NAG Library Function Document

nag_tsa_arma_roots (g13dxc)

1 Purpose

nag_tsa_arma_roots (g13dxc) calculates the zeros of a vector autoregressive (or moving average) operator.

2 Specification

```
#include <nag.h>
#include <nagg13.h>
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

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

where 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 you 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 Arguments

1: **k** – Integer *Input*

On entry: k, the dimension 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**[**ip** × **k** × **k**] – const double *Input*

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, **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**[**k** × **ip**] – double *Output*

On exit: the real parts of the eigenvalues.

5: **ri**[**k** × **ip**] – double *Output*

On exit: the imaginary parts of the eigenvalues.

6: **rmod**[**k** × **ip**] – double *Output*

On exit: the moduli of the eigenvalues.

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

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 3.2.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_EIGENVALUES

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

NE_INT

On entry, **ip** = ⟨*value*⟩.
Constraint: $\mathbf{ip} \geq 1$.

On entry, **k** = ⟨*value*⟩.
Constraint: $\mathbf{k} \geq 1$.

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 contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 3.6.6 in How to Use the NAG Library and its Documentation for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

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

8 Parallelism and Performance

nag_tsa_arma_roots (g13dxc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_tsa_arma_roots (g13dxc) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the x06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users' Notefor your implementation for any additional implementation-specific information.

9 Further Comments

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

10 Example

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

10.1 Program Text

```
/* nag_tsa_arma_roots (g13dxc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

#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;

  printf("nag_tsa_arma_roots (g13dxc) Example Program Results\n");
```

```
  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

#ifdef _WIN32
  scanf_s("%" NAG_IFMT "%" NAG_IFMT "%*[^\n] ", &k, &ip);
#else
  scanf("%" NAG_IFMT "%" NAG_IFMT "%*[^\n] ", &k, &ip);
#endif

  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)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

    /* Read the AR (or MA) parameters */
    npar = ip * k * k;
    for (i = 1; i <= npar; ++i)
#ifdef _WIN32
      scanf_s("%lf", &par[i - 1]);
#else
      scanf("%lf", &par[i - 1]);
#endif
#ifdef _WIN32
    scanf_s("%*[^\n] ");
#else
    scanf("%*[^\n] ");
#endif

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

END:
  NAG_FREE(par);
  NAG_FREE(ri);
```

```
  NAG_FREE(rmod);
  NAG_FREE(rr);

  return exit_status;
}
```

10.2 Program Data

```
nag_tsa_arma_roots (g13dxc) Example Program Data
 2 1
 0.802 0.065
 0.000 0.575
```

10.3 Program Results

```
nag_tsa_arma_roots (g13dxc) Example Program Results

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