NAG Library Function Document

nag_estimate_garchGJR (g13fec)

1 Purpose

nag_estimate_garchGJR (g13fec) estimates the arguments of a univariate regression-GJR GARCH(p,q) process (see Glosten et al. (1993)).

2 Specification

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

void nag_estimate_garchGJR (const double yt[], const double x[], Integer tdx,
                           Integer num, Integer p, Integer q, Integer nreg, Integer mn,
                           double theta[], double se[], double sc[], double covar[], Integer tdc,
                           double *hp, double et[], double ht[], double *lgf,
                           Nag_Garch_Stationary_Type stat_opt, Nag_Garch_Est_Initial_Type est_opt,
                           Integer max_iter, double tol, NagError *fail)
```

3 Description

A univariate regression-GJR GARCH(p,q) process, with p coefficients \( \alpha_i \), for \( i = 1,2,\ldots,p \), q coefficients, \( \beta_i \), for \( i = 1,2,\ldots,q \), mean \( b_0 \), and k linear regression coefficients \( b_i \), for \( i = 1,2,\ldots,k \), can be represented by:

\[
y_t = b_0 + x_t^T b + \epsilon_t
\]

\( \epsilon_t | \psi_{t-1} \sim N(0,h_t) \)

\[
h_t = \alpha_0 + \sum_{i=1}^{q}(\alpha_i + \gamma S_t)\epsilon_{t-i}^2 + \sum_{i=1}^{p}\beta_i h_{t-i}, \quad t = 1,\ldots,T.
\]

where \( S_t = 1 \), if \( \epsilon_t < 0 \), and \( S_t = 0 \), if \( \epsilon_t \geq 0 \). Here \( T \) is the number of terms in the sequence, \( y_t \) denotes the endogenous variables, \( x_t \) the exogenous variables, \( b_0 \) the mean, \( b \) the regression coefficients, \( \epsilon_t \) the residuals, \( \gamma \) is the asymmetry parameter, \( h_t \) is the conditional variance, and \( \psi_t \) the information set of all information up to time \( t \).

nag_estimate_garchGJR (g13fec) provides an estimate for \( \hat{\theta} \), the \( (p + q + k + 3) \times 1 \) parameter vector \( \theta = (b_0, b^T, \omega^T) \) where \( \omega^T = (\alpha_0, \alpha_1,\ldots,\alpha_q, \beta_1,\ldots,\beta_p, \gamma) \) and \( b^T = (b_1,\ldots,b_k) \).

\( \text{mn, nreg} \) can be used to simplify the GARCH(p,q) expression in equation (1) as follows:

No Regression or Mean

\[
y_t = \epsilon_t,
\]

\( \text{mn} = 0, \)

\( \text{nreg} = 0, \) and

\( \theta \) is a \( (p + q + 2) \times 1 \) vector.

No Regression

\[
y_t = b_0 + \epsilon_t,
\]

\( \text{mn} = 1, \)
\[ n_{\text{reg}} = 0, \text{ and} \]
\[ \theta \text{ is a } (p + q + 3) \times 1 \text{ vector.} \]

**Note:** if the \( y_t = \mu + \epsilon_t \), where \( \mu \) is known (not to be estimated by nag_estimate_garchGJR (g13fec)) then equation (1) can be written as \( y_t^\mu = \epsilon_t \), where \( y_t^\mu = y_t - \mu \). This corresponds to the case **No Regression or Mean**, with \( y_t \) replaced by \( y_t - \mu \).

**No Mean**

\[ y_t = x_T^T b + \epsilon_t, \]
\[ mn = 0, \]
\[ n_{\text{reg}} = k \text{ and} \]
\[ \theta \text{ is a } (p + q + k + 2) \times 1 \text{ vector.} \]

## 4 References


## 5 Arguments

**Note:** for convenience \( npar \) will be used here to denote the expression \( 2 + q + p + mn + n_{\text{reg}} \) representing the number of model parameters.

1. \( \text{y}[\text{num}] \) – const double
   
   *Input*
   
   On entry: the sequence of observations, \( y_t \), for \( t = 1, 2, \ldots, T \).

2. \( \text{x}[\text{num} \times \text{tdx}] \) – const double
   
   *Input*
   
   **Note:** the \( i \)th element of the \( j \)th vector \( X \) is stored in \( x[(i - 1) \times \text{tdx} + j - 1] \).
   
   On entry: row \( t \) of \( x \) must contain the time dependent exogenous vector \( x_t \), where \( x_T^T = (x_t^1, \ldots, x_t^k) \), for \( t = 1, 2, \ldots, T \).

3. \( \text{tdx} \) – Integer
   
   *Input*
   
   On entry: the stride separating matrix column elements in the array \( x \).
   
   **Constraint:** \( \text{tdx} \geq n_{\text{reg}} \).

4. \( \text{num} \) – Integer
   
