

NAG C Library Function Document

nag_estimate_agarchI (g13fac)

1 Purpose

nag_estimate_agarchI (g13fac) estimates the parameters of a standard univariate regression-GARCH(p, q) or a univariate regression-type I AGARCH(p, q) process (see Engle and Ng (1993)).

2 Specification

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

void nag_estimate_agarchI (const double yt[], const double x[], Integer tdx,
    Integer num, Integer p, Integer q, Integer nreg, Integer mn,
    Integer isym, 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

When **isym** = 0, nag_estimate_agarchI models a standard ($\gamma = 0$) univariate regression-GARCH(p, q) process, with p coefficients α_i , $i = 1, \dots, p$, q coefficients, β_i , $i = 1, \dots, q$, mean b_o , and k linear regression coefficients b_i , $i = 1, \dots, k$, which can be represented by:

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

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

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

When **isym** = 1, nag_estimate_agarchI models an asymmetric GARCH(p, q) process where the conditional variance h_t is given by:

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

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, ϵ_t the residuals, γ is the asymmetry parameter, h_t is the conditional variance, and ψ_t the information set of all information up to time t .

When **isym** = 1, nag_estimate_agarchI 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 $b^T = (b_1, \dots, b_k)$.

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

No Regression or Mean

$$y_t = \epsilon_t,$$

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

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

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

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

No Regression

$$y_t = b_0 + \epsilon_t,$$

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

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

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

θ is a $(p + q + 2) \times 1$ vector.

Note: if the $y_t = \mu + \epsilon_t$, where μ is known (not to be estimated by `nag_estimate_agarchI`) then (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

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

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

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

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

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

4 Parameters

Note: for convenience **npar** will be used here to denote the expression $1 + \mathbf{q} + \mathbf{p} + \text{isym} + \text{mn} + \text{nreg}$ representing the number of model parameters.

- 1: **yt[num]** – const double *Input*
On entry: the sequence of observations, y_t , $t = 1, \dots, T$.
- 2: **x[num][tdx]** – const double *Input*
On entry: row t of **x** contains the time dependent exogenous vector x_t , where $x_t^T = (x_t^1, \dots, x_t^k)$, for $t = 1, \dots, T$.
- 3: **tdx** – Integer *Input*
On entry: the second dimension of the array **x** as declared in the function from which `nag_estimate_agarchI` 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.
- 9: **isym** – Integer Input
On entry: if **isym** = 1 then the asymmetry term γ will be included in the model.
Constraint: **isym** = 0 or **isym** = 1
- 10: **theta[npar]** – double Input/Output
On entry: the initial parameter estimates for the vector θ . The first element contains the coefficient α_o , the next **q** elements contain the coefficients α_i , $i = 1, \dots, q$. The next **p** elements are the coefficients β_j , $j = 1, \dots, p$. If **isym** = 1 then the next element contains the asymmetry parameter γ . If **est_opt** = **Nag_Garch_Est_Initial_False** then (when **mn** = 1) the next term contains an initial estimate of the mean term b_0 and the remaining **nreg** elements are taken as initial estimates of the linear regression coefficients b_i , $i = 1, \dots, k$.
On exit: the estimated values $\hat{\theta}$ for the vector θ . The first element contains the coefficient α_o , the next **q** elements contain the coefficients α_i , $i = 1, \dots, q$. The next **p** elements are the coefficients β_j , $j = 1, \dots, p$. If **isym** = 1 then the next element contains the estimate for the asymmetry parameter γ . If **mn** = 1 then the next element contains an estimate for the mean term b_0 . The final **nreg** elements are the estimated linear regression coefficients b_i , $i = 1, \dots, k$.
- 11: **se[npar]** – double Output
On exit: the standard errors for $\hat{\theta}$. The first element contains the standard error for α_o , the next **q** elements contain the standard errors for α_i , $i = 1, \dots, q$, the next **p** elements are the standard errors for β_j , $j = 1, \dots, p$. If **isym** = 1 then the next element contains the standard error for γ . If **mn** = 1 then the next element contains the standard error for b_0 . The final **nreg** elements are the standard errors for b_j , $j = 1, \dots, k$.
- 12: **sc[npar]** – double Output
On exit: the scores for $\hat{\theta}$. The first element contains the score for α_o , the next **q** elements contain the score for α_i , $i = 1, \dots, q$, the next **p** elements are the scores for β_j , $j = 1, \dots, p$. If **isym** = 1 then the next element contains the score for γ . If **mn** = 1 then the next element contains the score for b_0 . The final **nreg** elements are the scores for b_j , $j = 1, \dots, k$.
- 13: **covar[npar][tdc]** – double Output
On exit: the covariance matrix of the parameter estimates $\hat{\theta}$, that is the inverse of the Fisher Information Matrix.
- 14: **tdc** – Integer Input
On entry: the second dimension of the array **covar** as declared in the function from which `nag_estimate_agarchI` is called.
Constraint: **tdc** \geq **npar**.
- 15: **hp** – double * Input/Output
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.
- 16: **et[num]** – double *Output*
On exit: the estimated residuals, ϵ_t , $t = 1, \dots, T$.
- 17: **ht[num]** – double *Output*
On exit: the estimated conditional variances, h_t , $t = 1, \dots, T$.
- 18: **lgf** – double * *Output*
On exit: the value of the log likelihood function at $\hat{\theta}$.
- 19: **stat_opt** – Nag_Garch_Stationary_Type *Input*
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.
- 20: **est_opt** – Nag_Garch_Est_Initial_Type *Input*
On entry: If **est_opt** = **Nag_Garch_Est_Initial_True** then the routine provides initial parameter estimates of the regression terms (b_o, b^T). If **est_opt** = **Nag_Garch_Est_Initial_False** then the initial estimates of the regression parameters (b_o, b^T) must be supplied by the user.
- 21: **max_iter** – Integer *Input*
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** \geq 0.
- 22: **tol** – double *Input*
On entry: the tolerance to be used by the optimisation routine when estimating the GARCH(p, q) parameters.
- 23: **fail** – NagError * *Input/Output*
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** \geq **nreg**.

