NAG Library Function Document

nag_tsa_inhom_iema (g13mec)

1 Purpose

nag_tsa_inhom_iema (g13mec) calculates the iterated exponential moving average for an inhomogeneous time series.

2 Specification

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

```
void nag_tsa_inhom_iema (Integer nb, double iema[], const double t[],
     double tau, Integer m, const double sinit[],
     const Nag_TS_Interpolation inter[], Integer *pn, double rcomm[],
     NagError *fail)
```

3 Description

nag_tsa_inhom_iema (g13mec) calculates the iterated exponential moving average for an inhomogeneous time series. The time series is represented by two vectors of length n; a vector of times, t; and a vector of values, z. Each element of the time series is therefore composed of the pair of scalar values (t_i, z_i), for $i = 1, 2, \ldots, n$. Time can be measured in any arbitrary units, as long as all elements of t use the same units.

The exponential moving average (EMA), with parameter τ, is an average operator, with the exponentially decaying kernel given by

$$\frac{e^{-t_i/\tau}}{\tau}.$$

The exponential form of this kernel gives rise to the following iterative formula for the EMA operator (see Zumbach and Müller (2001)):

$$\mathrm{EMA}[\tau; z](t_i) = \mu \mathrm{EMA}[\tau; z](t_{i-1}) + (\nu - \mu) z_{i-1} + (1 - \nu) z_i$$

where

$$\mu = e^{-\alpha} \quad \text{and} \quad \alpha = \frac{t_i - t_{i-1}}{\tau}.$$

The value of ν depends on the method of interpolation chosen. nag_tsa_inhom_iema (g13mec) gives the option of three interpolation methods:

1. Previous point: $\nu = 1$;
2. Linear: $\nu = (1 - \mu)/\alpha$;
3. Next point: $\nu = \mu$.

The m-iterated exponential moving average, $\mathrm{EMA}[\tau, m; z](t_i)$, $m > 1$, is defined using the recursive formula:

$$\mathrm{EMA}[\tau, m; z] = \mathrm{EMA}[\tau; \mathrm{EMA}[\tau, m - 1; z]]$$

with

$$\mathrm{EMA}[\tau, 1; z] = \mathrm{EMA}[\tau; z].$$

For large datasets or where all the data is not available at the same time, z and t can be split into arbitrary sized blocks and nag_tsa_inhom_iema (g13mec) called multiple times.

4 References

Dacorogna M M, Gencay R, Müller U, Olsen R B and Pictet O V (2001) *An Introduction to High-frequency Finance* Academic Press

Zumbach G O and Müller U A (2001) Operators on inhomogeneous time series *International Journal of Theoretical and Applied Finance* **4(1)** 147–178

5 Arguments

1: **nb** – Integer *Input*

On entry: b, the number of observations in the current block of data. The size of the block of data supplied in **iema** and **t** can vary; therefore **nb** can change between calls to nag_tsa_inhom_iema (g13mec).

Constraint: **nb** ≥ 0.

2: **iema**[**nb**] – double *Input/Output*

On entry: z_i, the current block of observations, for $i = k+1, \ldots, k+b$, where k is the number of observations processed so far, i.e., the value supplied in **pn** on entry.

On exit: the iterated EMA, with **iema**$[i-1] = \mathrm{EMA}[\tau, m; z](t_i)$.

3: **t**[**nb**] – const double *Input*

On entry: t_i, the times for the current block of observations, for $i = k+1, \ldots, k+b$, where k is the number of observations processed so far, i.e., the value supplied in **pn** on entry.

If $t_i \leq t_{i-1}$, **fail.code** = NE_NOT_STRICTLY_INCREASING will be returned, but nag_tsa_inhom_iema (g13mec) will continue as if t was strictly increasing by using the absolute value.

4: **tau** – double *Input*

On entry: τ, the argument controlling the rate of decay, which must be sufficiently large that $e^{-\alpha}$, $\alpha = (t_i - t_{i-1})/\tau$ can be calculated without overflowing, for all i.