   *Input*
   
   On entry: the number of terms in the sequence, \( T \).
   
   **Constraint:** \( \text{num} \geq npar \).

5. \( p \) – Integer
   
   *Input*
   
   On entry: the GARCH\((p,q)\) parameter \( p \).
   
   **Constraint:** \( p \geq 0 \).
6: \( q \) – Integer  
   \textit{Input}
   \textit{On entry}: the GARCH(\( p, q \)) parameter \( q \).
   \textit{Constraint}: \( q \geq 1 \).

7: \( \text{nreg} \) – Integer  
   \textit{Input}
   \textit{On entry}: \( k \), the number of regression coefficients.
   \textit{Constraint}: \( \text{nreg} \geq 0 \).

8: \( \text{mn} \) – Integer  
   \textit{Input}
   \textit{On entry}: if \( \text{mn} = 1 \), the mean term \( b_0 \) will be included in the model.
   \textit{Constraint}: \( \text{mn} = 0 \) or \( 1 \).

9: \( \text{theta}[\text{npar}] \) – double  
   \textit{Input/Output}
   \textit{On entry}: the initial parameter estimates for the vector \( \theta \).
   The first element contains the coefficient \( \alpha_o \), the next \( q \) elements contain the autoregressive coefficients \( \alpha_i \), for \( i = 1, 2, \ldots, q \).
   The next \( p \) elements are the moving average coefficients \( \beta_j \), for \( j = 1, 2, \ldots, p \).
   The next element contains the asymmetry parameter \( \gamma \).
   If \( \text{estopt} = \text{Nag_Garch_EstInitial} \), (when \( \text{mn} = 1 \)) the next term contains an initial estimate of the mean term \( b_0 \), and the remaining \( \text{nreg} \) elements are taken as initial estimates of the linear regression coefficients \( b_i \), for \( i = 1, 2, \ldots, k \).
   \textit{On exit}: the estimated values \( \hat{\theta} \) for the vector \( \theta \).
   The first element contains the coefficient \( \alpha_o \), the next \( q \) elements contain the coefficients \( \alpha_i \), for \( i = 1, 2, \ldots, q \).
   The next \( p \) elements are the coefficients \( \beta_j \), for \( j = 1, 2, \ldots, p \).
   The next element contains the estimate for the asymmetry parameter \( \gamma \).
   If \( \text{mn} = 1 \), the next element contains an estimate for the mean term \( b_0 \).
   The final \( \text{nreg} \) elements are the estimated linear regression coefficients \( b_i \), for \( i = 1, 2, \ldots, k \).

10: \( \text{se}[\text{npar}] \) – double  
    \textit{Output}
    \textit{On exit}: the standard errors for \( \hat{\theta} \).
    The first element contains the standard error for \( \alpha_o \).
    The next \( q \) elements contain the standard errors for \( \alpha_i \), for \( i = 1, 2, \ldots, q \).
    The next \( p \) elements are the standard errors for \( \beta_j \), for \( j = 1, 2, \ldots, p \).
    The next element contains the standard error for \( \gamma \).
    If \( \text{mn} = 1 \), the next element contains the standard error for \( b_0 \).
    The final \( \text{nreg} \) elements are the standard errors for \( b_j \), for \( j = 1, 2, \ldots, k \).

11: \( \text{sc}[\text{npar}] \) – double  
    \textit{Output}
    \textit{On exit}: the scores for \( \hat{\theta} \).
    The first element contains the score for \( \alpha_o \), the next \( q \) elements contain the score for \( \alpha_i \), for \( i = 1, 2, \ldots, q \).
    The next \( p \) elements are the scores for \( \beta_j \), for \( j = 1, 2, \ldots, p \).
The next element contains the score for $\gamma$.

If $mn = 1$, the next element contains the score for $b_0$.

The final $nreg$ elements are the scores for $b_j$, for $j = 1, 2, \ldots, k$.

12: $\text{covar}[npar \times \text{tdc}]$ – double

*Output*

*Note:* the $(i, j)$th element of the matrix is stored in $\text{covar}[(i - 1) \times \text{tdc} + j - 1]$.

*On exit:* the covariance matrix of the parameter estimates $\hat{\theta}$, that is the inverse of the Fisher Information Matrix.

13: $\text{tdc}$ – Integer

*Input*

*On entry:* the stride separating matrix column elements in the array $\text{covar}$.

*Constraint:* $\text{tdc} \geq npar$.

