NAG Toolbox

nag_rand_times_mv_varma (g05pj)

1 Purpose

nag_rand_times_mv_varma (g05pj) generates a realization of a multivariate time series from a vector autoregressive moving average (VARMA) model. The realization may be continued or a new realization generated at subsequent calls to nag_rand_times_mv_varma (g05pj).

2 Syntax

```
[r, state, x, ifail] = nag_rand_times_mv_varma(mode, n, xmean, ip, phi, iq,
theta, var, r, state, 'k', k, 'lr', lr)
[r, state, x, ifail] = g05pj(mode, n, xmean, ip, phi, iq, theta, var, r, state,
'k', k, 'lr', lr)
```

3 Description

Let the vector $X_t = (x_{1t}, x_{2t}, \ldots, x_{kt})^{\mathrm{T}}$, denote a k-dimensional time series which is assumed to follow a vector autoregressive moving average (VARMA) model of the form:

$$X_t - \mu = \phi_1(X_{t-1} - \mu) + \phi_2(X_{t-2} - \mu) + \cdots + \phi_p\big(X_{t-p} - \mu\big) + \epsilon_t - \theta_1\epsilon_{t-1} - \theta_2\epsilon_{t-2} - \cdots - \theta_q\epsilon_{t-q} \tag{1}$$

where $\epsilon_t = (\epsilon_{1t}, \epsilon_{2t}, \ldots, \epsilon_{kt})^{\mathrm{T}}$, is a vector of k residual series assumed to be Normally distributed with zero mean and covariance matrix Σ. The components of ϵ_t are assumed to be uncorrelated at non-simultaneous lags. The ϕ_i's and θ_j's are k by k matrices of parameters. $\{\phi_i\}$, for $i = 1, 2, \ldots, p$, are called the autoregressive (AR) parameter matrices, and $\{\theta_j\}$, for $j = 1, 2, \ldots, q$, the moving average (MA) parameter matrices. The parameters in the model are thus the p k by k ϕ-matrices, the q k by k θ-matrices, the mean vector μ and the residual error covariance matrix Σ. Let

$$A(\phi) = \begin{bmatrix} \phi_1 & I & 0 & . & . & . & 0 \\ \phi_2 & 0 & I & 0 & . & . & 0 \\ . & & & . & & & \\ . & & & & . & & \\ . & & & & & . & \\ \phi_{p-1} & 0 & . & . & . & 0 & I \\ \phi_p & 0 & . & . & . & 0 & 0 \end{bmatrix}_{pk \times pk} \quad \text{and} \quad B(\theta) = \begin{bmatrix} \theta_1 & I & 0 & . & . & . & 0 \\ \theta_2 & 0 & I & 0 & . & . & 0 \\ . & & & . & & & \\ . & & & & . & & \\ . & & & & & . & \\ \theta_{q-1} & 0 & . & . & . & 0 & I \\ \theta_q & 0 & . & . & . & 0 & 0 \end{bmatrix}_{qk \times qk}$$

where I denotes the k by k identity matrix.

The model (1) must be both stationary and invertible. The model is said to be stationary if the eigenvalues of $A(\phi)$ lie inside the unit circle and invertible if the eigenvalues of $B(\theta)$ lie inside the unit circle.

For $k \geq 6$ the VARMA model (1) is recast into state space form and a realization of the state vector at time zero computed. For all other cases the function computes a realization of the pre-observed vectors $X_0, X_{-1}, \ldots, X_{1-p}$, $\epsilon_0, \epsilon_{-1}, \ldots, \epsilon_{1-q}$, from (1), see Shea (1988). This realization is then used to generate a sequence of successive time series observations. Note that special action is taken for pure MA models, that is for $p = 0$.

At your request a new realization of the time series may be generated more efficiently using the information in a reference vector created during a previous call to nag_rand_times_mv_varma (g05pj). See the description of the argument **mode** in Section 5 for details.

The function returns a realization of $X_1, X_2, \ldots, X_n$. On a successful exit, the recent history is updated and saved in the array **r** so that nag_rand_times_mv_varma (g05pj) may be called again to generate a realization of $X_{n+1}, X_{n+2}, \ldots$, etc. See the description of the argument **mode** in Section 5 for details.

Further computational details are given in Shea (1988). Note, however, that nag_rand_times_mv_varma (g05pj) uses a spectral decomposition rather than a Cholesky factorization to generate the multivariate Normals. Although this method involves more multiplications than the Cholesky factorization method and is thus slightly slower it is more stable when faced with ill-conditioned covariance matrices. A method of assigning the AR and MA coefficient matrices so that the stationarity and invertibility conditions are satisfied is described in Barone (1987).

