

# NAG Library Function Document

## nag\_estimate\_agarchII (g13fcc)

### 1 Purpose

nag\_estimate\_agarchII (g13fcc) estimates the arguments of a univariate regression-type II AGARCH( $p, q$ ) process.

### 2 Specification

```
#include <nag.h>
#include <naggl3.h>

void nag_estimate_agarchII (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-type II AGARCH( $p, q$ ) process, with  $p$  coefficients  $\alpha_i$ , for  $i = 1, 2, \dots, p$ ,  $q$  coefficients,  $\beta_i$ , for  $i = 1, 2, \dots, q$ , mean  $b_o$ , and  $k$  linear regression coefficients  $b_i$ , for  $i = 1, 2, \dots, k$ , can be represented by:

$$y_t = b_o + x_t^T b + \epsilon_t \quad (1)$$

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

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

Here  $T$  is the number of terms in the sequence,  $y_t$  denotes the endogenous variables,  $x_t$  the exogenous variables,  $b_o$  the mean,  $b$  the regression coefficients,  $\epsilon_t$  the residuals,  $\gamma$  the asymmetry argument,  $h_t$  the conditional variance, and  $\psi_t$  the information set of all information up to time  $t$ .

nag\_estimate\_agarchII (g13fcc) provides an estimate for  $\hat{\theta}$ , the  $(p + q + k + 3) \times 1$  parameter vector  $\theta = (b_o, b^T, \omega^T)$  where  $\omega^T = (\alpha_0, \alpha_1, \dots, \alpha_q, \beta_1, \dots, \beta_p, \gamma)$  and  $b^T = (b_1, \dots, b_k)$ .

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

#### No Regression or Mean

$$y_t = \epsilon_t,$$

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

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

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

#### No Regression

$$y_t = b_o + \epsilon_t,$$

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

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

$$\theta \text{ is a } (p + q + 3) \times 1 \text{ vector.}$$

**Note:** if the  $y_t = \mu + \epsilon_t$ , where  $\mu$  is known (not to be estimated by nag\_estimate\_agarchII (g13fcc)) 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

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

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

$$\mathbf{nreg} = k \text{ and}$$

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

## 4 References

Bollerslev T (1986) Generalised autoregressive conditional heteroskedasticity *Journal of Econometrics* **31** 307–327

Engle R (1982) Autoregressive conditional heteroskedasticity with estimates of the variance of United Kingdom inflation *Econometrica* **50** 987–1008

Engle R and Ng V (1993) Measuring and testing the impact of news on volatility *Journal of Finance* **48** 1749–1777

Hamilton J (1994) *Time Series Analysis* Princeton University Press

## 5 Arguments

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

- 1: **yt[num]** – const double *Input*  
*On entry:* the sequence of observations,  $y_t$ , for  $t = 1, 2, \dots, T$ .
- 2: **x[num × tdx]** – const double *Input*  
**Note:** the  $i$ th element of the  $j$ th vector  $X$  is stored in  $\mathbf{x}[(i - 1) \times \mathbf{tdx} + j - 1]$ .  
*On entry:* row  $t$  of  $\mathbf{x}$  must contain the time dependent exogenous vector  $x_t$ , where  $x_t^T = (x_t^1, \dots, x_t^k)$ , for  $t = 1, 2, \dots, T$ .
- 3: **tdx** – Integer *Input*  
*On entry:* the stride separating matrix column elements in the array  $\mathbf{x}$ .  
*Constraint:*  $\mathbf{tdx} \geq \mathbf{nreg}$ .
- 4: **num** – Integer *Input*  
*On entry:* the number of terms in the sequence,  $T$ .  
*Constraint:*  $\mathbf{num} \geq \mathbf{npar}$ .
- 5: **p** – Integer *Input*  
*On entry:* the GARCH( $p, q$ ) argument  $p$ .  