14: $\text{hp}$ – double *

*Input/Output*

*On entry:* if $\text{est_opt} = \text{Nag Garch Est Initial False}$, $\text{hp}$ is the value to be used for the pre-observed conditional variance.

If $\text{est_opt} = \text{Nag Garch Est Initial True}$, $\text{hp}$ is not referenced.

*On exit:* if $\text{est_opt} = \text{Nag Garch Est Initial True}$, $\text{hp}$ is the estimated value of the pre-observed of the conditional variance.

15: $\text{et}[\text{num}]$ – double

*Output*

*On exit:* the estimated residuals, $\epsilon_t$, for $t = 1, 2, \ldots, T$.

16: $\text{ht}[\text{num}]$ – double

*Output*

*On exit:* the estimated conditional variances, $h_t$, for $t = 1, 2, \ldots, T$.

17: $\text{lgf}$ – double *

*Output*

*On exit:* the value of the log likelihood function at $\hat{\theta}$.

18: $\text{stat_opt}$ – $\text{Nag Garch Stationary Type}$

*Input*

*On entry:* if $\text{stat_opt} = \text{Nag Garch Stationary True}$, Stationary conditions are enforced.

If $\text{stat_opt} = \text{Nag Garch Stationary False}$, Stationary conditions are not enforced.

*Constraint:* $\text{stat_opt} = \text{Nag Garch Stationary True}$ or $\text{Nag Garch Stationary False}$.

19: $\text{est_opt}$ – $\text{Nag Garch Est Initial Type}$

*Input*

*On entry:* if $\text{est_opt} = \text{Nag Garch Est Initial True}$, the function provides initial parameter estimates of the regression terms $(b_0, b^T)$.

If $\text{est_opt} = \text{Nag Garch Est Initial False}$, you must supply the initial estimations of the regression parameters $(b_0, b^T)$.

*Constraint:* $\text{est_opt} = \text{Nag Garch Est Initial True}$ or $\text{Nag Garch Est Initial False}$.

20: $\text{max_iter}$ – Integer

*Input*

*On entry:* the maximum number of iterations to be used by the optimization function when estimating the $\text{GARCH}(p, q)$ parameters. If $\text{max_iter}$ is set to 0, the standard errors, score vector and variance-covariance are calculated for the input value of $\theta$ in $\text{theta}$; however the value of $\theta$ is not updated.

*Constraint:* $\text{max_iter} \geq 0$.  

---
21:  tol – double

   On entry: the tolerance to be used by the optimization function when estimating the GARCH(p,q)
   parameters.

22:  fail – NagError *

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

6  Error Indicators and Warnings

NE_2_INT_ARG_LT

   On entry, num = ⟨value⟩ while 2 + q + p + mn + nreg = ⟨value⟩. These parameters must satisfy
   num ≥ 2 + q + p + mn + nreg.

   On entry, tdc = ⟨value⟩ while 2 + q + p + mn + nreg = ⟨value⟩. These parameters must satisfy
   tdc ≥ 2 + q + p + mn + nreg.

   On entry, tdx = ⟨value⟩ while nreg = ⟨value⟩. These parameters must satisfy tdx ≥ nreg.

NE_ALLOC_FAIL

   Dynamic memory allocation failed.

NE_BAD_PARAM

   On entry, parameter est_opt had an illegal value.

   On entry, parameter stat_opt had an illegal value.

NE_INT_ARG_LT

   On entry, max_iter must not be less than 0: max_iter = ⟨value⟩.

   On entry, nreg = ⟨value⟩.

   Constraint: nreg ≥ 0.

   On entry, p = ⟨value⟩.

   Constraint: p ≥ 0.

   On entry, q = ⟨value⟩.

   Constraint: q ≥ 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.

NE_INVALID_INT_RANGE_2

   Value ⟨value⟩ given to mn is not valid. Correct range is 0 to 1.

NE_MAT_NOT_FULL_RANK

   Matrix X does not give a model of full rank.

NE_MAT_NOT_POS_DEF

   Attempt to invert the second derivative matrix needed in the calculation of the covariance matrix
   of the parameter estimates has failed. The matrix is not positive definite, possibly due to rounding
   errors.

7  Accuracy

   Not applicable.
8 Parallelism and Performance

Not applicable.

9 Further Comments

None.

10 Example

This example program illustrates the use of nag_estimate_garchGJR (g13fec) to model a GARCH(1,1) sequence generated by nag_rand_garchGJR (g05pfc), a six step forecast is then calculated using nag_forecast_garchGJR (g13ffc).

10.1 Program Text

