILE. The file should have the following structure. The routine uses the routines GAU0, CHI0 and BET0 to read in prior information depending on the type of model used. For the most general case where both means and precision's are free

the first NMIX + 1 records contains the Gaussian prior for the vector of mixture means, the next NMIX records contain the NMIX χ^2 priors for the mixtures and the last record contains the Beta prior for the mixture probabilities. For the scale mixture case, i.e. MXCODE=1, there are no Gaussian priors read in. The rest is as in the general case. For the mean mixture case, i.e. MXCODE=2, the format is the same as the general case except there is only one χ^2 prior to be read in.

OUTPUTS

AH0: NMIX×NMIX prior precision matrix for coefficient Gaussian prior

AB0: NMIX×1 prior mean vector for coefficient Gaussian prior

AHB0: NMIX×1 vector of the product of the prior precision matrix and the prior mean vector

ALPHA: Initial draw of the mean vector

ANU0: NMIX×1 vector of degrees of freedom parameters for precision priors

S0NU0: NMIX×1 vector of constant parameters for precision priors

H: NMIX×1 vector. Initial draw of precision parameters

R: NMIX×1 vector of prior parameters for probability beta distribution

P: NMIX×1 vector. Initial draw of probability parameters

JFIX: Precision whose value is fixed

PRI0: Log normalization constant for prior

USAGE

```
call NMXD0P(NFILE,TITLE,NMIX,IDCODE,MXCODE,  
AH0,LDMIX,AB0,AHB0,ALPHA,ANU0,S0NU0,H,R,P,JFIX,PRI0)
```

Last Updated: 4th June ,1997

RDSIM(NFILE,ITER,WPP,G)

This double precision subroutine reads an iteration record from the previously opened simulation input file assigned to unit NFILE in the calling program.

INPUT

NFILE : Unit number assigned to simulation input file as defined in the calling program

INPUTS VIA COMMON /ARDSIM/

LDA : Leading dimension of **G**

NPARSA : Number of simulated values to be read

OUTPUTS

ITER : Iteration number read from first line

WPP : A 3×1 vector containing log weight, log prior and log data density read from first line

G : A NPARSA×1 vector of simulated values read from first line

Notes:

Each iteration record should contain the variables in the following order:
ITER, WPP(1),WPP(2),WPP(3),G(1),...G(NPARSA).

USAGE

call RDSIM(NFILE,ITER,WPP,G)

Last Updated: 29th January, 1997

RDSIM0(NFILE,LD,NITER,NPARS)

This double precision subroutine opens and reads the first record from the simulation output file assigned to unit NFILE.

INPUTS

NFILE : Unit number assigned to simulation file

LD : Maximum number of parameters allowed in vector **G**

OUTPUTS

NITER : Number of iterations

NPARS : Number of parameters

USAGE

call RDSIM0(NFILE,LD,NITER,NPARS)

Last Updated: 29th January, 1997

WISH0(NFILE,M,TITLE,SIG0,LDSIG,ANU0,ANORM,P,LDP,A,LDA)

This double precision subroutine reads in the matrix and degrees of freedom parameter of a $m \times m$ matrix with a Wishart distribution. It computes and returns the normalizing constant of the density, a related matrix for use in drawing from the distribution, and one draw from the distribution.

INPUTS

NFILE: File unit number of file to be read

M: Order of matrix, m

TITLE: Information to be printed at the start of output file

LDSIG: Leading dimension of SIG0 as defined in the calling program

LDP: Leading dimension of P as defined in the calling program

LDA: Leading dimension of A as defined in the calling program

NB. The data is read in from the file assigned to unit number NFILE. The first record of this file is the degrees of freedom parameter. The next m records are the m rows of the prior matrix parameter, SIG0.

OUTPUTS

SIG0: $m \times m$ prior matrix parameter, Σ_0

ANU0: degrees of freedom parameter, ν_0

ANORM: The log normalizing constant for $W(\Sigma_0, \nu_0)$

P: Upper triangular $m \times m$ matrix such that $\mathbf{PP}' = \Sigma_0^{-1}$

A: An $m \times m$ random matrix drawn from $W(\Sigma_0, \nu_0)$

USAGE

call WISH0(NFILE,M,TITLE,SIG0,LDSIG,ANU0,ANORM,P,LDP,
A,LDA)

Last Updated: 29th January, 1997

WTSIM(NFILE,ITER,WPP,G)

This double precision subroutine writes an iteration record on the previously opened simulation output file assigned to unit NFILE.

INPUTS

NFILE : Unit number assigned to simulation output file

ITER : Iteration number written in first line

WPP : A 3×1 vector of log weight, log prior and log data p.d.f. written in first line

G : An NPARSA×1 vector of simulated values written in succeeding lines

INPUT VIA COMMON /ATWSIM/

NPARSA : Number of simulated values to write

USAGE

call WTSIM(NFILE,ITER,WPP,G)

Last Updated: 4th June, 1997

WTSIM0(NFILE,NITER,NPARS)

This subroutine writes the first record of a simulation output file assigned to unit NFILE.

INPUTS

NFILE : Unit number assigned to simulation output file

NITER : Number of iterations written in first line

NPARS : Number of parameters

USAGE

call WTSIM0(NFILE,NITER,NPARS)

Last Updated: 4th June, 1997

Section 2: Posterior Simulator Routines

GAU2(K,H0,LDH0,H1,LDH1,HB0,HB1,THETA1,THETA2)

This double precision subroutine generates two, antithetic, drawings from the posterior distribution, given a Gaussian prior and a Gaussian likelihood.

INPUTS

K:- Order of distribution

H0:- Prior precision matrix

LDH0:- Leading dimension of H0 as defined in the calling program

H1:- Data precision matrix

LDH1:- Leading dimension of H1 as defined in the calling program

HB0:- Product of prior precision matrix and prior mean

HB1:- Product of data precision matrix and data mean

OUTPUTS

THETA1: $k \times 1$ vector drawn from posterior

THETA2: $k \times 1$ vector; antithetic pair of THETA1

USAGE

call GAU2(K,H0,LDH0,H1,LDH1,HB0,HB1,THETA1,THETA2)

Last Updated: 4th June ,1997

HNRM1(NT,X,Y,H,HPRI,HBPRI,BETA)

This double precision subroutine simulates one draw from the coefficient posterior distribution arising from a normal regression model with a single regressor, known heteroscedasticity, and a normal prior for the coefficient.

