NAG Library Function Document

nag_tsa_multi_part_regsn (g13dpc)

1 Purpose

nag_tsa_multi_part_regsn (g13dpc) calculates the sample partial autoregression matrices of a multivariate time series. A set of likelihood ratio statistics and their significance levels are also returned. These quantities are useful for determining whether the series follows an autoregressive model and, if so, of what order.

2 Specification

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

```
void nag_tsa_multi_part_regsn (Integer k, Integer n, const double z[],
     Integer m, Integer *maxlag, double parlag[], double se[], double qq[],
     double x[], double pvalue[], double loglhd[], NagError *fail)
```

3 Description

Let $W_t = (w_{1t}, w_{2t}, \ldots, w_{kt})^{\mathrm{T}}$, for $t = 1, 2, \ldots, n$, denote a vector of k time series. The partial autoregression matrix at lag l, P_l, is defined to be the last matrix coefficient when a vector autoregressive model of order l is fitted to the series. P_l has the property that if W_t follows a vector autoregressive model of order p then $P_l = 0$ for $l > p$.

Sample estimates of the partial autoregression matrices may be obtained by fitting autoregressive models of successively higher orders by multivariate least squares; see Tiao and Box (1981) and Wei (1990). These models are fitted using a QR algorithm based on the functions nag_regsn_mult_linear_addrem_obs (g02dcc) and nag_regsn_mult_linear_delete_var (g02dfc). They are calculated up to lag m, which is usually taken to be at most $n/4$.

The function also returns the asymptotic standard errors of the elements of $\hat{P}_l$ and an estimate of the residual variance-covariance matrix $\hat{\Sigma}_l$, for $l = 1, 2, \ldots, m$. If S_l denotes the residual sum of squares and cross-products matrix after fitting an AR(l) model to the series then under the null hypothesis $H_0 : P_l = 0$ the test statistic

$$X_l = -\big((n - m - 1) - \tfrac{1}{2} - lk\big)\log\left(\frac{|S_l|}{|S_{l-1}|}\right)$$

is asymptotically distributed as χ^2 with k^2 degrees of freedom. X_l provides a useful diagnostic aid in determining the order of an autoregressive model. (Note that $\hat{\Sigma}_l = S_l/(n - l)$.) The function also returns an estimate of the maximum of the log-likelihood function for each AR model that has been fitted.

4 References

Tiao G C and Box G E P (1981) Modelling multiple time series with applications *J. Am. Stat. Assoc.* **76** 802–816

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

5 Arguments

1: **k** – Integer *Input*

On entry: k, the number of time series.

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

2: **n** – Integer *Input*

On entry: n, the number of observations in the time series.

Constraint: $\mathbf{n} \geq 4$.

3: **z**[**k** × **n**] – const double *Input*

On entry: $\mathbf{z}[(t-1)k+i-1]$ must contain the value for the ith series at time t, for $i = 1, 2, \ldots, k$ and $t = 1, 2, \ldots, n$.

4: **m** – Integer *Input*

On entry: m, the number of partial autoregression matrices to be computed. If in doubt set $\mathbf{m} = 10$.

Constraint: $\mathbf{m} \geq 1$ and $\mathbf{n} - \mathbf{m} - (\mathbf{k} \times \mathbf{m} + 1) \geq \mathbf{k}$.

5: **maxlag** – Integer * *Output*

On exit: the maximum lag up to which partial autoregression matrices (along with their likelihood ratio statistics and their significance levels) have been successfully computed. On a successful exit **maxlag** will equal **m**. If **fail**.**code** = MATRIX_ILL_CONDITIONED on exit then **maxlag** will be less than **m**.

6: **parlag**[**k** × **k** × **m**] – double *Output*

On exit: $\mathbf{parlag}[(l-1)k^2+(j-1)k+i-1]$ contains an estimate of the (i,j)th element of the partial autoregression matrix at lag l, for $l = 1, 2, \ldots, \mathbf{maxlag}$, $i = 1, 2, \ldots, k$ and $j = 1, 2, \ldots, k$.

7: **se**[**k** × **k** × **m**] – double *Output*

On exit: $\mathbf{se}[(l-1)k^2+(j-1)k+i-1]$ contains an estimate of the standard error of the corresponding element in **parlag**.

8: **qq**[**k** × **k** × **m**] – double *Output*

On exit: $\mathbf{qq}[(l-1)k^2+(j-1)k+i-1]$ contains an estimate of the (i,j)th element of the residual variance-covariance matrix, $\hat{\Sigma}_l$, for $l = 1, 2, \ldots, \mathbf{maxlag}$, $i = 1, 2, \ldots, k$ and $j = 1, 2, \ldots, k$.

