NAG Library Function Document

nag_rand_egarch (g05pgc)

1 Purpose

nag_rand_egarch (g05pgc) generates a given number of terms of an exponential GARCH($p,q$) process (see Engle and Ng (1993)).

2 Specification

```c
#include <nag.h>
#include <nagg05.h>

void nag_rand_egarch (Nag_ErrorDistn dist, Integer num, Integer ip,
                      Integer iq, const double theta[], Integer df, double ht[], double et[],
                      Nag_Boolean fcall, double r[], Integer lr, Integer state[],
                      NagError *fail)
```

3 Description

An exponential GARCH($p,q$) process is represented by:

$$
\ln(h_t) = \alpha_0 + \sum_{i=1}^{q} \alpha_i z_{t-i} + \sum_{i=1}^{q} \phi_i (|z_{t-i}| - E[|z_{t-i}|]) + \sum_{j=1}^{p} \beta_j \ln(h_{t-j}), \quad t = 1, 2, \ldots, T;
$$

where $z_t = \frac{\epsilon_t}{\sqrt{h_t}}$, $E[|z_{t-i}|]$ denotes the expected value of $|z_{t-i}|$, and $\epsilon_t | \psi_{t-1} = N(0, h_t)$ or $\epsilon_t | \psi_{t-1} = S_t(df, h_t)$. Here $S_t$ is a standardized Student’s $t$-distribution with $df$ degrees of freedom and variance $h_t$, $T$ is the number of observations in the sequence, $\epsilon_t$ is the observed value of the GARCH($p,q$) process at time $t$, $h_t$ is the conditional variance at time $t$, and $\psi_t$ the set of all information up to time $t$.

One of the initialization functions nag_rand_init_repeatable (g05kfc) (for a repeatable sequence if computed sequentially) or nag_rand_init_nonrepeatable (g05kgc) (for a non-repeatable sequence) must be called prior to the first call to nag_rand_egarch (g05pgc).

4 References


5 Arguments

1: dist – Nag_ErrorDistn

On entry: the type of distribution to use for \( \epsilon_t \).

\[ \text{dist} = \text{Nag_NormalDistn} \]
A Normal distribution is used.

\[ \text{dist} = \text{Nag_Tdistn} \]
A Student’s \( t \)-distribution is used.

Constraint: \( \text{dist} = \text{Nag_NormalDistn} \) or \( \text{Nag_Tdistn} \).

2: num – Integer

On entry: \( T \), the number of terms in the sequence.

Constraint: \( \text{num} \geq 0 \).

3: ip – Integer

On entry: the number of coefficients, \( \beta_i \), for \( i = 1, 2, \ldots, p \).

Constraint: \( \text{ip} \geq 0 \).

4: iq – Integer

On entry: the number of coefficients, \( \alpha_i \), for \( i = 1, 2, \ldots, q \).

Constraint: \( \text{iq} \geq 1 \).

5: \( \theta \{ 2 \times \text{iq} + \text{ip} + 1 \} \) – const double

On entry: the initial parameter estimates for the vector \( \theta \). The first element must contain the coefficient \( \alpha_0 \) and the next \( \text{iq} \) elements must contain the autoregressive coefficients \( \alpha_i \), for \( i = 1, 2, \ldots, q \). The next \( \text{iq} \) elements must contain the coefficients \( \phi_i \), for \( i = 1, 2, \ldots, q \). The next \( \text{ip} \) elements must contain the moving average coefficients \( \beta_j \), for \( j = 1, 2, \ldots, p \).

Constraints:

\[ \sum_{i=1}^{p} \beta_i \neq 1.0; \]
\[ \frac{\alpha_0}{1 - \sum_{i=1}^{p} \beta_i} \leq -\log \left( \text{nag_real_safe_small_number} \right). \]

6: df – Integer

On entry: the number of degrees of freedom for the Student’s \( t \)-distribution.

If \( \text{dist} = \text{Nag_NormalDistn} \), \( \text{df} \) is not referenced.

Constraint: if \( \text{dist} = \text{Nag_Tdistn} \), \( \text{df} > 2 \).