INPUTS

NT: Number of observations

X: $NT \times 1$ regressor vector

Y: $NT \times 1$ dependent vector

H: $NT \times 1$ vector of disturbance precisions

HPRI: Prior precision of coefficient

HBPRI: Product of prior precision and prior mean

OUTPUT

BETA: Draw from posterior distribution

USAGE

call HNRM1(NT,X,Y,H,HPRI,HBPRI,BETA)

Last Updated: 29th January, 1997

HNRMK(NK,NT,X,LDX,Y,H,HPRI,LDHPRI,HBPRI,BETA)

This double precision subroutine simulates one draw from the coefficient posterior distribution arising from a normal regression model with known heteroscedasticity and a normal prior for the coefficients.

INPUTS

NK: Number of regressors

NT: Number of observations

X: $NT \times NK$ matrix of regressors

LDX: Leading dimension of **X**

Y: $NT \times 1$ dependent vector

H: $NT \times 1$ vector of disturbance precisions

HPRI: $NK \times NK$ prior precision matrix

LDHPRI: Leading dimension of HPRI as defined in the calling routine

HBPRI: $NK \times 1$ vector of product between HBPRI and mean vector of coefficient prior

OUTPUT

BETA: $NK \times 1$ vector containing draw from posterior distribution

USAGE

call HNRMK(NK,NT,X,LDX,Y,H,HPRI,LDHPRI,HBPRI,BETA)

Last Updated: 29th January, 1997

NMXD(N,NMIX,LDMIX,IDCODE,MXCODE,X,H0,HA0,SONU0,ANU0,R,ALPHA,H,P,LT)

This double precision subroutine draws from the posterior distribution for a discrete mixture of normals density.

INPUTS

N: Sample size

NMIX: Number of normals in mixture

LDMIX: Leading dimension of **H0** as defined in the calling program

IDCODE: =1: Mean vector **ALPHA** in non-descending order

=2: Precision vector **H** in non-descending order

=3: Probability vector **P** in non-descending order

Otherwise, no identifying restrictions beyond prior distributions applied

MXCODE: =1: Scale mixture of normals (means fixed at 0)

=2: Mean mixture of normals (precisions all the same)

Otherwise both means and precisions are free

X: $N \times 1$ Data vector

H0: $NMIX \times NMIX$ prior precision matrix for **ALPHA** (ignored if **MXCODE**=1)

HA0: Product of prior precision and prior mean for **ALPHA** (ignored if **MXCODE** =1)

SONU0: Vector of constant parameters of χ^2 priors for **H** (only first element if **MXCODE** =2)

ANU0: Vector of degrees-of-freedom of χ^2 priors for **H** (only first element if **MXCODE** =2)

R: Vector of parameters of multivariate Beta prior for **P**

INPUT/OUTPUT

ALPHA: $NMIX \times 1$ vector of means. If **MXCODE** $\neq 1$, **NMXD** exchanges values in Gibbs sampler. If **MXCODE** = 1, all values of **ALPHA** are set the same on input and are left unchanged by **NMXD**.

H: $NMIX \times 1$ vector of precisions. If **MXCODE** $\neq 2$, **NMXD** exchanges values in Gibbs sampler. If **MXCODE** = 2, **H**(1) is the common precision and only **H**(1) is exchanged.

P: $NMIX \times 1$ vector of probabilities. **NMXD** exchanges values in Gibbs sampler

LT: $N \times 1$ vector of integers between 1 and $NMIX$ inclusive indicating mixture status for each observation. (need not be an input, as **LT** is drawn first)

USAGE

call NMXD(N,NMIX,LDMIX,IDCODE,MXCODE,X,H0,HA0,SONU0,
ANU0,R,ALPHA,H,P,LT)

Last Updated: 4th June , 1997

**NMXDP(N,NMIX,LDMIX,IDCODE,MXCODE,X,H0,HA0,S0NU0,ANU0,R,JFIX,
ALPHA,H,P,LT)**

This double precision subroutine draws from the posterior distribution for a discrete mixture of normals density, with one precision fixed at a specified value.

INPUTS

N: Sample size

NMIX: Number of normals in mixture

LDMIX: Leading dimension of **H0** as defined in the calling program

IDCODE: =1: Mean vector **ALPHA** in non-descending order

=2: Precision vector **H** in non-descending order

=3: Probability vector **P** in non-descending order

Otherwise, no identifying restrictions beyond prior distributions applied

MXCODE: =1: Scale mixture of normals (means fixed at 0)

=2: Mean mixture of normals (precision's all the same)

Otherwise both means and precision's are free

X: N×1 Data vector

H0: NMIX×NMIX prior precision matrix for **ALPHA** (ignored if

MXCODE=1)

HA0: Product of prior precision and prior mean for **ALPHA** (ignored if

MXCODE =1)

SONU0: Vector of constant parameters of χ^2 priors for **H** (only first element if **MXCODE** =2)

ANU0: Vector of degrees-of-freedom of χ^2 priors for **H** (only first element if **MXCODE** =2)

R: Vector of parameters of multivariate Beta prior for **P**

JFIX: Precision whose value is fixed

INPUT/OUTPUT

ALPHA: $NMIX \times 1$ vector of means. If **MXCODE** $\neq 1$, **NMXDP** exchanges values in Gibbs sampler. If **MXCODE** = 1, all values of **ALPHA** are set the same on input and are left unchanged by **NMXDP**.

H: $NMIX \times 1$ vector of precision's. If **MXCODE** $\neq 2$, **NMXD** exchanges values in Gibbs sampler. If **MXCODE** = 2, **H**(1) is the common precision and only **H**(1) is exchanged.

P: $NMIX \times 1$ vector of probabilities. **NMXD** exchanges values in Gibbs sampler

LT: $N \times 1$ vector of integers between 1 and **NMIX** inclusive indicating mixture status for each observation. (need not be an input, as **LT** is drawn first)

USAGE

call **NMXDP**(**N**,**NMIX**,**LDMIX**,**IDCODE**,**MXCODE**,**X**,**H0**,**HA0**,**SONU0**,
ANU0,**R**, **JFIX**,**ALPHA**,**H**,**P**,**LT**)

Last Updated: 4th June, 1997

NRMK(**NK**,**NT**,**X**,**LDX**,**Y**,**H**,**HPRI**,**LDHPRI**,**HBPRI**,**BETA**)

This double precision subroutine simulates one draw from the coefficient posterior distribution arising from a normal regression model with a normal prior for the coefficients.

INPUTS

NK: Number of regressors

NT: Number of observations

X: NT×NK matrix of regressors

LDX: Leading dimension of **X**

Y: NT×1 dependent vector

H: NT×1 vector of disturbance precisions

HPRI: NK×NK prior precision matrix

LDHPRI: Leading dimension of HPRI as defined in the calling routine

HBPRI: NK×1 vector of product between HBPRI and mean vector of coefficient prior

OUTPUTS

BETA: NK×1 vector containing draw from posterior distribution

PDFLG: Log posterior p.d.f. at the drawn value

USAGE

call NRMK(NK,NT,X,LDX,Y,H,HPRI,LDHPRI,HBPRI,BETA)

Last Updated: 29th January,1997

Chapter 5: Matrix/Vector Operations

This chapter contains routines that are used to manipulate matrices and vectors. These routines are used in many of the routines found in the previous chapters.