Constraint: **tau** > 0.0.

5: **m** – Integer *Input*

On entry: m, the number of times the EMA operator is to be iterated.

Constraint: **m** ≥ 1.

6: **sinit**[**m** + **2**] – const double *Input*

On entry: if **pn** $= 0$, the values used to start the iterative process, with

sinit$[0] = t_0$,

sinit$[1] = z_0$,

sinit$[j+1] = \mathrm{EMA}[\tau, j; z](t_0)$, for $j = 1, 2, \ldots,$ **m**.

If **pn** $\neq 0$, **sinit** is not referenced and may be **NULL**.

7: **inter**[**2**] – const Nag_TS_Interpolation *Input*

On entry: the type of interpolation used with **inter**[0] indicating the interpolation method to use when calculating $\mathrm{EMA}[\tau, 1; z]$ and **inter**[1] the interpolation method to use when calculating $\mathrm{EMA}[\tau, j; z]$, $j > 1$.

Three types of interpolation are possible:

inter$[i]$ = Nag_PreviousPoint
Previous point, with $\nu = 1$.

inter$[i]$ = Nag_Linear
Linear, with $\nu = (1 - \mu)/\alpha$.

inter$[i]$ = Nag_NextPoint
Next point, $\nu = \mu$.

Zumbach and Müller (2001) recommend that linear interpolation is used in second and subsequent iterations, i.e., **inter**$[1]$ = Nag_Linear, irrespective of the interpolation method used at the first iteration, i.e., the value of **inter**$[0]$.

Constraint: **inter**$[i - 1]$ = Nag_PreviousPoint, Nag_Linear or Nag_NextPoint, for $i = 1, 2$.

8: **pn** – Integer * *Input/Output*

On entry: k, the number of observations processed so far. On the first call to nag_tsa_inhom_iema (g13mec), or when starting to summarise a new dataset, **pn** must be set to 0. On subsequent calls it must be the same value as returned by the last call to nag_tsa_inhom_iema (g13mec).

On exit: $k + b$, the updated number of observations processed so far.

Constraint: **pn** ≥ 0.

9: **rcomm**[**m** + **20**] – double *Communication Array*

On entry: communication array, used to store information between calls to nag_tsa_inhom_iema (g13mec). If **rcomm** is **NULL** then **pn** must be set to zero and all the data must be supplied in one go.

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

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

NE_BAD_PARAM

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

NE_ILLEGAL_COMM

rcomm has been corrupted between calls.

NE_INT

On entry, **m** = ⟨*value*⟩.
Constraint: **m** ≥ 1.

On entry, **nb** = ⟨*value*⟩.
Constraint: **nb** ≥ 0.

On entry, **pn** = ⟨*value*⟩.
Constraint: **pn** ≥ 0.

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 the Essential Introduction 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 the Essential Introduction for further information.

NE_NOT_STRICTLY_INCREASING

On entry, $i = \langle value \rangle$, $\mathbf{t}[i-2] = \langle value \rangle$ and $\mathbf{t}[i-1] = \langle value \rangle$.
Constraint: **t** should be strictly increasing.

NE_PREV_CALL

If **pn** > 0 then **inter** must be unchanged since previous call.

On entry, **m** $= \langle value \rangle$.
On entry at previous call, **m** $= \langle value \rangle$.
Constraint: if **pn** > 0 then **m** must be unchanged since previous call.

On entry, **pn** $= \langle value \rangle$.
On exit from previous call, **pn** $= \langle value \rangle$.
Constraint: if **pn** > 0 then **pn** must be unchanged since previous call.

On entry, **tau** $= \langle value \rangle$.
On entry at previous call, **tau** $= \langle value \rangle$.
Constraint: if **pn** > 0 then **tau** must be unchanged since previous call.

NE_REAL

On entry, **tau** $= \langle value \rangle$.
Constraint: **tau** > 0.0.