On entry, **tdc** = *<value>* while $1+q+p+isym+mn+nreg = <value>$.

These parameters must satisfy $tdc \geq 1+q+p+isym+mn+nreg$.

On entry, **num** = *<value>* while $1+q+p+isym+mn+nreg = <value>$.

These parameters must satisfy $num \geq 1+q+p+isym+mn+nreg$.

NE_INVALID_INT_RANGE_2

Value *<value>* given to **mn** is not valid. Correct range is 0 to 1.

Value *<value>* given to **isym** 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

Engle R (1982) Autoregressive Conditional Heteroskedasticity with Estimates of the Variance of United Kingdom Inflation *Econometrica* **50** 987–1008

Bollerslev T (1986) Generalised Autoregressive Conditional Heteroskedasticity *Journal of Econometrics* **31** 307–327

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

7 See Also

None.

8 Example

This example program illustrates the use of `nag_estimate_agarchI` to model a GARCH(1,1) sequence generated by `nag_generate_agarchI` (`g05hkc`), a three step forecast is then calculated using `nag_forecast_agarchI` (`g13fbc`).

8.1 Program Text

```

/* nag_estimate_agarchI (g13fac) Example Program.
 *
 * Copyright 2000 Numerical Algorithms Group.
 *
 * 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>

int main(void)
{
    double *bx=0, *covar=0, *et=0, fac1, gamma, hp;
    double *ht=0, lgf, mean, *param=0;
    double *rvec=0, *sc=0, *se=0, *theta=0, tol;
    double *x=0, xterm, *yt=0, *cvar=0;
    Integer i, ip, iq, isym, j, k, nt;
    Integer exit_status=0;
    Integer tdc, tdx;
    Integer maxit, mn, num, num_startup, npar, 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);
    isym = 1;
    gamma = -0.3;
    nreg = 2;
    ip = 1;
    iq = 1;
    mn = 1;

    npar = iq + ip + 1;
    num = 1000;
    nt = 3;

    tdc = npar+mn+isym+nreg;
    tdx = nreg;

#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 HT(I) ht[(I)-1]
#define ET(I) et[(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) * tdc + ((J)-1)]

if ( !(bx = NAG_ALLOC (nreg, double))
    || !(covar = NAG_ALLOC ((npar+mn+isym+nreg) * (npar+mn+isym+nreg), dou-
ble))
    || !(et = NAG_ALLOC (num, double))
    || !(ht = NAG_ALLOC (num, double))
    || !(param = NAG_ALLOC (npar+mn+isym+nreg, double))
    || !(rvec = NAG_ALLOC (40, double))
    || !(sc = NAG_ALLOC (npar+mn+isym+nreg, double))
    || !(se = NAG_ALLOC (npar+mn+isym+nreg, double))
    || !(theta = NAG_ALLOC (npar+mn+isym+nreg, double))
    || !(x = NAG_ALLOC (num * nreg, double))
    || !(cvar = NAG_ALLOC (nt, double))
    || !(yt = NAG_ALLOC (num, double)))
{