Section 1: Matrix Operations

CPROD(N1,N2,LIST1,LIST2,ZZ,LDZ,XY,LDXY)

This double precision subroutine extracts the sums of cross-products of a subset of the variables in a sums-of-squares-products matrix. It is useful for reordering variables.

INPUTS

N1: Number of rows in output sums-of-cross-products matrix

N2 : Number of columns in output sums-of-cross-products matrix

LIST1 : $N1 \times 1$ vector of **ZZ** entries corresponding to rows of **XY**

LIST2 : $N2 \times 1$ vector of **ZZ** entries corresponding to columns of **XY**

ZZ : Sums-of-squares-and -cross-products matrix

LDZ: Leading dimension of **ZZ** as defined in the calling routine

LDXY: Leading dimension of **XY** as defined in the calling routine

OUTPUT

XY : $N1 \times N2$ sums-of-cross-products matrix

USAGE

call CPROD(N1,N2,LIST1,LIST2,ZZ,LDZ,XY,LDXY)

Last Updated: 29th January, 1997

DAT0(NFILE,NT1,NT2,NVARS,ZZ,LDZ)

This double precision subroutine reads a data file, writes some summary statistics, and returns the sums-of-squares-and-cross-products matrix.

INPUTS

NFILE : File unit number for read command as defined in calling program

NT1 : First record (observation) of file to read

NT2 : Last record (observation) of file to read

LDZ : Leading dimension of **ZZ** as defined in the calling program

OUTPUT

NVARS : Number of variables available on output data file

ZZ : $NK \times NK$ sums-of-squares-and-cross-products matrix

USAGE

call DAT0(NFILE,NT1,NT2,NVARS,ZZ,LDZ)

Last Updated: 4th June, 1997

DAT2(NFILE,NT1,NT2,NLAG,NVARS,ZZ,LDZ)

This double precision subroutine reads a data file, writes some summary statistics to output, and returns the sums-of-squares-and-cross-products matrix for variables and their lags.

INPUTS

NFILE: File unit number of file to be read

NT1: First record (observation) to include

NT2: Last record (observation) to include

NLAG: Number of lags of each variable

LDZ: Leading dimension of **ZZ**

OUTPUTS

NVARS: Number of variables available on data file

ZZ: Sums-of-squares-and-cross-products matrix

USAGE

call DAT2(NFILE,NT1,NT2,NLAG,NVARS,ZZ,LDZ)

Last Updated: 4th June, 1997

DFULL(N,A,LDA)

This double precision subroutine converts a symmetric matrix in IMSL symmetric storage mode (upper triangular) to one in full storage mode.

INPUTS

N : Order of symmetric matrix

A : An $N \times N$ symmetric matrix in symmetric storage mode

LDA : Leading dimension of **A** as defined in the calling program

OUTPUTS

A : The $N \times N$ symmetric matrix in full storage mode

USAGE

call DFULL(N,A,LDA)

Last Updated: 29th January, 1997

DMXXTU(N,A,LDA,B,LDB)

This double precision subroutine computes the matrix product $\mathbf{B} = \mathbf{A}\mathbf{A}'$, where \mathbf{A} is upper triangular. Note that only the upper triangle of \mathbf{B} is returned (IMSL storage mode)

INPUTS

N : Order of upper triangular matrix

A : An $N \times N$ upper triangular matrix

LDA : Leading dimension of \mathbf{A} as defined in the calling program

LDB : Leading dimension of \mathbf{B} as defined in the calling program

OUTPUTS

B : The upper triangular part of $\mathbf{B} = \mathbf{A}\mathbf{A}'$

USAGE

call DMXXTU(N,A,LDA,B,LDB)

Last Updated: 29th January, 1997

DPDCK(A,LDA,N,INFO)

This double precision subroutine uses the LAPACK routine DPOTRF to check whether a matrix is positive definite.

INPUTS

A : Matrix to be checked

LDA: Leading dimension as defined in calling program

N: Order of matrix **A**

OUTPUT

INFO : An integer that is equal to zero if the matrix is positive definite and non-zero if not.

USAGE

call DPDCK(A,LDA,N,INFO)

Last Updated: 29th January, 1997

DTLGPD(N,A,LDA)

This double precision real function returns the log determinant of a positive definite matrix.

INPUTS

N : Order of matrix

A : An $N \times N$ positive definite matrix

LDA : leading dimension of **A** as defined in the calling program

OUTPUT

DTLGPD : $\log|A|$

USAGE

user_var=DTLGPD(N,A,LDA)

Last Updated: 29th January, 1997

PDSYMR(NFILE,N,A,LDA)

This double precision subroutine reads an $N \times N$ matrix from a file, whose unit number is NFILE, and checks to see that it is symmetric and positive definite.

INPUTS

NFILE: The unit number of the file that is read as defined in the calling program

N : Order of the matrix **A**

LDA : The leading dimension of **A** as defined in the calling program

OUTPUT

A : The $N \times N$ matrix that was read in from file whose unit number is NFILE

USAGE

call PDSYMR(NFILE,N,A,LDA)

Last Updated: 29th January, 1997

PROVAR(N,P,LDP,SIGMA,LDSIG)

This double precision subroutine constructs the variance matrix from a multivariate projection matrix. This routine is the inverse of the routine VARPRO.

INPUTS

N : Order of multivariate distribution

P : An $N \times N$ projection matrix

LDP : Leading dimension of **P** as defined in the calling program

LDSIG : Leading dimension of **SIG** as defined in the calling program

OUTPUT

SIGMA : Variance matrix as constructed from **P**

USAGE

call PROVAR(N,P,LDP,SIGMA,LDSIG)

Last Updated: 29th January, 1997

VARPRO(N,SIGMA,LDSIG,P,LDP)

This double precision subroutine constructs a recursive projection matrix **P**, given a variance matrix, Σ . The linear projection of x_i on x_1, \dots, x_{i-1} is

$$x_i = p_{i1}x_1 + \dots + p_{i,i-1}x_{i-1} + u_i$$

where the standard deviation of $u_i = p_{ii}$.