9: **x**[**m**] – double *Output*

On exit: $\mathbf{x}[l-1]$ contains X_l, the likelihood ratio statistic at lag l, for $l = 1, 2, \ldots, \mathbf{maxlag}$.

10: **pvalue**[**m**] – double *Output*

On exit: $\mathbf{pvalue}[l-1]$ contains the significance level of the statistic in the corresponding element of **x**.

11: **loglhd**[**m**] – double *Output*

On exit: $\mathbf{loglhd}[l-1]$ contains an estimate of the maximum of the log-likelihood function when an AR$(l-1)$ model has been fitted to the series, for $l = 1, 2, \ldots, \mathbf{maxlag}$.

12: **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

MATRIX_ILL_CONDITIONED

The recursive equations used to compute the partial autoregression matrices are ill-conditioned. They have been computed up to lag $\langle\mathit{value}\rangle$.

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

On entry, argument $\langle\mathit{value}\rangle$ had an illegal value.

NE_INT

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

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

On entry, $\mathbf{n} = \langle\mathit{value}\rangle$.
Constraint: $\mathbf{n} \geq 4$.

NE_INT_3

On entry, $\mathbf{k} = \langle\mathit{value}\rangle$, $\mathbf{m} = \langle\mathit{value}\rangle$ and $\mathbf{n} = \langle\mathit{value}\rangle$.
Constraint: $\mathbf{n} - \mathbf{m} - (\mathbf{k} \times \mathbf{m} + 1) \geq \mathbf{k}$.

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 2.7.6 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 2.7.5 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

The computations are believed to be stable.

8 Parallelism and Performance

nag_tsa_multi_part_regsn (g13dpc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_tsa_multi_part_regsn (g13dpc) 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' Note for your implementation for any additional implementation-specific information.

9 Further Comments

The time taken is roughly proportional to nmk.

For each order of autoregressive model that has been estimated, nag_tsa_multi_part_regsn (g13dpc) returns the maximum of the log-likelihood function. An alternative means of choosing the order of a vector AR process is to choose the order for which Akaike's information criterion is smallest. That is, choose the value of l for which $-2 \times \mathbf{loglhd}[l] + 2lk^2$ is smallest. You should be warned that this does not always lead to the same choice of l as indicated by the sample partial autoregression matrices and the likelihood ratio statistics.

10 Example

This example computes the sample partial autoregression matrices of two time series of length 48 up to lag 10.