One of the initialization functions nag_rand_init_repeat (g05kf) (for a repeatable sequence if computed sequentially) or nag_rand_init_nonrepeat (g05kg) (for a non-repeatable sequence) must be called prior to the first call to nag_rand_times_mv_varma (g05pj).

4 References

Barone P (1987) A method for generating independent realisations of a multivariate normal stationary and invertible ARMA(p, q) process *J. Time Ser. Anal.* **8** 125–130

Shea B L (1988) A note on the generation of independent realisations of a vector autoregressive moving average process *J. Time Ser. Anal.* **9** 403–410

5 Parameters

5.1 Compulsory Input Parameters

1: **mode** – INTEGER

A code for selecting the operation to be performed by the function.

mode $= 0$
Set up reference vector and compute a realization of the recent history.

mode $= 1$
Generate terms in the time series using reference vector set up in a prior call to nag_rand_times_mv_varma (g05pj).

mode $= 2$
Combine the operations of **mode** $= 0$ and 1.

mode $= 3$
A new realization of the recent history is computed using information stored in the reference vector, and the following sequence of time series values are generated.

If **mode** $= 1$ or 3, then you must ensure that the reference vector **r** and the values of **k**, **ip**, **iq**, **xmean**, **phi**, **theta**, **var** and *ldvar* have not been changed between calls to nag_rand_times_mv_varma (g05pj).

Constraint: **mode** $= 0$, 1, 2 or 3.

2: **n** – INTEGER

n, the number of observations to be generated.

Constraint: **n** ≥ 0.

3: **xmean**(**k**) – REAL (KIND=nag_wp) array

μ, the vector of means of the multivariate time series.

4: **ip** – INTEGER

p, the number of autoregressive parameter matrices.

Constraint: **ip** ≥ 0.

5: **phi**(**k** × **k** × **ip**) – REAL (KIND=nag_wp) array

Must contain the elements of the **ip** × **k** × **k** autoregressive parameter matrices of the model, $\phi_1, \phi_2, \ldots, \phi_p$. If **phi** is considered as a three-dimensional array, dimensioned as **phi**(**k**, **k**, **ip**), then the (i,j)th element of ϕ_l would be stored in **phi**(i,j,l); that is, **phi**$((l-1) \times k \times k + (j-1) \times k + i)$ must be set equal to the (i,j)th element of ϕ_l, for $l = 1, 2, \ldots, p$, $i = 1, 2, \ldots, k$ and $j = 1, 2, \ldots, k$.

Constraint: the elements of **phi** must satisfy the stationarity condition.

6: **iq** – INTEGER

q, the number of moving average parameter matrices.

Constraint: **iq** ≥ 0.

7: **theta**(**k** × **k** × **iq**) – REAL (KIND=nag_wp) array

Must contain the elements of the **iq** × **k** × **k** moving average parameter matrices of the model, $\theta_1, \theta_2, \ldots, \theta_q$. If **theta** is considered as a three-dimensional array, dimensioned as **theta**(**k**,**k**,**iq**), then the (i,j)th element of θ_l would be stored in **theta**(i,j,l); that is, **theta**$((l-1) \times k \times k + (j-1) \times k + i)$ must be set equal to the (i,j)th element of θ_l, for $l = 1, 2, \ldots, q$, $i = 1, 2, \ldots, k$ and $j = 1, 2, \ldots, k$.

Constraint: the elements of **theta** must be within the invertibility region.

8: **var**($ldvar$, **k**) – REAL (KIND=nag_wp) array

$ldvar$, the first dimension of the array, must satisfy the constraint $ldvar \geq$ **k**.

var(i,j) must contain the (i,j)th element of Σ, for $i = 1, 2, \ldots,$ **k** and $j = 1, 2, \ldots,$ **k**. Only the lower triangle is required.

Constraint: the elements of **var** must be such that Σ is positive semidefinite.

9: **r**(**lr**) – REAL (KIND=nag_wp) array

If **mode** $= 1$ or 3, the array **r** as output from the previous call to nag_rand_times_mv_varma (g05pj) must be input without any change.

If **mode** $= 0$ or 2, the contents of **r** need not be set.

10: **state**(:) – INTEGER array

Note: the actual argument supplied **must** be the array **state** supplied to the initialization routines nag_rand_init_repeat (g05kf) or nag_rand_init_nonrepeat (g05kg).

Contains information on the selected base generator and its current state.

5.2 Optional Input Parameters

1: **k** – INTEGER

Default: the dimension of the array **xmean** and the first dimension of the array **var** and the second dimension of the array **var**. (An error is raised if these dimensions are not equal.)

k, the dimension of the multivariate time series.

Constraint: **k** ≥ 1.

2: **lr** – INTEGER

Default: the dimension of the array **r**.

The dimension of the array **r**.