INPUTS

N : Order of multivariate distribution

SIGMA : $N \times N$ variance matrix Σ

LDSIG: Leading dimension of Σ as defined in the calling program

LDP : Leading dimension of **P** as defined in the calling program

OUTPUT

P : The $N \times N$ projection matrix

USAGE

call VARPRO(N,SIGMA,LDSIG,P,LDP)

Last Updated: 29th January, 1997

PRCOR(HTITLE,N,A,LDA)

This double precision subroutine prints the normalized lower triangle of a symmetric $N \times N$ matrix . e.g. it converts the variance matrix to a correlation matrix before printing.

INPUTS

HTITLE: Header to identify matrix

N: Order of matrix

A: Matrix to be printed

LDA: Leading dimension of **A** as defined in the calling program

USAGE

call PRCOR(HTITLE,N,A,LDA)

Last Updated: 4th June ,1997

PRSYM(HTITLE,N,A,LDA)

This double precision subroutine prints the lower triangle of a symmetric $N \times N$ matrix.

INPUTS

HTITLE : Character string printed at top of matrix

N : Order of matrix

A : An $N \times N$ symmetric matrix

LDA : Leading dimension of **A** as defined in the calling program

USAGE

call PRSYM(HTITLE,N,A,LDA)

Last Updated: 29th January, 1997

TRAB(NRA,NCA,A,LDA,NRB,NCB,B,LDB)

This double precision real function compute the trace of a matrix product **AB**.

INPUTS

NRA : Number of rows in **A**

NCA : Number of columns in **A**

A : An NRA×NCA matrix

LDA : The leading dimension of **A** as defined in the calling routine

NRB : Number of rows in **B**

NCB : Number of columns in **B**

B : An NRB×NCB matrix

LDB : The leading dimension of **B** as defined in the calling routine

OUTPUT

TRAB : $\text{tr}(\mathbf{AB})$

USAGE

user_var= TRAB(NRA,NCA,A,LDA,NRB,NCB,B,LDB)

Last Updated: 29th January, 1997

UCRGRG(M,N,A,LDA,B,LDB)

This double precision subroutine copies an $M \times N$ matrix **A** to an $M \times N$ matrix **B**.

INPUTS

M : Number of rows in **A** and **B**

N : Number of columns in **A** and **B**

A : The $M \times N$ matrix to be copied

LDA : The leading dimension of **A** as defined in the calling routine

LDB : The leading dimension of **B** as defined in the calling routine

OUTPUTS

B : The copy of **A**

USAGE

call UCRGRG(M,N,A,LDA,B,LDB)

Last Updated: 29th January, 1997

UCROSS(NRA,NCA,A,LDA,NRB,NCB,B,NRC,NCC,C,LDC)

This double precision subroutine calculates the Kronecker product of matrices **A** and **B** and places it in **C**. That is, $C = A \otimes B$

INPUTS

NRA : Number of rows in **A**

NCA : Number of columns in **A**

A : An $NRA \times NCA$ matrix

LDA : Leading dimension of **A** as defined in the calling program

NRB : Number of rows in **B**

NCB : Number of columns in **B**

B : An $NRB \times NCB$ matrix

LDB : Leading dimension of **B** as defined in the calling program

NRC : Number of rows in **C**

NCC : Number of columns in **C**

LDC : Leading dimension of **C** as defined in the calling program

OUTPUT

C : The Kronecker product $\mathbf{A} \otimes \mathbf{B}$

USAGE

call UCROSS(NRA,NCA,A,LDA,NRB,NCB,B,LDB,NRC,NCC, C,LDC)

Last Updated: 29th January, 1997

ULFTDS(N,A,LDA,R,LDR)

This double precision subroutine computes the upper triangle Cholesky factor of a real symmetric positive definite matrix.

INPUTS

N : Order of the matrix

A : An $N \times N$ positive definite symmetric real matrix

LDA : Leading dimension of **A** as defined in the calling program

LDR : Leading dimension of **R** as defined in the calling program

OUTPUT

R : The $N \times N$ upper triangular Cholesky factorization of the matrix **A**

USAGE

call ULFTDS(N,A,LDA,R,LDR)

Last Updated: 29th January, 1997

UMADD(M,N,A,LDA,B,LDB,C,LDC)

This double precision subroutine adds two matrices element by element to form the matrix $C = A + B$.

INPUTS

M : Number of rows in the matrices **A**, **B**, and **C**

N : Number of columns in the matrices **A**, **B**, and **C**

A : A $M \times N$ matrix

LDA : Leading dimension of **A** as defined in the calling program

B : A $M \times N$ matrix

LDB: Leading dimension of **B** as defined in the calling program

LDC: Leading dimension of **C** as defined in the calling program

OUTPUT

C : The $M \times N$ matrix sum of **A** and **B**.

USAGE

call UMADD(M,N,A,LDA,B,LDB,C,LDC)

Last Updated: 29th January, 1997

UMSUB(M,N,A,LDA,B,LDB,C,LDC)

This double precision subroutine forms the matrix $\mathbf{C} = \mathbf{A} - \mathbf{B}$.

INPUTS

M : Number of rows in the matrices **A**, **B**, and **C**

N : Number of columns in the matrices **A**, **B**, and **C**

A : A $M \times N$ matrix

LDA : Leading dimension of **A** as defined in the calling program

B : A $M \times N$ matrix

LDB: Leading dimension of **B** as defined in the calling program

LDC: Leading dimension of **C** as defined in the calling program

OUTPUT

C : The $M \times N$ matrix $\mathbf{A} - \mathbf{B}$

USAGE

call **UMSUB(M,N,A,LDA,B,LDB,C,LDC)**

Last Updated: 29th January, 1997

UMSADD(M,N,A,LDA,B,LDB,C,LDC,S)

This double precision subroutine forms the matrix $\mathbf{C} = s\mathbf{A} + \mathbf{B}$.

INPUTS

M : Number of rows in the matrices **A**, **B**, and **C**

N : Number of columns in the matrices **A**, **B**, and **C**

A : A $M \times N$ matrix

LDA : Leading dimension of **A** as defined in the calling program

B : A $M \times N$ matrix

LDB: Leading dimension of **B** as defined in the calling program

LDC: Leading dimension of **C** as defined in the calling program

S : Scalar used to scale the matrix **A**

OUTPUT

C : The $M \times N$ matrix $C = sA + B$

USAGE

call UMSADD(M,N,A,LDA,B,LDB,C,LDC,S)

Last Updated: 29th January, 1997

UMISET(N,A,LDA)

This double precision subroutine returns an $N \times N$ identity matrix in **A** .

INPUTS

N : Order of the matrix

LDA : Leading dimension of **A** as defined in the calling program

OUTPUT

A : The identity matrix I_N .

USAGE

call UMISET(N,A,LDA)

Last Updated: 29th January, 1997

UM0SET(M,N,A,LDA)

This double precision subroutine returns an $M \times N$ zero matrix.