10.1 Program Text

```
/* nag_tsa_multi_part_regsn (g13dpc) 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>

static void zprint(Integer, Integer, double *,
                   double *, double *, double *, double *);

int main(void)
{
  /* Scalars */
  Integer exit_status, i, j, k, m, maxlag, n, kmax;
  NagError fail;

  /* Arrays */
  double *loglhd = 0, *parlag = 0, *pvalue = 0, *qq = 0, *se = 0, *z = 0;
  double *x = 0;

#define Z(I, J) z[(J-1)*kmax + I - 1]

  INIT_FAIL(fail);

  exit_status = 0;

  printf("nag_tsa_multi_part_regsn (g13dpc) 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 "%" NAG_IFMT "%*[^\n] ", &k, &n, &m);
```

```
#else
  scanf("%" NAG_IFMT "%" NAG_IFMT "%" NAG_IFMT "%*[^\n] ", &k, &n, &m);
#endif

  if (k > 0 && n >= 1 && m >= 1) {
    /* Allocate arrays */
    if (!(loglhd = NAG_ALLOC(m, double)) ||
        !(parlag = NAG_ALLOC(k * k * m, double)) ||
        !(pvalue = NAG_ALLOC(m, double)) ||
        !(qq = NAG_ALLOC(k * k * m, double)) ||
        !(se = NAG_ALLOC(k * k * m, double)) ||
        !(z = NAG_ALLOC(k * n, double)) || !(x = NAG_ALLOC(m, double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

    kmax = k;

    for (i = 1; i <= k; ++i) {
      for (j = 1; j <= n; ++j)
#ifdef _WIN32
        scanf_s("%lf", &Z(i, j));
#else
        scanf("%lf", &Z(i, j));
#endif
#ifdef _WIN32
      scanf_s("%*[^\n] ");
#else
      scanf("%*[^\n] ");
#endif
    }

    /* nag_tsa_multi_part_regsn (g13dpc).
     * Multivariate time series, partial autoregression matrices
     */
    nag_tsa_multi_part_regsn(k, n, z, m, &maxlag, parlag, se, qq, x, pvalue,
                             loglhd, &fail);
    if (fail.code != NE_NOERROR) {
      printf("Error from nag_tsa_multi_part_regsn (g13dpc).\n%s\n",
             fail.message);
      exit_status = 1;
      goto END;
    }
    zprint(k, maxlag, parlag, se, qq, x, pvalue);
  }

END:
  NAG_FREE(loglhd);
  NAG_FREE(parlag);
  NAG_FREE(pvalue);
  NAG_FREE(qq);
  NAG_FREE(se);
  NAG_FREE(z);
  NAG_FREE(x);

  return exit_status;
}

static void zprint(Integer k, Integer maxlag, double *parlag, double *se,
                   double *qq, double *x, double *pvalue)
{
  /* Scalars */
  double sum;
  Integer i2, i, j, lf;

  /* Arrays */
  char st[7];

#define SE(I, J, K)     se[((K-1)*k + (J-1))*k + I - 1]
```

```
#define QQ(I, J, K)     qq[((K-1)*k + (J-1))*k + I - 1]
#define PARLAG(I, J, K) parlag[((K-1)*k + (J-1))*k + I - 1]

  if (k > 1) {
    printf("\n");
    printf(" Partial Autoregression Matrices    Indicator"
           "  Residual   Chi-Square  Pvalue\n");
    printf("                                     Symbols"
           "   Variances   Statistic\n");
    printf(" -------------------------------    ---------"
           "  ---------  ----------- ------\n");
  }

  if (k == 1) {
    printf("\n");
    printf(" Partial Autoregression Function    Indicator"
           "  Residual   Chi-Square  Pvalue\n");
    printf("                                     Symbols");
    printf("   Variances");
    printf("   Statistic\n");
    printf(" -------------------------------    ---------"
           "  ---------  ----------- ------\n");
  }

  for (lf = 1; lf <= maxlag; ++lf) {
    for (j = 1; j <= k; ++j) {
      sum = PARLAG(1, j, lf);
      st[j] = '.';
      if (sum > SE(1, j, lf) * 1.96)
        st[j] = '+';
      if (sum < SE(1, j, lf) * -1.96)
        st[j] = '-';
    }

    if (k == 1) {
      printf("\n");
      printf(" Lag %2" NAG_IFMT " :", lf);
      for (j = 1; j <= k; ++j) {
        printf("%6.3f", PARLAG(1, j, lf));
        printf("%14s", "");
      }
      for (i2 = 1; i2 <= k; ++i2)
        printf("%c", st[i2]);
      printf("%14.3f%13.3f%9.3f\n", QQ(1, 1, lf), x[lf - 1], pvalue[lf - 1]);
      printf("           ");
      for (j = 1; j <= k; ++j)
        printf("(%6.3f ) ", SE(1, j, lf));
      printf("\n");
    }
    else if (k == 2) {
      printf("\n");
      printf(" Lag %2" NAG_IFMT " :", lf);
      for (j = 1; j <= k; ++j)
        printf("%8.3f", PARLAG(1, j, lf));
      printf("%14s", "");
      for (i2 = 1; i2 <= k; ++i2)
        printf("%c", st[i2]);
      printf("%13.3f %12.3f %8.3f\n", QQ(1, 1, lf), x[lf - 1],
             pvalue[lf - 1]);
      printf("           ");
      for (j = 1; j <= k; ++j)
        printf("(%5.3f) ", SE(1, j, lf));
      printf("\n");
    }
    else if (k == 3) {
      printf("\n");
      printf(" Lag %2" NAG_IFMT " :", lf);
      for (j = 1; j <= k; ++j)
        printf("%8.3f      ", PARLAG(1, j, lf));
      for (i2 = 1; i2 <= k; ++i2)
        printf("%c", st[i2]);
      printf("%12.3f%13.3f%9.3f\n", QQ(1, 1, lf), x[lf - 1], pvalue[lf - 1]);
      printf("           ");
      for (j = 1; j <= k; ++j)
        printf("(%5.3f) ", SE(1, j, lf));
      printf("\n");
    }
    else if (k == 4) {
      printf("\n");
      printf(" Lag %2" NAG_IFMT "\n", lf);
      for (j = 1; j <= k; ++j)
        printf("%8.3f     ", PARLAG(1, j, lf));
      for (i2 = 1; i2 <= k; ++i2)
        printf("%c", st[i2]);
      printf("%12.3f%13.3f%9.3f\n", QQ(1, 1, lf), x[lf - 1], pvalue[lf - 1]);
      printf("   ");
      for (j = 1; j <= k; ++j)
        printf("(%5.3f) ", SE(1, j, lf));
      printf("\n");
    }

    for (i = 2; i <= k; ++i) {
      for (j = 1; j <= k; ++j) {
        sum = PARLAG(i, j, lf);
        st[j] = '.';
        if (sum > SE(i, j, lf) * 1.96)
          st[j] = '+';
        if (sum < SE(i, j, lf) * -1.96)
          st[j] = '-';
      }
      if (k == 2) {
        printf("         ");
        for (j = 1; j <= k; ++j)
          printf("%8.3f", PARLAG(i, j, lf));
        printf("              ");
        for (i2 = 1; i2 <= k; ++i2)
          printf("%c", st[i2]);
        printf("%13.3f\n", QQ(i, i, lf));
        printf("           ");
        for (j = 1; j <= k; ++j)
          printf("(%5.3f) ", SE(i, j, lf));
        printf("\n");
      }
      else if (k == 3) {
        printf("         ");
        for (j = 1; j <= k; ++j)
          printf("%8.3f      ", PARLAG(i, j, lf));
        for (i2 = 1; i2 <= k; ++i2)
          printf("%c", st[i2]);
        printf("%12.3f\n", QQ(i, i, lf));
        printf("           ");
        for (j = 1; j <= k; ++j)
          printf("(%5.3f) ", SE(i, j, lf));
        printf("\n");
      }
      else if (k == 4) {
        for (j = 1; j <= k; ++j)
          printf("%8.3f     ", PARLAG(i, j, lf));
        for (i2 = 1; i2 <= k; ++i2)
          printf("%c", st[i2]);
        printf("%12.3f\n", QQ(i, i, lf));
        printf("   ");
        for (j = 1; j <= k; ++j)
          printf("(%5.3f) ", SE(i, j, lf));
        printf("\n");
      }
    }
  }

  return;
}
```