Constraints:

if $\mathbf{k} \geq 6$, $\mathbf{lr} \geq \left(5r^2+1\right) \times \mathbf{k}^2 + (4r+3) \times \mathbf{k} + 4$;
if $\mathbf{k} < 6$, $\mathbf{lr} \geq \left((\mathbf{ip}+\mathbf{iq})^2+1\right) \times \mathbf{k}^2 +$
$(4 \times (\mathbf{ip}+\mathbf{iq})+3) \times \mathbf{k} + \max\left\{\mathbf{k}r(\mathbf{k}r+2), \mathbf{k}^2(\mathbf{ip}+\mathbf{iq})^2 + l(l+3) + \mathbf{k}^2(\mathbf{iq}+1)\right\} + 4$.

Where $r = \max(\mathbf{ip}, \mathbf{iq})$ and if $\mathbf{ip} = 0$, $l = \mathbf{k}(\mathbf{k}+1)/2$, or if $\mathbf{ip} \geq 1$, $l = \mathbf{k}(\mathbf{k}+1)/2 + (\mathbf{ip}-1)\mathbf{k}^2$.

See Section 9 for some examples of the required size of the array **r**.

5.3 Output Parameters

1: **r**(**lr**) – REAL (KIND=nag_wp) array

Information required for any subsequent calls to the function with **mode** $= 1$ or 3. See Section 9.

2: **state**(:) – INTEGER array

Contains updated information on the state of the generator.

3: $\mathbf{x}(ldx, \mathbf{n})$ – REAL (KIND=nag_wp) array

$\mathbf{x}(i,t)$ will contain a realization of the ith component of X_t, for $i = 1, 2, \ldots, k$ and $t = 1, 2, \ldots, n$.

4: **ifail** – INTEGER

ifail $= 0$ unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

Constraint: **mode** $= 0$, 1, 2 or 3.

ifail $= 2$

Constraint: $\mathbf{n} \geq 0$.

ifail $= 3$

Constraint: $\mathbf{k} \geq 1$.

ifail $= 5$

Constraint: $\mathbf{ip} \geq 0$.

ifail $= 6$

On entry, the AR parameters are outside the stationarity region.

ifail $= 7$

Constraint: $\mathbf{iq} \geq 0$.

ifail $= 8$

On entry, the moving average parameter matrices are such that the model is non-invertible.

ifail $= 9$

On entry, the covariance matrix **var** is not positive semidefinite to ***machine precision***.

ifail $= 10$

Constraint: $ldvar \geq$ **k**.

ifail $= 11$

k is not the same as when **r** was set up in a previous call.

ifail $= 12$

On entry, **lr** is not large enough, **lr** $= \langle value \rangle$: minimum length required .

ifail $= 13$

On entry, **state** vector has been corrupted or not initialized.

ifail $= 15$

Constraint: $ldx \geq$ **k**.

ifail $= 20$

An excessive number of iterations were required by the NAG function used to evaluate the eigenvalues of the matrices used to test for stationarity or invertibility.

ifail $= 21$

The reference vector cannot be computed because the AR parameters are too close to the boundary of the stationarity region.

ifail $= 22$

An excessive number of iterations were required by the NAG function used to evaluate the eigenvalues of the covariance matrix.

ifail $= 23$ (*warning*)

An excessive number of iterations were required by the NAG function used to evaluate the eigenvalues stored in the reference vector.

ifail $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

ifail $= -399$

Your licence key may have expired or may not have been installed correctly.

ifail $= -999$

Dynamic memory allocation failed.

7 Accuracy

The accuracy is limited by the matrix computations performed, and this is dependent on the condition of the argument and covariance matrices.

8 Further Comments

Note that, in reference to **ifail** $= 8$, nag_rand_times_mv_varma (g05pj) will permit moving average parameters on the boundary of the invertibility region.

The elements of **r** contain amongst other information details of the spectral decompositions which are used to generate future multivariate Normals. Note that these eigenvectors may not be unique on different machines. For example the eigenvectors corresponding to multiple eigenvalues may be

permuted. Although an effort is made to ensure that the eigenvectors have the same sign on all machines, differences in the signs may theoretically still occur.

The following table gives some examples of the required size of the array **r**, specified by the argument **lr**, for $k = 1, 2$ or 3, and for various values of p and q.

		q			
		0	1	2	3
		13	20	31	46
	0	36	56	92	144
		85	124	199	310
		19	30	45	64
	1	52	88	140	208
		115	190	301	448
p					
		35	50	69	92
	2	136	188	256	340
		397	508	655	838
		57	76	99	126
	3	268	336	420	520
		877	1024	1207	1426

Note that nag_tsa_uni_arma_roots (g13dx) may be used to check whether a VARMA model is stationary and invertible.

The time taken depends on the values of p, q and especially n and k.