INPUTS

N : Row dimension of the matrix

M : Column dimension of the matrix

LDA : Leading dimension of **A** as defined in the calling program

OUTPUT

A : The $M \times N$ zero matrix.

USAGE

call UM0SET(N,M,A,LDA)

Last Updated: 29th January, 1997

UNVEC(M,N,B,A,LDA)

This double precision subroutine unvectorizes a vector into an $M \times N$ matrix **A**. The vector **B** is assumed to contain the columns of **A** stored successively. This routine is the inverse of **VEC**.

INPUTS

M : The row dimension of **A**

N : The column dimension of **A**

B: The vectorized matrix. **B** is a vector of length MN

LDA : The leading dimension of **A** as defined in the calling program

OUTPUT

A : The M×N matrix that was vectorized in **B**.

USAGE

call UNVEC(M,N,B,A,LDA)

Last Updated: 29th January, 1997

UQUAF(N,X,A,LDA)

This double precision real function forms the quadratic from $Q = \mathbf{x}'\mathbf{A}\mathbf{x}$ where **A** is symmetric.

INPUTS

N : Dimension of **x** and order of **A**

X : **x** vector

A : **A** matrix

LDA : Leading dimension of **A** as defined in the calling program

OUTPUT

UQUAF : The quadratic form $\mathbf{x}'\mathbf{A}\mathbf{x}$

USAGE

user_var=UQUAF(N,X,A,LDA)

Last Updated: 29th January, 1997

USCAL2(NA,NB,X,A,LDA)

This double precision subroutine scales a general matrix **A** by x .

INPUTS

NA : Number of rows in **A**

NB : Number of columns in **A**

X : Scalar used to scale the matrix **A**

A : An $NA \times NB$ real matrix

LDA : Leading dimension of **A** as defined in the calling program

OUTPUT

A : The scaled $NA \times NB$ matrix $x\mathbf{A}$

USAGE

call USCAL2(NA,NB,X,A,LDA)

Last Updated: 29th January, 1997

UXTAXF(N,M,X,LDX,A,LDA,B,LDB)

This double precision subroutine forms the matrix product $\mathbf{B} = \mathbf{X}'\mathbf{A}\mathbf{X}$ where **A** is symmetric.

INPUTS

N : Row dimension of **X** and order of **A**

M : Column dimension of **X** and order of **B**

X : **X** matrix

LDX : Leading dimension of **X** as defined in the calling program

A : **A** matrix (full storage)

LDA : Leading dimension of **A** as defined in the calling program

LDB : Leading dimension of **B** as defined in the calling program

OUTPUT

B : The matrix product $\mathbf{X}'\mathbf{A}\mathbf{X}$

USAGE

call UXTAXF(N,M,X,LDX,A,LDA,B,LDB)

Last Updated: 29th January, 1997

UXAXTF(N,M,X,LDX,A,LDA,B,LDB)

This double precision subroutine forms the matrix product $\mathbf{B} = \mathbf{X}\mathbf{A}\mathbf{X}'$ where **A** is symmetric.

INPUTS

N : Row dimension of **X** and order of **A**

M : Column dimension of **X** and order of **B**

X : **X** matrix

LDX : Leading dimension of **X** as defined in the calling program

A : **A** matrix (full storage)

LDA : Leading dimension of **A** as defined in the calling program

LDB : Leading dimension of **B** as defined in the calling program

OUTPUT

B : The matrix product $\mathbf{X}\mathbf{A}\mathbf{X}'$

USAGE

call UXAXTF(N,M,X,LDX,A,LDA,B,LDB)

Last Updated: 29th January, 1997

Section 2: Vector Operations

UVADD(N,A,B,C)

This double precision subroutine adds two vectors, $C = A + B$.

INPUTS

N : Order of the vectors **A**, **B** and **C**

A : An $N \times 1$ vector

B : An $N \times 1$ vector

OUTPUT

C : The $N \times 1$ vector $A + B$

USAGE

call UVADD(N,A,B,C)

Last Updated: 29th January, 1997

UVSUB(N,A,B,C)

This double precision subroutine adds two vectors, $C = A - B$.

INPUTS

N : Order of the vectors **A**, **B** and **C**

A : An $N \times 1$ vector

B : An $N \times 1$ vector

OUTPUT

C : The $N \times 1$ vector **A** – **B**

USAGE

call UVSUB(N,A,B,C)

Last Updated: 29th January, 1997

UVSADD(N,A,S,B,C)

This double precision subroutine adds two vectors, **C** = $s\mathbf{A} + \mathbf{B}$.

INPUTS

N : Order of the vectors **A**, **B** and **C**

A : An $N \times 1$ vector

S : The scalar s

B : An $N \times 1$ vector

OUTPUT

C : The $N \times 1$ vector $s\mathbf{A} + \mathbf{B}$

USAGE

call UVSADD(N,A,B,C)

Last Updated: 29th January, 1997

VEC(M,N,A,LDA,B)

This double precision subroutine vectorizes a matrix, $\mathbf{B} = \text{vec}(\mathbf{A})$. That is \mathbf{B} is a vector containing the columns of \mathbf{A} stored successively.

INPUT

M : Row dimension of \mathbf{A}

N : Column dimension of \mathbf{A}

A : An $M \times N$ matrix

LDA : Leading dimension of \mathbf{A} as defined in the calling program

OUTPUT

B : An $MN \times 1$ vector containing the columns of \mathbf{A}

USAGE

call VEC(M,N,A,LDA,B)

Last Updated: 29th January, 1997

VECLT(N,A,LDA,B)

This double precision subroutine vectorizes the lower triangle of a square matrix of order N . The first N elements of \mathbf{B} contain the first column of \mathbf{A} , the next $N-1$ elements of \mathbf{B} contain the last $N-1$ elements of the second column of \mathbf{A} , ..., the next $N-j$ elements of \mathbf{B} contain the last $N-j$ elements of the $(j+1)$ th column of

A,, and the last element of **B** contains the element in the last row and column of **A**.

INPUTS

N : Order of the matrix **A**

A : $N \times N$ matrix whose lower triangle is to be vectorized

LDA: Leading dimension of **A** as defined in the calling program

OUTPUT

B : Vectorized lower triangle of **A**

USAGE

call VECLT(N,A,LDA,B)

Last Updated: 29th January, 1997

Chapter 6: Parameter Transformations

This chapter contains routines for transforming parameters of a model.

