NAG C Library Function Document

nag_estimate_garchGJR (g13fec)

1 Purpose

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

2 Specification

```c
#include <nag.h>
#include <nagl3.h>
void nag_estimate_garchGJR (const double y[], 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 $\alpha_i$, $i = 1, \ldots, p$, q coefficients, $\beta_i$, $i = 1, \ldots, q$, mean $b_0$, and k linear regression coefficients $b_i$, $i = 1, \ldots, k$, can be represented by:

$$y_t = b_0 + x_t^T b + \epsilon_t$$

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

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

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 $t$.

The routine nag_estimate_garchGJR provides an estimate for $\hat{\theta}$, the $(p + q + k + 3) \times 1$ parameter vector $\theta = (b_0, b^T, \omega^2)$ where $\omega^2 = (\alpha_0, \alpha_1, \ldots, \alpha_q, \beta_1, \ldots, \beta_p, \gamma)$ and $b^T = (b_1, \ldots, b_k)$.

**mn, nreg** (see Section 4) can be used to simplify the GARCH(p,q) expression in equation (1) as follows:

**No Regression or Mean**

$$y_t = \epsilon_t,$$

$$\text{mn} = 0,$$

$$\text{nreg} = 0, \text{ and}$$

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

**No Regression**

$$y_t = b_0 + \epsilon_t,$$

$$\text{mn} = 1,$$

$$\text{nreg} = 0, \text{ and}$$

$\theta$ is a $(p + q + 3) \times 1$ vector.
Note: if the $y_t = \mu + \epsilon_t$, where $\mu$ is known (not to be estimated by nag_estimate_garchGJR) then equation (1) can be written as $y_t^u = \epsilon_t$, where $y_t^u = y_t - \mu$. This corresponds to the case No Regression or Mean, with $y_t$ replaced by $y_t - \mu$.

No Mean

$$y_t = x_t^Tb + \epsilon_t,$$

$$mn = 0,$$

$$nreg = k$$

and

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

4 Parameters

Note: for convenience npar will be used here to denote the expression $2+q+p+mn+nreg$ representing the number of model parameters.

1: $y\left[num\right]$ – const double

*Input*

*On entry:* the sequence of observations, $y_t$, $t = 1, \ldots, T$.

2: $x\left[num\middle|tdx\right]$ – const double

*Input*

*On entry:* row $t$ of $x$ contains the time dependent exogenous vector $x_t$, where $x_t^T = (x_t^1, \ldots, x_t^k)$, for $t = 1, \ldots, T$.

3: $tdx$ – Integer

*Input*

*On entry:* the second dimension of the array $x$ as declared in the function from which nag_estimate_garchGJR is called.

*Constraint:* $tdx \geq nreg$.

4: $num$ – Integer

*Input*

*On entry:* the number of terms in the sequence, $T$.

*Constraint:* $num \geq npar$.

5: $p$ – Integer

*Input*

*On entry:* the GARCH($p, q$) parameter $p$.

*Constraint:* $p \geq 0$.

6: $q$ – Integer

*Input*

*On entry:* the GARCH($p, q$) parameter $q$.

*Constraint:* $q \geq 1$.

7: $nreg$ – Integer

*Input*

*On entry:* the number of regression coefficients, $k$.

*Constraint:* $nreg \geq 0$.

8: $mn$ – Integer

*Input*

*On entry:* if $mn = 1$ then the mean term $b_0$ will be included in the model.

*Constraint:* $mn = 0$ or $mn = 1$. 
theta[npar] – double

On entry: the initial parameter estimates for the vector \( \theta \). The first element contains the coefficient \( \alpha_o \), the next \( q \) elements contain the coefficients \( \alpha_i, i = 1, \ldots, q \). The next \( p \) elements are the coefficients \( \beta_j, j = 1, \ldots, p \). The next element contains the asymmetry parameter \( \gamma \). If est_opt = Nag_Garch_Est_Initial_False then (when \( mu = 1 \)) 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 \( b_i, i = 1, \ldots, k \).