9 Example

This program generates two realizations, each of length 48, from the bivariate AR(1) model

$$X_t - \mu = \phi_1(X_{t-1} - \mu) + \epsilon_t$$

with

$$\phi_1 = \begin{bmatrix} 0.80 & 0.07 \\ 0.00 & 0.58 \end{bmatrix},$$

$$\mu = \begin{bmatrix} 5.00 \\ 9.00 \end{bmatrix},$$

and

$$\Sigma = \begin{bmatrix} 2.97 & 0 \\ 0.64 & 5.38 \end{bmatrix}.$$

The pseudorandom number generator is initialized by a call to nag_rand_init_repeat (g05kf). Then, in the first call to nag_rand_times_mv_varma (g05pj), **mode** $= 2$ in order to set up the reference vector before generating the first realization. In the subsequent call **mode** $= 3$ and a new recent history is generated and used to generate the second realization.

9.1 Program Text

```
     function g05pj_example

fprintf('g05pj example results\n\n');

% Initialize the generator to a repeatable sequence
seed  = [nag_int(1762543)];
genid = nag_int(1);
subid = nag_int(1);
[state, ifail] = g05kf( ...
                        genid, subid, seed);

n     = nag_int(48);
xmean = [5; 9];
ip    = nag_int(1);
iq    = nag_int(0);
phi   = [0.8; 0; 0.07; 0.58];
theta = [];
var   = [2.97, 0; 0.64, 5.38];
r     = zeros(600, 1);

% Generate the first realisation
% Use mode = 2 to set up R and generate values
mode = nag_int(2);
[r, state, x, ifail] = g05pj( ...
                              mode, n, xmean, ip, phi, iq, ...
                              theta, var, r, state);
% Display the results
fprintf(' Realisation Number 1\n\n');
for i=1:2
  fprintf(' Series number %d\n', i);
  fprintf(' ---------------\n');
  for j=1:6
    fprintf(' %8.3f', x(i, (j-1)*8+1:j*8));
    fprintf('\n');
  end
  fprintf('\n');
end
fprintf('\n');

% Generate a second realisation
% Use mode = 3 to reset the series and generate values
mode = nag_int(3);
[r, state, x, ifail] = g05pj( ...
                              mode, n, xmean, ip, phi, iq, ...
                              theta, var, r, state);
% Display the results
fprintf(' Realisation Number 2\n\n');
for i=1:2
  fprintf(' Series number %d\n', i);
  fprintf(' ---------------\n');
  for j=1:6
    fprintf(' %8.3f', x(i, (j-1)*8+1:j*8));
    fprintf('\n');
  end
  fprintf('\n');
end
fprintf('\n');
```

9.2 Program Results

```
     g05pj example results

 Realisation Number 1

 Series number 1
 ---------------
    4.833    2.813    3.224    3.825    1.023    1.415    2.184    3.005
    5.547    4.832    4.705    5.484    9.407   10.335    8.495    7.478
```

```
   6.373    6.692    6.698    6.976    6.200    4.458    2.520    3.517
   3.054    5.439    5.699    7.136    5.750    8.497    9.563   11.604
   9.020   10.063    7.976    5.927    4.992    4.222    3.982    7.107
   3.554    7.045    7.025    4.106    5.106    5.954    8.026    7.212

Series number 2
---------------
   8.458    9.140   10.866   10.975    9.245    5.054    5.023   12.486
  10.534   10.590   11.376    8.793   14.445   13.237   11.030    8.405
   7.187    8.291    5.920    9.390   10.055    6.222    7.751   10.604
  12.441   10.664   10.960    8.022   10.073   12.870   12.665   14.064
  11.867   12.894   10.546   12.754    8.594    9.042   12.029   12.557
   9.746    5.487    5.500    8.629    9.723    8.632    6.383   12.484

Realisation Number 2

Series number 1
---------------
   5.396    4.811    2.685    5.824    2.449    3.563    5.663    6.209
   3.130    4.308    4.333    4.903    1.770    1.278    1.340   -0.527
   1.745    3.211    4.478    5.170    5.365    4.852    6.080    6.464
   2.765    2.148    6.641    7.224   10.316    7.102    5.604    3.934
   4.839    3.698    5.210    5.384    7.652    7.315    7.332    7.561
   7.537    7.788    6.868    7.575    6.108    6.188    8.132   10.310

Series number 2
---------------
  11.345   10.070   13.654   12.409   11.329   13.054   12.465    9.867
  10.263   13.394   10.553   10.331    7.814    8.747   10.025   11.167
  10.626    9.366    9.607    9.662   10.492   10.766   11.512   10.813
  10.799    8.780    9.221   14.245   11.575   10.620    8.282    5.447
   9.935    9.386   11.627   10.066   11.394    7.951    7.907   12.616
  15.246    9.962   13.216   11.350   11.227    6.021    6.968   12.428
```