TRMARK(LFWD,N,VP,VPTR,TRJAC)

This double precision subroutine performs a normalizing transformation for a Markov transition matrix. In the forward transformation the transformation is from p_{ij} to $q_{ij} = \log\left(\frac{p_{ij}}{p_{ii}}\right) \quad \forall \quad j \neq i$. In the backward transformation the inverse transformation is returned. The routine also returns the log Jacobian of the transformation. This is what you need to add to the log density in the new parameterization.

INPUTS

LFWD: Logical input. If `.TRUE.` the routine performs the forward transformation. If `.FALSE.` the routine performs the backwards transformation.

N: Order of the Markov matrix

INPUT/OUTPUT

VP: Markov matrix, stored in vectorized form (by columns). Input in the forward transformation, output in the backwards transformation

VPTR: Transformed Markov matrix, stored in vectorized form (by rows) . Output in the forward transformation, input in the backwards transformation.

OUTPUT

TRJAC: Log Jacobian of transformation.

USAGE

call TRMARK(LFWD,N,VP,VPTR,TRJAC)

Last Updated: 4th June, 1997

TRPD(LFWD,N,VA,VATR,TRJAC)

This double precision subroutine performs a normalizing transformation on a symmetric positive definite matrix, and returns the log Jacobian of transformation. The transformation is based on the Choleski decomposition of the positive definite matrix \mathbf{A} , $\mathbf{A} = \mathbf{L}\mathbf{L}'$, where \mathbf{L} is lower triangular. The transformation is to the logs of the diagonal elements of \mathbf{L} , and the untransformed elements in the strict lower triangle of \mathbf{A} . The log Jacobian is what you add to the log density of the original distribution, to get the correct log density for the transformed variables.

INPUTS

LFWD: Logical input. If .TRUE. then the forward transformation is carried out. If .FALSE. the backward transformation is carried out.

N: Order of the positive definite matrix

INPUTS/OUTPUTS

VA: positive definite matrix in lower triangular storage, by columns

VATR: Transformed Choleski decomposition in lower triangular storage, by columns

OUTPUT

TRJAC: log Jacobian of the transformation

USAGE

call TRPD(LFWD,N,VA,VATR,TRJAC)

Last Updated: 4th June, 1997

Chapter 7: Auxiliary Routines

This chapter contains routines used in JGLIB that do not fall in the above chapters.

DAT1(NFILE,NT1,NT2,NVARS,D,LDD,NCD)

This double precision subroutine reads a data file, writes some summary statistics to output, and returns records t1 through t2 in the corresponding rows of **D**

INPUTS

NFILE : The unit number of the file to be read as defined in the calling program

NT1 : The first record to include, t1

NT2 : The last record to include, t2

LDD : Leading dimension of **D** as defined in the calling program

NCD : Number of columns allocated to **D** in calling routine

OUTPUTS

NVARS : Number of variables available on data file

D : (t2-t1)×NCD matrix containing records t1 through t2 of data file

USAGE

call DAT1(NFILE,NT1,NT2,NVARS,D,LDD,NCD)

Last Updated: 4th June, 1997

DATFMC(NFILE,NOPT,NM,AN,LDM)

This double precision subroutine reads a first-order Markov chain model data file, and returns the data matrix.

INPUTS

NFILE : Unit number of data file to be read as defined in the calling program

NOPT : If $\text{NOPT} > 0$ print matrix, otherwise do not.

NM : Number of categories

LDM : Leading dimension of **A** as defined in the calling program

OUTPUT

AN : The $\text{NM} \times \text{NM}$ data matrix of the first-order Markov chain model

USAGE

call DATFMC(NFILE,NOPT,NM,AN,LDM)

Last Updated: 4th June, 1997

FOPEN(HTITLE,N,LNEW)

This subroutine reads a file name from input unit 5 and opens the file.

INPUTS

HTITLE:- Information to print about opened file

N : File number to open

LNEW : If =.true. then file has status = 'new'; if =.false. then file has status = 'old'

USAGE

call FOPEN(N,LNEW)

Last Updated: 4th June, 1997

INIT

This double precision subroutine initializes the common /CONST/ and sets the seed of the random number generator.

USAGE

call INIT

Last Updated: 4th June, 1997

LFLIP(PLOG)

This logical function returns the value .TRUE. with probability whose log is PLOG if PLOG < 0 . Otherwise it returns the value .FALSE. .

INPUT

PLOG : Log(p) where p is the probability of returning a .TRUE. value

OUTPUT

LFLIP : .TRUE. with probability p and .FALSE. with probability 1-p

USAGE

user_var = LFLIP(PLOG)

Last Updated: 29th January, 1997

LISTR(NFILE,N,MAX,LDUPCK,LIST)

This subroutine reads a list of integer variables from a file. It checks for variables being in range and optionally for repetitions.

INPUTS

NFILE : Unit number of file to be read as defined in the calling program

N : Length of list

MAX : Largest integer permitted

LDUPCK : If .TRUE. the routine checks for duplication of entries

OUTPUT

LIST : List of integers read

USAGE

call LISTR(NFILE,N,MAX,LDUPCK,LIST)

Last Updated: 29th January, 1997

LORDER(N,A)

This logical function checks a string of numbers for nondescending order.

INPUTS

N : Number of elements to check

A : N×1 array of elements

OUTPUT

LORDER : .TRUE. if $A_i \leq A_{i+1}$, $i=1,\dots,n-1$

.FALSE. if $A_i > A_{i+1}$ for some I

USAGE

user_var=LORDER(N,A)

Last Updated: 29th January, 1997

NDISC(N,P)

This integer function randomly selects one from N categories, where the probabilities are proportional to the vector P.

INPUTS

N : Number of categories to choose from

P : N×1 vector of probabilities

OUTPUT

NDISC : integer $\in \{1, \dots, N\}$

USAGE

user_var = NDISC(N,P)

Last Updated: 29th January, 1997

NOWPR(NFILE)

This subroutine prints the current day of the week, date, and time to the file whose unit is NFILE. Note that this file must have been opened before calling this routine.

INPUT

NFILE : The unit number of the file that the information is to be written to.

USAGE

call NOWPR(NFILE)

Last Updated: 29th January, 1997

PEND(A1)

This subroutine prints a message and then calls **TERM** to halt execution of the program.

INPUT

A1 : Character string to be printed.

USAGE

call PEND(A1)

Last Updated: 29th January, 1997

PINTI(A1,K1,K2,K3)

This subroutine checks to see if the integer **K1** is in the range (**K2**,**K3**). If it is, then the routine returns normally to the program. If not, the routine prints an error message to the standard output and calls **TERM** to halt execution of program.