*Constraint:*  $\mathbf{p} \geq 0$ .
- 6: **q** – Integer *Input*  
*On entry:* the GARCH( $p, q$ ) argument  $q$ .  
*Constraint:*  $\mathbf{q} \geq 1$ .

- 7: **nreg** – Integer Input  
*On entry:*  $k$ , the number of regression coefficients.  
*Constraint:*  $\mathbf{nreg} \geq 0$ .
- 8: **mn** – Integer Input  
*On entry:* if  $\mathbf{mn} = 1$ , the mean term  $b_0$  will be included in the model.  
*Constraint:*  $\mathbf{mn} = 0$  or  $1$ .
- 9: **theta**[*npar*] – double Input/Output  
*On entry:* the initial parameter estimates for the vector  $\theta$ .  
 The first element contains the coefficient  $\alpha_o$ , the next  $\mathbf{q}$  elements contain the autoregressive coefficients  $\alpha_i$ , for  $i = 1, 2, \dots, q$ .  
 The next  $\mathbf{p}$  elements are the moving average coefficients  $\beta_j$ , for  $j = 1, 2, \dots, p$ .  
 The next element contains the asymmetry argument  $\gamma$ .  
 If **est\_opt** = Nag\_Garch\_Est\_Initial\_False, (when  $\mathbf{mn} = 1$ ) the next term contains an initial estimate of the mean term  $b_o$  and the remaining **nreg** elements are taken as initial estimates of the linear regression coefficients  $b_i$ , for  $i = 1, 2, \dots, k$ .  
*On exit:* the estimated values  $\hat{\theta}$  for the vector  $\theta$ .  
 The first element contains the coefficient  $\alpha_o$ , the next  $\mathbf{q}$  elements contain the coefficients  $\alpha_i$ , for  $i = 1, 2, \dots, q$ .  
 The next  $\mathbf{p}$  elements are the coefficients  $\beta_j$ , for  $j = 1, 2, \dots, p$ .  
 The next element contains the estimate for the asymmetry parameter  $\gamma$ .  
 If  $\mathbf{mn} = 1$ , the next element contains an estimate for the mean term  $b_o$ .  
 The final **nreg** elements are the estimated linear regression coefficients  $b_i$ , for  $i = 1, 2, \dots, k$ .
- 10: **se**[*npar*] – double Output  
*On exit:* the standard errors for  $\hat{\theta}$ .  
 The first element contains the standard error for  $\alpha_o$ .  
 The next  $\mathbf{q}$  elements contain the standard errors for  $\alpha_i$ , for  $i = 1, 2, \dots, q$ .  
 The next  $\mathbf{p}$  elements are the standard errors for  $\beta_j$ , for  $j = 1, 2, \dots, p$ .  
 The next element contains the standard error for  $\gamma$ .  
 If  $\mathbf{mn} = 1$ , the next element contains the standard error for  $b_o$ .  
 The final **nreg** elements are the standard errors for  $b_j$ , for  $j = 1, 2, \dots, k$ .
- 11: **sc**[*npar*] – double Output  
*On exit:* the scores for  $\hat{\theta}$ .  
 The first element contains the score for  $\alpha_o$ .  
 The next  $\mathbf{q}$  elements contain the score for  $\alpha_i$ , for  $i = 1, 2, \dots, q$ .  
 The next  $\mathbf{p}$  elements are the scores for  $\beta_j$ , for  $j = 1, 2, \dots, p$ .  
 The next element contains the score for  $\gamma$ .  
 If  $\mathbf{mn} = 1$ , the next element contains the score for  $b_o$ .  
 The final **nreg** elements are the scores for  $b_j$ , for  $j = 1, 2, \dots, k$ .

- 12: **covar**[ $npar \times tdc$ ] – double *Output*  
**Note:** the  $(i, j)$ th element of the matrix is stored in **covar**[( $i - 1$ )  $\times$   $tdc + j - 1$ ].  
*On exit:* the covariance matrix of the parameter estimates  $\hat{\theta}$ , that is the inverse of the Fisher Information Matrix.
- 13: **tdc** – Integer *Input*  
*On entry:* the stride separating matrix column elements in the array **covar**.  
*Constraint:* **tdc**  $\geq npar$ .
- 14: **hp** – double \* *Input/Output*  
*On entry:* if **est\_opt** = Nag\_Garch\_Est\_Initial\_False, **hp** is the value to be used for the pre-observed conditional variance.  
 If **est\_opt** = Nag\_Garch\_Est\_Initial\_True, **hp** is not referenced.  