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 \( \alpha_i, i = 1, \ldots, q \). The next \( p \) elements are the coefficients \( \beta_j, j = 1, \ldots, p \). The next element contains the estimate for the asymmetry parameter \( \gamma \). If \( mu = 1 \) then the next element contains an estimate for the mean term \( \hat{b}_0 \). The final \text{nreg} elements are the estimated linear regression coefficients \( \hat{b}_i, i = 1, \ldots, k \).

se[npar] – double

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 \( \alpha_i, i = 1, \ldots, q \), the next \( p \) elements are the standard errors for \( \beta_j, j = 1, \ldots, p \). The next element contains the standard error for \( \gamma \). If \( mu = 1 \) then the next element contains the standard error for \( \hat{b}_0 \). The final \text{nreg} elements are the standard errors for \( \hat{b}_i, j = 1, \ldots, k \).

sc[npar] – double

On exit: the scores for \( \hat{\theta} \). The first element contains the score for \( \alpha_o \), the next \( q \) elements contain the score for \( \alpha_i, i = 1, \ldots, q \), the next \( p \) elements are the scores for \( \beta_j, j = 1, \ldots, p \). The next element contains the score for \( \gamma \). If \( mu = 1 \) then the next element contains the score for \( \hat{b}_0 \). The final \text{nreg} elements are the scores for \( \hat{b}_i, j = 1, \ldots, k \).

covar[npar][tdc] – double

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


tdc – Integer

On entry: the second dimension of the array covar as declared in the function from which nag_estimate_garchGJR is called.

Constraint: \( tdc \geq npar \).

hp – double *

On entry: If est_opt = Nag_Garch_Est_Initial_False then hp is the value to be used for the pre-observed conditional variance. If est_opt = Nag_Garch_Est_Initial_True then hp is not referenced.

On exit: If est_opt = Nag_Garch_Est_Initial_True then hp is the estimated value of the pre-observed conditional variance.

et[num] – double

On exit: the estimated residuals, \( \epsilon_t, t = 1, \ldots, T \).

ht[num] – double

On exit: the estimated conditional variances, \( h_t, t = 1, \ldots, T \).

lgf – double *

On exit: the value of the log likelihood function at \( \hat{\theta} \).
18:  stat_opt – Nag_Garch_Stationary_Type  
  
  On entry: If stat_opt = Nag_Garch_Stationary_True then Stationary conditions are enforced. If stat_opt = Nag_Garch_Stationary_False then Stationary conditions are not enforced.

19:  est_opt – Nag_Garch_Est_Initial_Type  
  
  On entry: If est_opt = Nag_Garch_Est_Initial_True then the routine provides initial parameter estimates of the regression terms (b0, bT). If est_opt = Nag_Garch_Est_Initial_False then the initial estimates of the regression parameters (b0, bT) must be supplied by the user.

20:  max_iter – Integer  
  
  On entry: the maximum number of iterations to be used by the optimisation routine when estimating the GARCH(p, q) parameters. If max_iter is set to 0 then the standard errors, score vector and variance-covariance are calculated for the input value of θ in theta; however the value of θ is not updated.
  