INPUTS

A1 : Character string to be printed

K1 : Integer to be tested

K2 : Integer. Lower bound of acceptable range

K3 : Integer: Upper bound of acceptable range

USAGE

call PINTI(A1,K1,K2,K3)

Last Updated: 29th January, 1997

PINTR(A1,R1,R2,R3)

This double precision subroutine checks to see if the real number R1 is in the range (R2,R3). If it is, then the routine returns normally to the calling program. If not, the routine prints an error message to the standard output and calls TERM to halt execution of program.

INPUTS

A1 : Character string to be printed

R1 : Integer to be tested

R2 : Integer. Lower bound of acceptable range

R3 : Integer: Upper bound of acceptable range

USAGE

call PINTI(A1,R1,R2,R3)

Last Updated: 29th January, 1997

PLINE(NSKIP,A1)

This subroutine prints information to the standard output.

INPUTS

NSKIP : Number of lines to skip when printing

A1 : Character string to print

OUTPUT

This routine skips **NSKIP** number of lines and then prints **A1**.

USAGE

call PLINE2(NSKIP,A1)

Last Updated: 29th January, 1997

PLINE1(NSKIP,A1,I1,A2)

This subroutine prints information to the standard output.

INPUTS

NSKIP : Number of lines to skip when printing

A1 : Character string to print

N1 : Integer to print

A2 : Character string to print

OUTPUT

This routine skips **NSKIP** number of lines and then prints **A1**, **N1**, **A2** in that order.

USAGE

call PLINE1(NSKIP,A1,N1,A2)

Last Updated: 29th January, 1997

PLINE2(NSKIP,A1,I1,A2,I2,A3)

This subroutine prints information to the standard output.

INPUTS

NSKIP : Number of lines to skip when printing

A1 : Character string to print

N1 : Integer to print

A2 : Character string to print

N2 : Integer to print

A3 : Character string to print

OUTPUT

This routine skips NSKIP number of lines and then prints A1, N1, A2, N2, A3 in that order.

USAGE

call PLINE2(NSKIP,A1,N1,A2,N2,A3)

Last Updated: 29th January, 1997

PLISTR(NFILE,N,A)

This double precision subroutine reads a list of positive numbers from a file and checks to see whether they are actually positive.

INPUTS

NFILE: Unit number of file that contains the positive numbers that are to be read in

N: Length of list

OUTPUT

A: List of real constants read

USAGE

call PLISTR(NFILE,N,A)

Last Updated: 4th June , 1997

PSIMPA1,N,P)

This double precision subroutine checks whether a vector is in the unit simplex. If it isn't the routine prints an error message and terminates the program.

INPUTS

A1: Character string to be printed

N: Order of vector to be checked

P: $N \times 1$ vector to be checked

USAGE

call PSIMP(A1,N,P)

Last Updated: 4th June, 1997

RTMIN(N,A)

This double precision real valued function returns the smallest modulus of the roots of a polynomial of order N.

INPUTS

N: Order of polynomial

A: N+1×1 vector of coefficients in ascending order. i.e. beginning with order 0

OUTPUT

RTMIN: smallest modulus

USAGE

user_var = RTMIN(N,A)

Last Updated: 4th June, 1997

TERM

This routine prints a terminal message and then halts execution of the program.

USAGE

call TERM

Last Updated: 29th January, 1997

UMIN(A,B)

This double precision real function returns the minimum of two real numbers.

INPUTS

A : real number

B : real number

OUTPUT

UMIN : The minimum of A and B

USAGE

user_var = UMIN(A,B)

Last Updated: 29th January, 1997

UMAX(A,B)

This double precision real function returns the maximum of two real numbers.

INPUTS

A : real number

B : real number

OUTPUT

UMAX : The maximum of A and B

USAGE

`user_var = UMAX(A,B)`

Last Updated: 29th January, 1997

INDEX OF ROUTINES

—A—

- ACFAR**
AR(p) representation, 2
- ARACF**, 2
Autocovariance function from AR(p)
representation, 2
- ARPA CF**
Computes partial autocorrelation function from
AR(p) representation, 3
- ARPALJ**
Computes log determinant Jacobian of the
transformation from partial autocorrelation
function coefficients to AR(p) coefficients, 4
- ARSDII**
Integrates spectral density of univariate
autoregressive process and Cosine function, 7

—B—

- BERN**
generates Bernoulli random variable, 15
- BET**
Draws one multivariate Beta deviate, 16
- BETO**
Reads in parameters of Beta(N,R) and prints out
summary information, 51
- BETK**
Evaluates log density kernel of a multivariate
Beta(N,R) deviate, 17
- BETN**
Evaluates log normalizing constant for a Beta(N,R)
deviate, 17

—C—

- CHI**
Generates a Chi-squared random deviate, 18
- CHIO**
Reads in parameters of multiple Chi-square
distributions and prints out summary
information, 52
- CHIOP**
reads information from file and returns constants
and drawings from multiple chi-square
distributions, 53
- CHIK**
Evaluates the log density kernel of a Chi-squared
random deviate, 19
- CHIN**
Evaluates the log normalizing constant of a Chi-
square deviate, 19
- CHIO**
Draws N independent Chi-square deviates in non-
descending order, 20
- CHIO1**

Draws N independent chi-square distributions
subject to drawings being non-descending, 21

- CHIOP**
draws N independent chi-squares that are
monotonically nondecreasing and where one of
the chi-squares is set at a pre-set value, 22
- CPROD**
Extracts the sums-of-cross-products of a subset of
rows and columns of a matrix, 71

—D—

- DAT0**
Reads a data file and returns the sums-of-cross-
products matrix, 72
- DAT1**
Reads a data file, writes some summary statistics
and returns a subset of the data records, 96
- DAT2**
reads data dfile and produces sums-of-squares-and-
cross-products matrix for variables and their
lags, 72
- DATFMC**
Reads a first order Markov chain model data file
and returns the data matrix, 96
- DFULL**
Converts a symmetric matrix stored in IMSL
symmetric storage mode to full storage mode, 73
- DMXXTU**
Computes the matrix product XX' , 74
- DPDCK**
Checks a matrix for positive definiteness, 74
- DTLGPD**
Returns the log determinant of a positive definite
matrix, 75

—F—

- FARSDII**
Evaluates product of spectral density and Cosine
function at a given frequency, 8
- FOPEN**
Opens a file, 97

—G—

- GAU**
Draws two antithetic normal random vectors, 23
- GAU0**
Reads in parameters of a multivariate Normal
distribution and prints out summary information,
54
- GAU1MX**
Draws a normal deviate randomly from a given set
of Normal distributions, 24
- GAU2**

generates two, antithetic, drawings from the prior distribution, given a Gaussian prior and likelihood, 63

GAUA

Draws two antithetic vectors from general Normal distribution, 25

GAUB

This subroutine draws two antithetic vectors from a normal distribution, 25

GAUI

Draws a random Normal vector subject to the vector being an invertible lag operator, 26