7: \( h \{ \text{num} \} \) – double

On exit: the conditional variances \( h_t \), for \( t = 1, 2, \ldots, T \), for the GARCH\((p,q)\) sequence.

8: \( e \{ \text{num} \} \) – double

On exit: the observations \( \epsilon_t \), for \( t = 1, 2, \ldots, T \), for the GARCH\((p,q)\) sequence.

9: fcall – Nag_Boolean

On entry: if \( \text{fcall} = \text{Nag_TRUE} \), a new sequence is to be generated, otherwise a given sequence is to be continued using the information in \( r \).
On entry: the array contains information required to continue a sequence if fc\text{call} = \text{Nag\_FALSE}.

On exit: contains information that can be used in a subsequent call of nag\_rand\_egarch (g05pgc), with fc\text{call} = \text{Nag\_FALSE}.

\textbf{l\text{r}} – Integer

\textit{input}

On entry: the dimension of the array \textit{r}.

\textit{Constraint:} \textit{l\text{r}} \geq 2 \times (\textit{ip} + 2 \times \textit{iq} + 2).

\textbf{state}[\text{dim}] – Integer

\textit{communication array}

\textbf{Note:} the dimension, \textit{dim}, of this array is dictated by the requirements of associated functions that must have been previously called. This array MUST be the same array passed as argument \textit{state} in the previous call to nag\_rand\_init\_repeatable (g05kfc) or nag\_rand\_init\_nonrepeatable (g05kgc).

On entry: contains information on the selected base generator and its current state.

On exit: contains updated information on the state of the generator.

\textbf{fail} – NagError*

\textit{input/output}

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

6 Error Indicators and Warnings

\textbf{NE\_ALLOC\_FAIL}

Dynamic memory allocation failed.

See Section 3.2.1.2 in the Essential Introduction for further information.

\textbf{NE\_BAD\_PARAM}

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

\textbf{NE\_INT}

On entry, \textit{df} = \textit{\langle value\rangle}.

Constraint: \textit{df} \geq 3.

On entry, \textit{ip} = \textit{\langle value\rangle}.

Constraint: \textit{ip} \geq 0.

On entry, \textit{iq} = \textit{\langle value\rangle}.

Constraint: \textit{iq} \geq 1.

On entry, \textit{l\text{r}} is not large enough, \textit{l\text{r}} = \textit{\langle value\rangle}: minimum length required = \textit{\langle value\rangle}.

On entry, \textit{num} = \textit{\langle value\rangle}.

Constraint: \textit{num} \geq 0.

\textbf{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.

\textbf{NE\_INVALID\_STATE}

On entry, \textit{state} vector has been corrupted or not initialized.
7 Accuracy

Not applicable.

8 Parallelism and Performance

nag_rand_egarch (g05pgc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

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

None.

10 Example

This example first calls nag_rand_init_repeatable (g05kfc) to initialize a base generator then calls nag_rand_egarch (g05pgc) to generate two realizations, each consisting of ten observations, from an exponential GARCH(1,1) model.

10.1 Program Text

/* nag_rand_egarch (g05pgc) Example Program. *
 * Copyright 2014 Numerical Algorithms Group. *
 * Mark 9, 2009. *
 */
/* Pre-processor includes */
#include <stdio.h>
#include <math.h>
#include <nag.h>
#include <nag.h>
#include <nag_stdtlib.h>
#include <nag05.h>

int main(void)
{
    /* Integer scalar and array declarations */
    Integer exit_status = 0;
    Integer lr, i, lstate;
    Integer *state = 0;

    /* NAG structures */
    NagError fail;
    Nag_Boolean fcall;
/* Double scalar and array declarations */
double  *et = 0, *ht = 0, *r = 0;

/* Number of terms to generate */
Integer  num = 10;

/* Normally distributed errors */
Nag_ErrorDistn  dist = Nag_NormalDistn;
Integer  df = 0;

/* Set up the parameters for the series being generated */
Integer  ip = 1;
Integer  iq = 1;
double  theta[] = { 0.1e0, -0.3e0, 0.1e0, 0.9e0 };

