NAG C Library Function Document

nag_tsa_arma_roots (g13dxc)

1 Purpose

G13DXF calculates the zeros of a vector autoregressive (or moving average) operator.

2 Specification

```
void nag_tsa_arma_roots (Integer k, Integer ip, const double par[], double rr[],
     double ri[], double rmod[], NagError *fail)
```

3 Description

Consider the vector autoregressive moving average (VARMA) model

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

where W_t denotes a vector of k time series and ϵ_t is a vector of k residual series having zero mean and a constant variance-covariance matrix. The components of ϵ_t are also assumed to be uncorrelated at non-simultaneous lags. $\phi_1, \phi_2, \ldots, \phi_p$ denotes a sequence of k by k matrices of autoregressive (AR) parameters and $\theta_1, \theta_2, \ldots, \theta_q$ denotes a sequence of k by k matrices of moving average (MA) parameters. μ is a vector of length k containing the series means. Let

$$A(\phi) = \begin{bmatrix} \phi_1 & I & 0 & . & . & . & 0 \\ \phi_2 & 0 & I & 0 & . & . & 0 \\ . & & & . & & & \\ . & & & & . & & \\ . & & & & & . & \\ \phi_{p-1} & 0 & . & . & . & 0 & I \\ \phi_p & 0 & . & . & . & 0 & 0 \end{bmatrix}_{pk \times pk}$$

where I denotes the k by k identity matrix.

The model (1) is said to be stationary if the eigenvalues of $A(\phi)$ lie inside the unit circle. Similarly let

$$B(\theta) = \begin{bmatrix} \theta_1 & I & 0 & . & . & . & 0 \\ \theta_2 & 0 & I & 0 & . & . & 0 \\ . & & & . & & & \\ . & & & & . & & \\ . & & & & & . & \\ \theta_{q-1} & 0 & . & . & . & 0 & I \\ \theta_q & 0 & . & . & . & 0 & 0 \end{bmatrix}_{qk \times qk} .$$

Then the model is said to be invertible if the eigenvalues of $B(\theta)$ lie inside the unit circle.

nag_tsa_arma_roots (g13dxc) returns the pk eigenvalues of $A(\phi)$ (or the qk eigenvalues of $B(\theta)$) along with their moduli, in descending order of magnitude. Thus to check for stationarity or invertibility the user should check whether the modulus of the largest eigenvalue is less than one.

4 References

Wei W W S (1990) *Time Series Analysis: Univariate and Multivariate Methods* Addison–Wesley

5 Parameters

1: **k** – Integer *Input*

On entry: the dimension, k, of the multivariate time series.

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

2: **ip** – Integer *Input*

On entry: the number of AR (or MA) parameter matrices, p (or q).

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

3: **par**[dim] – const double *Input*

Note: the dimension, dim, of the array **par** must be at least $\mathbf{ip} \times \mathbf{k} \times \mathbf{k}$.

On entry: the AR (or MA) parameter matrices read in row by row in the order $\phi_1, \phi_2, \ldots, \phi_p$ (or $\theta_1, \theta_2, \ldots, \theta_q$). That is, $\mathbf{par}[(l-1) \times k \times k + (i-1) \times k + j - 1]$ must be set equal to the (i, j)th element of ϕ_l, for $l = 1, 2, \ldots, p$ (or the (i, j)th element of θ_l, for $l = 1, 2, \ldots, q$).

4: **rr**[dim] – double *Output*

Note: the dimension, dim, of the array **rr** must be at least $\mathbf{k} \times \mathbf{ip}$.

On exit: the real parts of the eigenvalues.

5: **ri**[dim] – double *Output*

Note: the dimension, dim, of the array **ri** must be at least $\mathbf{k} \times \mathbf{ip}$.

On exit: the imaginary parts of the eigenvalues.

6: **rmod**[dim] – double *Output*

Note: the dimension, dim, of the array **rmod** must be at least $\mathbf{k} \times \mathbf{ip}$.

On exit: the moduli of the eigenvalues.

7: **fail** – NagError * *Input/Output*

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, $\mathbf{ip} = \langle value \rangle$.
Constraint: $\mathbf{ip} \geq 1$.

On entry, $\mathbf{k} = \langle value \rangle$.
Constraint: $\mathbf{k} \geq 1$.

NE_EIGENVALUES

An excessive number of iterations have been required to calculate the eigenvalues.

NE_ALLOC_FAIL

Memory allocation failed.

NE_BAD_PARAM

On entry, parameter $\langle value \rangle$ had an illegal value.

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please consult NAG for assistance.

7 Accuracy

The accuracy of the results depends on the original matrix and the multiplicity of the roots.

8 Further Comments

The time taken is approximately proportional to kp^3 (or kq^3).

9 Example

This example program finds the eigenvalues of $A(\phi)$ where $k = 2$ and $p = 1$ and $\phi_1 = \begin{bmatrix} 0.802 & 0.065 \\ 0.000 & 0.575 \end{bmatrix}$.

9.1 Program Text

```
/* nag_tsa_arma_roots (g13dxc) Example Program.
 *
 * Copyright 2002 Numerical Algorithms Group.
 *
 * Mark 7, 2002.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg13.h>

int main(void)
{
  /* Scalars */
  Integer exit_status, i, ip, k, npar;
  NagError fail;

  /* Arrays */
  double *par = 0, *ri = 0, *rmod = 0, *rr = 0;

  INIT_FAIL(fail);
  exit_status = 0;

  Vprintf("g13dxc Example Program Results\n");

  /* Skip heading in data file */
  Vscanf("%*[^\n] ");

  Vscanf("%ld%ld%*[^\n] ", &k, &ip);

  if (k > 0 && ip > 0)
    {
      /* Allocate arrays */
      if ( !(par = NAG_ALLOC(k*k*ip, double)) ||
           !(ri = NAG_ALLOC(k*ip, double)) ||
           !(rmod = NAG_ALLOC(k*ip, double)) ||
           !(rr = NAG_ALLOC(k*ip, double)) )
        {
          Vprintf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }

      /* Read the AR (or MA) parameters */
      npar = ip * k * k;
```

```
      for (i = 1; i <= npar; ++i)
        Vscanf("%lf", &par[i-1]);
      Vscanf("%*[^\n] ");

      g13dxc(k, ip, par, rr, ri, rmod, &fail);
      if (fail.code != NE_NOERROR)
        {
          Vprintf("Error from g13dxc.\n%s\n",   fail.message);
          exit_status = 1;
          goto END;
        }
      Vprintf("\n");
      Vprintf("        Eigenvalues       Moduli\n");
      Vprintf("        -----------       ------\n");
      for (i = 1; i <= k * ip; ++i)
        {
          if (ri[i-1] >= 0.0)
            Vprintf("%10.3f  + %6.3f i  %8.3f\n", rr[i-1], ri[i-1], rmod[i-1]);
          else
            Vprintf("%10.3f  - %6.3f i  %8.3f\n", rr[i-1], -ri[i-1], rmod[i-1]);
        }
    }
  else
    Vprintf(" Either k or ip is out of range\n");

 END:
  if (par) NAG_FREE(par);
  if (ri) NAG_FREE(ri);
  if (rmod) NAG_FREE(rmod);
  if (rr) NAG_FREE(rr);

  return exit_status;
}
```

9.2 Program Data

```
g13dxc Example Program Data
 2 1
 0.802 0.065
 0.000 0.575
```

9.3 Program Results

```
g13dxc Example Program Results

        Eigenvalues       Moduli
        -----------       ------
     0.802  +  0.000 i     0.802
     0.575  +  0.000 i     0.575
```