GAUI0

Reads in parameters of a multivariate Normal distribution and prints out summary information, 55

GAUI1

Draws a multivariate Normal deviate using independence Metropolis with uniform source, 27

GAUI2

Draws from multivariate Normal using independence Metropolis and provides importance sampling weights, 27

GAUIN

Evaluates log normalizing constant given an invertibility condition, 28

GAUK

Computes the log density kernel of a multivariate Normal random deviate, 29

GAUN

Evaluates the log normalizing constant of a multivariate Normal distribution, 30

GAUT

Draws from a truncated multivariate Normal distribution without rejection, 30

GAUTK

Evaluates the log density kernel of a truncated multivariate Normal distribution, 32

GAUTN

Evaluates the log normalizing constant of a truncated Normal distribution, 33

GHK

Approximates the probability that a Normal random deviate satisfies given inequality constraints, 34

—H—

HNRMI

Simulates one draw from the coefficient posterior distribution of a normal regression model with a single regressor and known heteroscedasticity, 64

HNRMK

Simulates one draw from the posterior distribution of a normal multiple regression model with known heteroscedasticity, 65

—I—

INIT

Initializes the common block /CONST/, 98

—L—

LBERN

returns .TRUE. with probability P, 15

LFLIP

Returns the value .TRUE. with a given probability, 98

LINOP

Determines whether an autoregressive lag operator is invertible, 4

LISTR

Reads a list of integers, checks for repetitions and checks whether variables are in range, 99

LORDER

Checks a list of numbers for nondescending order, 99

—N—

NDISC

Random selection from one of a number of categories, 100

NMXD

draws from the posterior distribution for a discrete mixture of normals density, 66

NMXD0

Reads in parameters for a discrete mixture of Normals model and returns summary information, 56

NMXD0P

reads in parameters for discrete mixture of normals distribution for the case where one of the precisions is fixed at a pre-set value, 58

NMXDP

draws from the posterior for a discrete mixture of normals distribution where one precision is fixed at a pre-set value, 68

NOWPR

Prints current day of week, date and time to file, 100

NRMK

Simulates one draw from the coefficient posterior distribution from a normal multiple regression model, 69

—P—

PACFAR

Computes AR(p) from partial autocorrelation function, 5

PDSYMR

Checks whether a matrix is symmetric and positive definite, 76

PEND

Prints a message and then halts execution of program, 101

PINTI

Checks a condition and optionally halts execution of program, 101

PINTR

Real version of PINTI, 102

PLINE

Prints information to standard output, 103

PLINE1

Prints information to standard output, 103

PLINE2

Prints information to standard output, 104

PRCOR

prints normalized lower triangle of a matrix, 78

PROVAR

Constructs the variance matrix from a multivariate projection matrix, 76

PRSYM

Prints the lower triangle of a symmetric matrix, 78

PSIMP

checks vector to see if it is in the unit simplex, 105

—Q—

QUANT

Computes quantiles from the sampled distribution of a random variable, 49

—R—

RDSIM

Reads an iteration record from a file, 60

RDSIM0

Reads the first iteration record from a simulation file, 60

RTMIN

returns smallest modulus of the roots of a polynomial, 106

—S—

STT

make two antithetic drawings from a multivariate Student-t distribution, 40

STTA

make two antithetic drawings from a multivariate Student-t distribution where the precision matrix is already factored, 40

STTB

make two antithetic drawings from a multivariate Student-t distribution given the upper triangular factor of the variance matrix, 41

STTK

function that evaluates the log density kernel of a multivariate Student-t distribution, 42

STTN

evaluates log normalizing constant for a multivariate Student-t distribution, 43

—T—

TERM

Prints a terminal message and halts execution of program, 106

TRAB

Computes the trace of a matrix product AB, 79

TRMARK

performs transformation of the parameters of a Markov transition matrix and returns log Jacobian of transformation, 93

TRPD

transforms parameters of a positive definite matrix and returns log Jacobian of transformation, 94

—U—

UCRGRG

Copies a matrix, 80

UCROSS

Calculates the Kronecker product of two matrices, 80

ULFTDS

Computes the upper triangular Cholesky factor of a real positive definite matrix, 81

UM0SET

Returns the zero matrix, 85

UMADD

Adds two matrices, 82

UMAX

Returns the maximum of two real numbers, 107

UMIN

Returns the minimum of two real numbers, 107

UMISET

Returns the identity matrix, 84

UMSADD

Scales a matrix and adds it to another matrix, 83

UMSUB

Subtracts one matrix from another, 83

UNVEC

Unvectorizes a vector into a matrix, 85

UQUAF

Computes the quadratic form $x'Ax$, 86

URNMVN

Draws one random vector from a multivariate Normal distribution, 36

USCAL2

Scale a general matrix, 87

UVADD

Adds two vectors, 89

UVSADD

Scales a vector and adds it to another, 90

UVSUB

Subtracts one vector from another, 89

UXAXTF

Forms the matrix product XAX' where A is symmetric, 88

UXTAXF

Forms the matrix product $X'AX$ where A is symmetric, 87

—V—

VARPRO

Constructs a recursive projection matrix from a variance matrix, 77

VEC

Vectorize a matrix, 91

VECLT

Vectorizes the lower triangle of a square matrix, 91

—**W**—

WISH

Draws one realization from a Wishart distribution, 44

WISH0

Reads in parameters for a Wishart distribution and returns summary information, 61

WISHA

draws from Wishart distribution using already factored matrix parameter, 44

WISHB

draws from Wishart distribution using upper triangular factor, 45

WISHK

Evaluate the log density kernel of a Wishart distribution, 46

WISHN

Evaluates the log normalizing constant for a Wishart distribution, 46

WTSIM

Writes an iteration record to a file, 62

WTSIM0

Writes the first record of a simulation file to a file, 63

—**X**—

XCHIT

Draws a truncated Chi-square random deviate, 22

XNOT

Draws a truncated Normal random deviate, 37

XNOTK

Evaluates the log density kernel of a truncated Normal distribution, 38

XNOTN

Evaluates the log normalizing factor of a Normal distribution, 39

—**Y**—

YLC

Draws from a distribution whose log density is strictly concave, 47

YUM

Draws from a distribution with a unimodal density on a finite support, 48

—**Z**—

ZI1

Calculates inverse of a z-transform, 9

ZI2

Inverts a matrix z-transform, 10

ZM1

Calculates the product of two z-transforms, 10

ZM2

Calculates the product of two matrix z-transforms, 11

ZQ1

Calculates the quotient of two z-transforms, 12

ZQ2

Calculates the quotient of two matrix z-transforms, 13