*On exit:* if **est\_opt** = Nag\_Garch\_Est\_Initial\_True, **hp** is the estimated value of the pre-observed of the conditional variance.
- 15: **et[num]** – double *Output*  
*On exit:* the estimated residuals,  $\epsilon_t$ , for  $t = 1, 2, \dots, T$ .
- 16: **ht[num]** – double *Output*  
*On exit:* the estimated conditional variances,  $h_t$ , for  $t = 1, 2, \dots, T$ .
- 17: **lgf** – double \* *Output*  
*On exit:* the value of the log-likelihood function at  $\hat{\theta}$ .
- 18: **stat\_opt** – Nag\_Garch\_Stationary\_Type *Input*  
*On entry:* if **stat\_opt** = Nag\_Garch\_Stationary\_True, Stationary conditions are enforced.  
 If **stat\_opt** = Nag\_Garch\_Stationary\_False, Stationary conditions are not enforced.  
*Constraint:* **stat\_opt** = Nag\_Garch\_Stationary\_True or Nag\_Garch\_Stationary\_False.
- 19: **est\_opt** – Nag\_Garch\_Est\_Initial\_Type *Input*  
*On entry:* if **est\_opt** = Nag\_Garch\_Est\_Initial\_True, the function provides initial parameter estimates of the regression terms  $(b_0, b^T)$ .  
 If **est\_opt** = Nag\_Garch\_Est\_Initial\_False, you must supply the initial estimations of the regression parameters  $(b_0, b^T)$ .  
*Constraint:* **est\_opt** = Nag\_Garch\_Est\_Initial\_True or Nag\_Garch\_Est\_Initial\_False.
- 20: **max\_iter** – Integer *Input*  
*On entry:* the maximum number of iterations to be used by the optimization function when estimating the GARCH( $p, q$ ) arguments. If **max\_iter** is set to 0, the standard errors, score vector and variance-covariance are calculated for the input value of  $\theta$  in **theta**; however the value of  $\theta$  is not updated.  
*Constraint:* **max\_iter**  $\geq 0$ .
- 21: **tol** – double *Input*  
*On entry:* the tolerance to be used by the optimization function when estimating the GARCH( $p, q$ ) parameters.

22: **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

### NE\_2\_INT\_ARG\_LT

On entry, **num** =  $\langle value \rangle$  while  $2 + \mathbf{q} + \mathbf{p} + \mathbf{mn} + \mathbf{nreg} = \langle value \rangle$ . These arguments must satisfy  $\mathbf{num} \geq 2 + \mathbf{q} + \mathbf{p} + \mathbf{mn} + \mathbf{nreg}$ .

On entry, **tdc** =  $\langle value \rangle$  while  $2 + \mathbf{q} + \mathbf{p} + \mathbf{mn} + \mathbf{nreg} = \langle value \rangle$ . These arguments must satisfy  $\mathbf{tdc} \geq 2 + \mathbf{q} + \mathbf{p} + \mathbf{mn} + \mathbf{nreg}$ .

On entry, **tdx** =  $\langle value \rangle$  while  $\mathbf{nreg} = \langle value \rangle$ . These arguments must satisfy  $\mathbf{tdx} \geq \mathbf{nreg}$ .

### NE\_ALLOC\_FAIL

Dynamic memory allocation failed.

### NE\_BAD\_PARAM

On entry, argument **est\_opt** had an illegal value.

On entry, argument **stat\_opt** had an illegal value.

### NE\_INT\_ARG\_LT

On entry, **max\_iter** must not be less than 0: **max\_iter** =  $\langle value \rangle$ .

On entry, **nreg** =  $\langle value \rangle$ .

Constraint:  $\mathbf{nreg} \geq 0$ .

On entry, **p** =  $\langle value \rangle$ .

Constraint:  $\mathbf{p} \geq 0$ .

On entry, **q** =  $\langle value \rangle$ .

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

### 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  $\langle value \rangle$  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

nag\_estimate\_agarchII (g13fcc) is not threaded in any implementation.

## 9 Further Comments

None.

## 10 Example

This example program illustrates the use of nag\_estimate\_agarchII (g13fcc) to model a GARCH(1,1) sequence generated by nag\_rand\_agarchII (g05pec), a three step forecast is then calculated using nag\_forecast\_agarchII (g13fdc).

### 10.1 Program Text