NE_REAL_ARRAY

On entry, $i = \langle value \rangle$, $\mathbf{t}[i-2] = \langle value \rangle$ and $\mathbf{t}[i-1] = \langle value \rangle$.
Constraint: $\mathbf{t}[i-1] \neq \mathbf{t}[i-2]$ if linear interpolation is being used.

NW_OVERFLOW_WARN

Truncation occurred to avoid overflow, check for extreme values in **t**, **iema** or for **tau**. Results are returned using the truncated values.

7 Accuracy

Not applicable.

8 Parallelism and Performance

nag_tsa_inhom_iema (g13mec) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_tsa_inhom_iema (g13mec) 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

Approximately $4m$ real elements are internally allocated by nag_tsa_inhom_iema (g13mec).

The more data you supply to nag_tsa_inhom_iema (g13mec) in one call, i.e., the larger **nb** is, the more efficient the function will be.

Checks are made during the calculation of α to avoid overflow. If a potential overflow is detected the offending value is replaced with a large positive or negative value, as appropriate, and the calculations performed based on the replacement values. In such cases **fail.code** = NW_OVERFLOW_WARN is returned. This should not occur in standard usage and will only occur if extreme values of **iema**, **t** or **tau** are supplied.

10 Example

The example reads in a simulated time series, (t, z) and calculates the iterated exponential moving average.

10.1 Program Text

```
/* nag_tsa_inhom_iema (g13mec) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 24, 2013.
 */
/* Pre-processor includes */
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg13.h>

int main(void)
{
  /* Integer scalar and array declarations */
  Integer i, ierr, m, nb, pn;
  Integer exit_status = 0;

  /* NAG structures and types */
  NagError fail;
  Nag_TS_Interpolation inter[2];

  /* Double scalar and array declarations */
  double tau;
  double *iema = 0, *rcomm = 0, *sinit = 0, *t = 0;

  /* Character scalar and array declarations */
  char cinter[40];

  /* Initialise the error structure */
  INIT_FAIL(fail);

  printf("nag_tsa_inhom_iema (g13mec) Example Program Results\n\n");

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

  /* Read in the number of iterations required */
#ifdef _WIN32
  scanf_s("%"NAG_IFMT"%*[^\n] ",&m);
#else
  scanf("%"NAG_IFMT"%*[^\n] ",&m);
#endif
```

```
  /* Read in the interpolation method to use */
#ifdef _WIN32
  scanf_s("%39s",cinter, _countof(cinter));
#else
  scanf("%39s",cinter);
#endif
  inter[0] = (Nag_TS_Interpolation) nag_enum_name_to_value(cinter);
#ifdef _WIN32
  scanf_s("%39s",cinter, _countof(cinter));
#else
  scanf("%39s",cinter);
#endif
  inter[1] = (Nag_TS_Interpolation) nag_enum_name_to_value(cinter);

  /* Read in the decay parameter */
#ifdef _WIN32
  scanf_s("%lf%*[^\n] ", &tau);
#else
  scanf("%lf%*[^\n] ", &tau);
#endif

  /* Read in the initial values */
  if (!(sinit = NAG_ALLOC(m + 2, double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
  for (i = 0; i < m + 2; i++)
    {
#ifdef _WIN32
      scanf_s("%lf", &sinit[i]);
#else
      scanf("%lf", &sinit[i]);
#endif
    }
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

  /* Print some titles */
  printf("                         Iterated\n");
  printf("              Time          EMA\n");
  printf(" -------------------------------\n");

  if (!(rcomm = NAG_ALLOC(m + 20, double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* Loop over each block of data */
  for(pn = 0;;)
    {
      /* Read in the number of observations in this block */
#ifdef _WIN32
      ierr = scanf_s("%"NAG_IFMT, &nb);
#else
      ierr = scanf("%"NAG_IFMT, &nb);
#endif
      if (ierr == EOF || ierr < 1) break;
#ifdef _WIN32
      scanf_s("%*[^\n] ");
#else
      scanf("%*[^\n] ");
#endif

      /* Reallocate IEMA and T to the required size */
```

```
      NAG_FREE(iema);
      NAG_FREE(t);
      if (!(iema = NAG_ALLOC(nb, double)) ||
          !(t = NAG_ALLOC(nb, double)))
        {
          printf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }

      /* Read in the data for this block */
      for (i = 0; i < nb; i++)
        {
#ifdef _WIN32
          scanf_s("%lf%lf",&t[i], &iema[i]);
#else
          scanf("%lf%lf",&t[i], &iema[i]);
#endif
        }
#ifdef _WIN32
      scanf_s("%*[^\n] ");
#else
      scanf("%*[^\n] ");
#endif

      /* Call nag_tsa_inhom_iema (g13mec) to update the iterated EMA for
         this block of data. The routine overwrites the input data with
         the iterated EMA */
      nag_tsa_inhom_iema(nb,iema,t,tau,m,sinit,inter,&pn,rcomm,&fail);
      if (fail.code != NE_NOERROR)
        {
          printf("Error from nag_tsa_inhom_iema (g13mec).\n%s\n",
                 fail.message);
          exit_status = -1;
          goto END;
        }

      /* Display the results for this block of data */
      for (i = 0; i < nb; i++)
        {
          printf(" %3"NAG_IFMT"    %10.1f    %10.3f\n",pn-nb+i+1,t[i],
                 iema[i]);
        }
      printf("\n");
    }

 END:
  NAG_FREE(iema);
  NAG_FREE(t);
  NAG_FREE(sinit);
  NAG_FREE(rcomm);

  return(exit_status);
}
```