    Vprintf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

Vprintf ("g13fac Example Program Results \n\n");
seed = 11;

mean = 3.0;

if (nreg > 0)
{
    for (i = 1; i <= num; ++i)
{
    fac1 = (double) i *0.01;
    X (i, 1) = sin (fac1) * 0.7 + 0.01;
    X (i, 2) = fac1 * 0.1 + 0.5;
}
    BX (1) = 1.5;
    BX (2) = 2.5;
}

PARAM (1) = 0.15;
PARAM (2) = 0.1;
PARAM (3) = 0.8;
PARAM (4) = 0.1;

fcall = Nag_Garch_Fcall_True;
g05cbc(seed);
num_startup = num;
g05hkc (num_startup, ip, iq, &PARAM (1), gamma, &HT (1), &YT (1),
        fcall, &RVEC (1), &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from g05hkc.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

fcall = Nag_Garch_Fcall_False;

```

```

g05hkc (num, ip, iq, &PARAM (1), gamma, &HT (1), &YT (1),
        fcall, &RVEC (1), &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from g05hkc.\n%s\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);
}

for (i = 1; i <= npar; ++i)
    THETA (i) = PARAM (i) * 0.5;

if (isym == 1)
    THETA (npar + isym) = gamma * 0.5;

if (mn == 1)
    THETA (npar + isym + 1) = mean * 0.5;

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

maxit = 50;
tol = 1e-12;
stat_opt = Nag_Garch_Stationary_True;
est_opt = Nag_Garch_Est_Initial_True;

g13fac (&YT (1), &X (1, 1), tdx, num, ip, iq, nreg, mn, isym,
        &THETA (1), &SE (1), &SC (1), &COVAR (1, 1), tdc, &hp,
        &ET (1), &HT (1), &lgf, stat_opt, est_opt, maxit, tol, &fail);

if (fail.code != NE_NOERROR)
{
    Vprintf("Error from g13fac.\n%s\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));

if (isym)
    Vprintf ("%20.4f          (%6.4f) %20.4f\n", THETA (npar+isym), SE (npar+
isym), gamma);
if (mn)

```



```

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

/* now forecast nt steps ahead */

if (isym)
{
    gamma = 0.0;
}
else
{
    gamma = THETA(npar+isym);
}

g13fbc(num,nt,ip,iq,&THETA(1),gamma,&CVAR(1),&HT(1),&ET(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 (et) NAG_FREE (et);
if (ht) NAG_FREE (ht);
if (param) NAG_FREE (param);
if (rvec) NAG_FREE (rvec);
if (sc) NAG_FREE (sc);
if (se) NAG_FREE (se);
if (theta) NAG_FREE (theta);
if (x) NAG_FREE (x);
if (cvar) NAG_FREE (cvar);
if (yt) NAG_FREE (yt);

return exit_status;
}

```

8.2 Program Data

None.

8.3 Program Results

g13fac Example Program Results

Parameter estimates	Standard errors	Correct values
0.0902	(0.0361)	0.1500
0.1030	(0.0253)	0.1000
0.8433	(0.0390)	0.8000
-0.1509	(0.1836)	-0.3000
3.0840	(0.1395)	3.0000
1.4989	(0.0790)	1.5000
2.4402	(0.1354)	2.5000

3 step forecast = 1.5355