```

/* nag_estimate_agarchII (g13fcc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * NAG C Library
 *
 * Mark 26, 2016.
 */
#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, lstate, lr;
    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, *et = 0, *ht = 0, *sc = 0;
    double *se = 0, *theta = 0, *x = 0, *yt = 0, *r = 0;

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

    /* Set the seed */
    Integer seed[] = { 111 };
    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 = 1500;

/* MA and AR parameters */
Integer ip = 1;
Integer iq = 1;
double param[] = { 0.08, 0.2, 0.7 };

/* Asymmetry parameter */
double gamma = -0.4;

/* Regression parameters */
Integer nreg = 2;
double mean = 3.0;
double bx[] = { 1.5, 2.5 };
/* ... end of parameters for (randomly generated) time series */

/* 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 = 3;
/* ... end of model fitting options */

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

printf("nag_estimate_agarchII (g13fcc) Example Program Results\n\n");

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

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

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

/* Allocate arrays */
if (!(covar = NAG_ALLOC((npar + mn + nreg + 1) * tdc, double)) ||
    !(et = NAG_ALLOC(num, double)) ||
    !(ht = NAG_ALLOC(num, double)) ||
    !(sc = NAG_ALLOC(npar + mn + nreg + 1, double)) ||
    !(se = NAG_ALLOC(npar + mn + nreg + 1, double)) ||
    !(state = NAG_ALLOC(lstate, Integer)) ||
    !(r = NAG_ALLOC(lr, double)) ||
    !(theta = NAG_ALLOC(npar + mn + nreg + 1, double)) ||
    !(x = NAG_ALLOC(num * tdx, double)) ||
    !(cvar = NAG_ALLOC(nt, double)) || !(yt = NAG_ALLOC(num, double)))
{
    printf("Allocation failure\n");
    exit_status = -1;
}

```

```

    goto END;
}

/* Initialize 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) * .01;
    X(i, 0) = sin(fac1) * 0.7 + 0.01;
    X(i, 1) = fac1 * 0.1 + 0.5;
}

/* Generate a realization of a random AGARCH II time series to use */
fcall = Nag_TRUE;
nag_rand_agarchII(dist, num, ip, iq, param, gamma, df, ht, yt, fcall, r, lr,
    state, &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_rand_agarchII (g05pec).\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 + mn] = mean * 0.5;
}
if (est_opt != Nag_Garch_Est_Initial_True) {
    for (i = 0; i < nreg; ++i)
        theta[npar + mn + 1 + i] = bx[i] * 0.5;
}

/* nag_estimate_agarchII (g13fcc).
 * Univariate time series, parameter estimation for a GARCH
 * process with asymmetry of the form
 *  $(|\epsilon_{t-1}| + \gamma \epsilon_{t-1})^2$ 
 */
nag_estimate_agarchII(yt, x, tdx, num, ip, iq, nreg, mn, theta, se, sc,
    covar, tdc, &hp, et, ht, &lgf, stat_opt, est_opt,
    maxit, tol, &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_estimate_agarchII (g13fcc).\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 == 1)
    printf("%20.4f          (%6.4f) %20.4f\n", theta[npar + mn],
           se[npar + mn], mean);
for (j = 0; j < nreg; ++j)
    printf("%20.4f          (%6.4f) %20.4f\n",
           theta[mn + npar + 1 + j], se[mn + npar + 1 + j], bx[j]);

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

/* nag_forecast_agarchII (g13fdc).
 * Univariate time series, forecast function for a GARCH
 * process with asymmetry of the form
 * (|epsilon_(t-1)| + gamma epsilon_(t-1))^2
 */
nag_forecast_agarchII(num, nt, ip, iq, theta, gamma, cvar, ht, et, &fail);
printf("\n%" NAG_IFMT " step forecast = %8.4f\n", nt, cvar[nt - 1]);

END:
NAG_FREE(covar);
NAG_FREE(et);
NAG_FREE(ht);
NAG_FREE(state);
NAG_FREE(r);
NAG_FREE(sc);
NAG_FREE(se);
NAG_FREE(theta);
NAG_FREE(cvar);
NAG_FREE(x);
NAG_FREE(yt);

return exit_status;
}

```

## 10.2 Program Data

None.

## 10.3 Program Results

nag\_estimate\_agarchII (g13fcc) Example Program Results

Parameter estimates	Standard errors	Correct values
0.0916	(0.0147)	0.0800
0.2268	(0.0324)	0.2000
0.6596	(0.0326)	0.7000
-0.4188	(0.0686)	-0.4000
2.9804	(0.0565)	3.0000
1.5345	(0.0362)	1.5000
2.5314	(0.0424)	2.5000

3 step forecast = 0.4238

---