10.2 Program Data

```
nag_tsa_inhom_iema (g13mec) Example Program Data
2                                         :: m
Nag_NextPoint Nag_Linear  2.0             :: inter[0:1],tau
5.0 0.5 0.5 0.5                           :: sinit

5                                         :: nb
 7.5  0.6
 8.2  0.6
18.1  0.8
22.8  0.1
25.8  0.2                                 :: End of t and iema for first block

10                                        :: nb
```

```
26.8  0.2
31.1  0.5
38.4  0.7
45.9  0.1
48.2  0.4
48.9  0.7
57.9  0.8
58.5  0.3
63.9  0.2
65.2  0.5                                :: End of t and iema for second block

15                                       :: nb
66.6  0.2
67.4  0.3
69.3  0.8
69.9  0.6
73.0  0.1
75.6  0.7
77.0  0.9
84.7  0.6
86.8  0.3
88.0  0.1
88.5  0.1
91.0  0.4
93.0  1.0
93.7  1.0
94.0  0.1                                :: End of t and iema for third block
```

10.3 Program Results

```
nag_tsa_inhom_iema (g13mec) Example Program Results

                     Iterated
           Time         EMA
 --------------------------------
   1          7.5        0.531
   2          8.2        0.544
   3         18.1        0.754
   4         22.8        0.406
   5         25.8        0.232

   6         26.8        0.217
   7         31.1        0.357
   8         38.4        0.630
   9         45.9        0.263
  10         48.2        0.241
  11         48.9        0.279
  12         57.9        0.713
  13         58.5        0.717
  14         63.9        0.385
  15         65.2        0.346

  16         66.6        0.330
  17         67.4        0.315
  18         69.3        0.409
  19         69.9        0.459
  20         73.0        0.377
  21         75.6        0.411
  22         77.0        0.536
  23         84.7        0.632
  24         86.8        0.538
  25         88.0        0.444
  26         88.5        0.401
  27         91.0        0.331
  28         93.0        0.495
  29         93.7        0.585
  30         94.0        0.612
```

This example plot shows the exponential moving average for the same data using three different values of τ and illustrates the effect on the EMA of altering this argument.