10.2 Program Data

```
nag_tsa_multi_part_regsn (g13dpc) Example Program Data
2 48 10 :  k, no. of series,  n, no. of obs in each series,  m, no. of lags
-1.490 -1.620  5.200  6.230  6.210  5.860  4.090  3.180
 2.620  1.490  1.170  0.850 -0.350  0.240  2.440  2.580
 2.040  0.400  2.260  3.340  5.090  5.000  4.780  4.110
 3.450  1.650  1.290  4.090  6.320  7.500  3.890  1.580
 5.210  5.250  4.930  7.380  5.870  5.810  9.680  9.070
 7.290  7.840  7.550  7.320  7.970  7.760  7.000  8.350
 7.340  6.350  6.960  8.540  6.620  4.970  4.550  4.810
 4.750  4.760 10.880 10.010 11.620 10.360  6.400  6.240
 7.930  4.040  3.730  5.600  5.350  6.810  8.270  7.680
 6.650  6.080 10.250  9.140 17.750 13.300  9.630  6.800
 4.080  5.060  4.940  6.650  7.940 10.760 11.890  5.850
 9.010  7.500 10.020 10.380  8.150  8.370 10.730 12.140 : End of time series
```

10.3 Program Results

```
nag_tsa_multi_part_regsn (g13dpc) Example Program Results

 Partial Autoregression Matrices    Indicator  Residual   Chi-Square  Pvalue
                                     Symbols   Variances   Statistic
 -------------------------------    ---------  ---------  ----------- ------

 Lag  1 :   0.757   0.062              +.        2.731       49.884    0.000
          (0.092) (0.092)
            0.061   0.570              .+        5.440
          (0.129) (0.130)

 Lag  2 :  -0.161  -0.135              ..        2.530        3.347    0.502
          (0.145) (0.109)
           -0.093  -0.065              ..        5.486
          (0.213) (0.160)

 Lag  3 :   0.237   0.044              ..        1.755       13.962    0.007
          (0.128) (0.095)
            0.047  -0.248              ..        5.291
          (0.222) (0.165)

 Lag  4 :  -0.098   0.152              ..        1.661        7.071    0.132
          (0.134) (0.099)
            0.402  -0.194              ..        4.786
          (0.228) (0.168)

 Lag  5 :   0.257  -0.026              ..        1.504        5.184    0.269
          (0.141) (0.106)
            0.400  -0.021              ..        4.447
          (0.242) (0.183)

 Lag  6 :  -0.075   0.112              ..        1.480        2.083    0.721
          (0.156) (0.111)
            0.196  -0.106              ..        4.425
          (0.269) (0.192)

 Lag  7 :  -0.054   0.097              ..        1.478        5.074    0.280
          (0.166) (0.121)
            0.574  -0.080              +.        3.838
          (0.267) (0.195)

 Lag  8 :   0.147   0.041              ..        1.415       10.991    0.027
          (0.188) (0.128)
            0.916  -0.242              +.        2.415
          (0.246) (0.167)

 Lag  9 :  -0.039   0.099              ..        1.322        3.936    0.415
          (0.251) (0.140)
           -0.500   0.173              ..        2.196
          (0.324) (0.181)
 Lag 10 :   0.189   0.131                ..         1.206         3.175    0.529
          (0.275) (0.157)
          -0.183  -0.040                 ..         2.201
          (0.371) (0.212)
```