  Constraint: max_iter ≥ 0.

21:  tol – double  
  
  On entry: the tolerance to be used by the optimisation routine when estimating the GARCH(p, q) parameters.

22:  fail – NagError *  
  
  The NAG error parameter (see the Essential Introduction).

5  Error Indicators and Warnings

NE_BAD_PARAM
  
  On entry, parameter stat_opt had an illegal value.
  On entry, parameter est_opt had an illegal value.

NE_INT_ARG_LT
  
  On entry, nreg must not be less than 0: nreg = <value>.
  On entry, q must not be less than 1: q = <value>.
  On entry, p must not be less than 0: p = <value>.
  On entry, max_iter must not be less than 0: max_iter = <value>.

NE_2_INT_ARG_LT
  
  On entry, tdx = <value> while nreg = <value>.
  These parameters must satisfy tdx ≥ nreg.

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

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

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.

NE_ALLOC_FAIL

Memory allocation failed.

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 consult NAG for assistance.

6 Further Comments

6.1 Accuracy

Not applicable.

6.2 References


7 See Also

None.

8 Example

This example program illustrates the use of nag_estimate_garchGJR to model a GARCH(1,1) sequence generated by nag_generate_garchGJR (g05hmc), a six step forecast is then calculated using nag_forecast_garchGJR (g13frc).

8.1 Program Text

/* nag_estimate_garchGJR (g13fec) Example Program.  *
   * Copyright 2000 Numerical Algorithms Group.     *
   *  NAG C Library                                  *
   *  Mark 6, 2000.                                  *
   */
#include <nag.h>
#include <nag_stdf.h>
#include <stdio.h>
#include <ctype.h>
#include <math.h>
#include <nagc05.h>
#include <naggl3.h>

int main(void) {
    double *bx=0, *covar=0, *etm=0, fac1, gamma, hp, *ht=0, *htm=0, lgf;
    double *param=0, *rvec=0, *sc=0, *se=0, *theta=0, tol;
    double mean, *x=0, xterm, *cvar=0, *yt=0;
    Integer exit_status = 0;
    Integer i, ip, iq, j, k, nt;
    Integer tdx, tdc, maxit, mn, num, num_startup, npar;
    Integer nreg, seed;

    Nag_Garch_Est_Initial_Type est_opt;
    Nag_Garch_Stationary_Type stat_opt;
    Nag_Garch_Fcall_Type fcall;
    NagError fail;

    INIT_FAIL(fail);
    num = 1000;
    mn = 1;
    mean = 4.0;
    nreg = 2;
    ip = 1;
    iq = 1;
    npar = iq + ip + 1;
    nt = 6;
    tdx = nreg;
    tdc = npar+mn+nreg+1;

#define YT(I) yt[(I)-1]
#define THETA(I) theta[(I)-1]
#define SE(I) se[(I)-1]
#define SC(I) sc[(I)-1]
#define RVEC(I) rvec[(I)-1]
#define PARAM(I) param[(I)-1]
#define HTM(I) htm[(I)-1]
#define HT(I) ht[(I)-1]
#define ETM(I) etm[(I)-1]
#define BX(I) bx[(I)-1]
#define CVAR(I) cvar[(I)-1]
#define X(I,J) x[((I)-1) * tdx + ((J)-1)]
#define COVAR(I,J) covar[((I)-1) * tdx + ((J)-1)]

    Vprintf ("gL3fec Example Program Results \n\n");

if (!bx = NAG_ALLOC (nreg, double))
    |   !covar = NAG_ALLOC ((npar+mn+nreg+1) * (npar+mn+nreg+1), double))
    |   !etm = NAG_ALLOC (num, double))
    |   !ht = NAG_ALLOC (num, double))
    |   !htm = NAG_ALLOC (num, double))
    |   !param = NAG_ALLOC (npar+mn+nreg+1, double))
    |   !rvec = NAG_ALLOC (40, double))
    |   !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*nreg, double))

gL3fec.6

NAG C Library Manual
NP3491/6