```c
/* nag_estimate_garchGJR (g13fec) Example Program. *
 * Copyright 2014 Numerical Algorithms Group.
 * NAG C Library *
 * Mark 6, 2000. *
 */
#include <nag.h>
#include <nag_stdlib.h>
#include <stdio.h>
#include <ctype.h>
#include <math.h>
#include <nagg05.h>
#include <nagg13.h>
#define X(I, J) x[(I) *tdx + (J)]

int main(void)
{
    /* Integer scalar and array declarations */
    Integer exit_status = 0;
    Integer i, j, k, npar, tdc, tdx, lr, lstate;
    Integer *state = 0;
    /* NAG structures and data types */
    NagError fail;
    Nag_Boolean fcall;
    /* Double scalar and array declarations */
    double fac1, hp, lgf, xterm;
    double *covar = 0, *cvar = 0, *etm = 0, *ht = 0;
    double *htm = 0, *r = 0, *sc = 0, *se = 0, *theta = 0;
    double *x = 0, *yt = 0;
    /* Choose the base generator */
    Nag_BaseRNG genid = Nag_Basic;
    Integer subid = 0;
    /* Set the seed */
    Integer seed[] = { 1762543 }; 
    Integer lseed = 1;
    /* Set parameters for the (randomly generated) time series ... */
    /* Generate data assuming normally distributed errors */
    Nag_ErrorDistn dist = Nag_NormalDistn;
    double df = 0;
    /* Size of the time series */
    Integer num = 1000;
```

/* MA and AR parameters */
Integer ip = 1;
Integer iq = 1;
double param[] = { 0.4, 0.1, 0.7 };

/* Asymmetry parameter */
double gamma = 0.1;

/* Regression parameters */
Integer nreg = 2;
double mean = 4.0;
double bx[] = { 1.5, 2.5 };

/* When fitting a model to the time series ... */
/* Include mean in the model */
Integer mn = 1;

/* Use the following maximum number of iterations and tolerance */
Integer maxit = 50;
double tol = 1e-12;

/* Enforce stationary conditions */
Nag_Garch_Stationary_Type stat_opt = Nag_Garch_Stationary_True;

/* Estimate initial values for regression parameters */
Nag_Garch_Est_Initial_Type est_opt = Nag_Garch_Est_Initial_True;

/* Set the number of values to forecast from the fitted model */
Integer nt = 6;

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

printf("nag_estimate_garchGJR (g13fec) Example Program Results \n\n\n");

/* 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;
}

/* Derive various amounts */
npar = iq + ip + 1;
tdx = nreg;
tdc = npar + mn + nreg + 1;

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

if (!((cvar = NAG_ALLOC((npar + mn + nreg + 1) * tdc, double))
    || ((etm = NAG_ALLOC(num, double))
    || (ht = NAG_ALLOC(num, double))
    || (htm = NAG_ALLOC(num, double))
    || (r = NAG_ALLOC(lr, double))
    || (state = NAG_ALLOC(lstate, Integer))
    || (sc = NAG_ALLOC(npar + mn + nreg + 1, double))
    || (se = NAG_ALLOC(npar + mn + nreg + 1, double))
    || (theta = NAG_ALLOC(npar + mn + nreg + 1, double))
    || (cvar = NAG_ALLOC(nt, double))
    || (x = NAG_ALLOC(num * tdx, double))
    || (yt = NAG_ALLOC(num, double)))
{
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;
}

/* Set up the time dependent exogenous matrix x */
for (i = 0; i < num; ++i)
{
    fac1 = (double)(i + 1) * 0.01;
    X(i, 1) = sin(fac1) * 0.7 + 0.01;
    X(i, 0) = fac1 * 0.1 + 0.5;
}

/* Generate a realization of a random GARCH GJR time series and discard it */
fcall = Nag_TRUE;
nag_rand_garchGJR(dist, num, ip, iq, param, gamma, df, ht, yt, fcall, r, lr, 
state, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_rand_garchGJR (g05pfc).\n%s\n", fail.message);
    exit_status = 1;
goto END;
}

/* Generate a realization of a random GARCH GJR time series to use */
fcall = Nag_FALSE;
nag_rand_garchGJR(dist, num, ip, iq, param, gamma, df, ht, yt, fcall, r, lr, 
state, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_rand_garchGJR (g05pfc).\n%s\n", fail.message);
    exit_status = 1;
goto END;
}

/* Adjust the randomly generated time series to take into account for the 
exogenous matrix x */
for (i = 0; i < num; ++i)
{
    xterm = 0.0;
    for (k = 0; k < nreg; ++k)
        xterm += X(i, k) * bx[k];
    if (mn == 1)
        yt[i] = mean + xterm + yt[i];
    else
        yt[i] = xterm + yt[i];
}

/* Set initial estimates for the parameters */
for (i = 0; i < npar; ++i)
    theta[i] = param[i] * 0.5;
theta[npar] = gamma * 0.5;
if (mn == 1)
    theta[npar + 1] = mean * 0.5;
for (i = 0; i < nreg; ++i)
    theta[npar + 1 + mn + i] = bx[i] * 0.5;

/* nag_estimate_garchGJR (g13fec). */
Univariate time series, parameter estimation for an asymmetric Glosten, Jagannathan and Runkle (GJR) GARCH process