/* Choose the base generator */
Nag_BaseRNG  genid = Nag_Basic;
Integer  subid = 0;

/* Set the seed */
Integer  seed[] = { 1762543 };
Integer  lseed = 1;

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

/* Get the length of the state array */
lstate = -1;
nag_rand_init_repeatable(genid, subid, seed, lseed, state, &lstate, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_rand_init_repeatable (g05kfc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Calculate the size of the reference vector */
lr = 2*(2*iq+ip+2);

/* Allocate arrays */
if (! et = NAG_ALLOC(num, double)) ||
    !(ht = NAG_ALLOC(num, double)) ||
    !(r = NAG_ALLOC(lr, double)) ||
    !(state = NAG_ALLOC(lstate, Integer)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

/* Initialise the generator to a repeatable sequence */
nag_rand_init_repeatable(genid, subid, seed, lseed, state, &lstate, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_rand_init_repeatable (g05kfc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Generate the first realization */
fcall = Nag_TRUE;
nag_rand_egarch(dist, num, ip, iq, theta, df, ht, et, fcall, r, lr, state, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_rand_egarch (g05pgc).\n%s\n", fail.message);
}
exit_status = 1;
goto END;
}

/* Display the results */
printf(" Realization Number 1\n");
printf(" I HT(I) ET(I)\n");
for (i = 0; i < num; i++)
    printf(" %5f %16.4f %16.4f\n", i+1, ht[i], et[i]);

END:
NAG_FREE(et);
NAG_FREE(ht);
NAG_FREE(r);
NAG_FREE(state);
return exit_status;

10.2 Program Data

None.

10.3 Program Results

nag_rand_egarch (g05pgc) Example Program Results

<p>| Realization Number 1 |     |     |
|----------------------|-----|--|---|</p>
<table>
<thead>
<tr>
<th>I</th>
<th>HT(I)</th>
<th>ET(I)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.5098</td>
<td>0.5526</td>
</tr>
<tr>
<td>2</td>
<td>2.1785</td>
<td>-1.8383</td>
</tr>
<tr>
<td>3</td>
<td>3.3844</td>
<td>1.2180</td>
</tr>
<tr>
<td>4</td>
<td>2.6780</td>
<td>1.3672</td>
</tr>
<tr>
<td>5</td>
<td>2.0953</td>
<td>-1.8178</td>
</tr>
<tr>
<td>6</td>
<td>3.2813</td>
<td>-0.0343</td>
</tr>
<tr>
<td>7</td>
<td>2.9958</td>
<td>-0.5094</td>
</tr>
<tr>
<td>8</td>
<td>3.0815</td>
<td>1.3978</td>
</tr>
<tr>
<td>9</td>
<td>2.3961</td>
<td>0.0070</td>
</tr>
<tr>
<td>10</td>
<td>2.2445</td>
<td>0.6661</td>
</tr>
</tbody>
</table>

<p>| Realization Number 2 |     |     |
|----------------------|-----|--|---|</p>
<table>
<thead>
<tr>
<th>I</th>
<th>HT(I)</th>
<th>ET(I)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.9327</td>
<td>-2.2795</td>
</tr>
<tr>
<td>2</td>
<td>3.5577</td>
<td>-1.2249</td>
</tr>
<tr>
<td>3</td>
<td>4.1461</td>
<td>0.6424</td>
</tr>
<tr>
<td>4</td>
<td>3.4455</td>
<td>-2.9920</td>
</tr>
<tr>
<td>5</td>
<td>5.9199</td>
<td>0.5777</td>
</tr>
<tr>
<td>---</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>6</td>
<td>4.8221</td>
<td>-1.2894</td>
</tr>
<tr>
<td>7</td>
<td>5.3174</td>
<td>-1.6473</td>
</tr>
<tr>
<td>8</td>
<td>6.1095</td>
<td>6.1689</td>
</tr>
<tr>
<td>9</td>
<td>3.1579</td>
<td>2.2935</td>
</tr>
<tr>
<td>10</td>
<td>2.2189</td>
<td>0.1141</td>
</tr>
</tbody>
</table>