```c
|| !(yt = NAG_ALLOC (num, double)))
{
    Vprintf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

seed = 11;
gamma = 0.1;
BX (1) = 1.5;
BX (2) = 2.5;

for (i = 1; i <= num; ++i)
{
    fac1 = (double) i * 0.01;
    X (i, 2) = sin (fac1) * 0.7 + 0.01;
    X (i, 1) = fac1 * 0.1 + 0.5;
}

PARAM (1) = 0.4;
PARAM (2) = 0.1;
PARAM (3) = 0.7;

fcall = Nag_Garch_Fcall_True;
go5c6c(seed);
num_startup = 200;
go5hmc (num_startup, ip, iq, &PARAM (1), gamma, &HT (1), &YT (1),
fcall, &RVEC (1), &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from go5hmc.\n\n", fail.message);
    exit_status = 1;
    goto END;
}

fcall = Nag_Garch_Fcall_False;
go5hmc (num, ip, iq, &PARAM (1), gamma, &HT (1), &YT (1),
fcall, &RVEC (1), &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from go5hmc.\n\n", fail.message);
    exit_status = 1;
    goto END;
}

for (i = 1; i <= num; ++i)
{
    xterm = 0.0;
    for (k = 1; k <= nreg; ++k)
        xterm += X (i, k) * BX (k);

    if (mn == 1)
        YT (i) = mean + xterm + YT (i);
    else
        YT (i) = xterm + YT (i);
}

est_opt = Nag_Garch_Est_Initial_TRUE;
stat_opt = Nag_Garch_Stationary_TRUE;
maxit = 50;
```
tol = 1e-12;

for (i = 1; i <= npar; ++i)
    THETA (i) = PARAM (i) * 0.5;
    THETA (npar + 1) = gamma * 0.5;
    if (mn == 1)
        THETA (npar + 2) = mean * 0.5;

for (i = 1; i <= nreg; ++i)
    THETA (npar + 1 + mn + i) = BX (i) * 0.5;

gl3fec (&YT (1), &X (1, 1), tdx, num, ip, iq, nreg, mn,
    &THETA (1), &SE (1), &SC (1), &COVAR (1, 1), tdc, &hp,
    &ETM (1), &HTM (1), &lgf, stat_opt, est_opt, maxit, tol, &fail);
    if (fail.code != NE_NOERROR)
        {
            Vprintf("Error from gl3fec.\n\n", fail.message);
            exit_status = 1;
            goto END;
        }

        Vprintf ("Parameter estimates Standard errors Correct va-

        lues\n");
        for (j = 1; j <= npar; ++j)
            Vprintf ("%20.4f (%6.4f) %20.4f\n", THETA (j), SE (j),
            PARAM (j));

        Vprintf ("%20.4f (%6.4f) %20.4f\n", THETA (npar+1), SE (npar+1),
            gamma);
        if (mn)
            Vprintf ("%20.4f (%6.4f) %20.4f\n", THETA (npar+2), SE (npar+2),
                mean);
        for (j = 1; j <= nreg; ++j)
            Vprintf ("%20.4f (%6.4f) %20.4f\n", THETA (npar+1+mn+j), SE(n-
                par+1+mn+j), BX(j));

    /* now forecast nt steps ahead */

    gamma = THETA(npar+1);

    gl3ffc(num,nt,ip,iq,&THETA(1),gamma,&CVAR(1),&HTM(1),&ETM(1),&fail);
    Vprintf ("\n\ld step forecast = %8.4f\n",nt,CVAR(nt));

END:
    if (bx) NAG_FREE (bx);
    if (covar) NAG_FREE (covar);
    if (etm) NAG_FREE (etm);
    if (ht) NAG_FREE (ht);
    if (htm) NAG_FREE (htm);
    if (param) NAG_FREE (param);
    if (sc) NAG_FREE (sc);
    if (se) NAG_FREE (se);
    if (theta) NAG_FREE (theta);
    if (cvar) NAG_FREE (cvar);
    if (x) NAG_FREE (x);
    if (yt) NAG_FREE (yt);
    if (rvec) NAG_FREE (rvec);
8.2 Program Data
None.

8.3 Program Results

<table>
<thead>
<tr>
<th>Parameter estimates</th>
<th>Standard errors</th>
<th>Correct values</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.4326</td>
<td>(0.1356)</td>
<td>0.4000</td>
</tr>
<tr>
<td>0.0685</td>
<td>(0.0333)</td>
<td>0.1000</td>
</tr>
<tr>
<td>0.7173</td>
<td>(0.0672)</td>
<td>0.7000</td>
</tr>
<tr>
<td>0.1326</td>
<td>(0.0553)</td>
<td>0.1000</td>
</tr>
<tr>
<td>4.1205</td>
<td>(0.1730)</td>
<td>4.0000</td>
</tr>
<tr>
<td>1.3950</td>
<td>(0.1658)</td>
<td>1.5000</td>
</tr>
<tr>
<td>2.4518</td>
<td>(0.1037)</td>
<td>2.5000</td>
</tr>
</tbody>
</table>

6 step forecast = 2.2549