```c
nag_estimate_garchGJR(yt, x, tdx, num, ip, iq, nreg, mn,
theta, se, sc, covar, tdc, &hp,
etm, htm, &lgf, stat_opt, est_opt, maxit,
tol, &fail);
if (fail.code != NE_NOERROR)
{
  printf("Error from nag_estimate_garchGJR (g13fec).\n\n%s\n", fail.message);
  exit_status = 1;
goto END;
}

/* Display the results */
printf(" Parameter estimates Standard errors Correct values\n";
for (j = 0; j < npar; ++j)
  printf("%20.4f (%6.4f) %20.4f\n", theta[j], se[j],
param[j]);
printf("%20.4f (%6.4f) %20.4f\n", theta[npar], se[npar],
gamma);
if (mn)
  printf("%20.4f (%6.4f) %20.4f\n", theta[npar + 1],
  se[npar + 1], mean);
for (j = 0; j < nreg; ++j)
  printf("%20.4f (%6.4f) %20.4f\n", theta[npar + 1 + mn + j],
    se[npar + 1 + mn + j], bx[j]);

/* Now forecast nt steps ahead */
gamma = theta[npar];

/* nag_forecast_garchGJR (g13ffc). */
univariate time series, forecast function for an asymmetric Glosten, Jagannathan and Runkle (GJR) GARCH process

```nag_forecast_garchGJR(num, nt, ip, iq, theta, gamma, cvar, htm, etm, &fail);
printf("%"NAG_IFMT" step forecast = %8.4f\n", nt, cvar[nt-1]);
END:
NAG_FREE(covar);
NAG_FREE(ect);
NAG_FREE(ette);
NAG_FREE(htm);
NAG_FREE(sc);
NAG_FREE(se);
NAG_FREE(theta);
NAG_FREE(cvar);
NAG_FREE(x);
NAG_FREE(yt);
NAG_FREE(state);
return exit_status;

10.2 Program Data
None.
### 10.3 Program Results

*nag_estimate_garchGJR (g13fec) Example Program Results*

<table>
<thead>
<tr>
<th>Parameter estimates</th>
<th>Standard errors</th>
<th>Correct values</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3852</td>
<td>(0.1074)</td>
<td>0.4000</td>
</tr>
<tr>
<td>0.0603</td>
<td>(0.0280)</td>
<td>0.1000</td>
</tr>
<tr>
<td>0.7207</td>
<td>(0.0568)</td>
<td>0.7000</td>
</tr>
<tr>
<td>0.1674</td>
<td>(0.0495)</td>
<td>0.1000</td>
</tr>
<tr>
<td>4.0146</td>
<td>(0.1709)</td>
<td>4.0000</td>
</tr>
<tr>
<td>1.4593</td>
<td>(0.1613)</td>
<td>1.5000</td>
</tr>
<tr>
<td>2.4538</td>
<td>(0.1006)</td>
<td>2.5000</td>
</tr>
</tbody>
</table>

6 step forecast = 1.7344