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)).

### Specification
```c
#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)
```

### Description
When \(\text{isym} = 0\), \(\text{nag Estimate AgarchI (g13fac)}\) models a standard \((\gamma = 0)\) univariate regression-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 \(i = 1, 2, \ldots, k\), which can be represented by:

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

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

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

When \(\text{isym} = 1\), \(\text{nag Estimate AgarchI (g13fac)}\) models an asymmetric GARCH\((p,q)\) process where the conditional variance \(h_t\) is given by:

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

Here \(T\) 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, \(\gamma\) the asymmetry parameter, \(h_t\) the conditional variance, and \(\psi_t\) the information set of all information up to time \(t\).

When \(\text{isym} = 1\), \(\text{nag Estimate AgarchI (g13fac)}\) 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 \(b^T = (b_1, \ldots, b_k)\).

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

- **No Regression or Mean**
  
  \[
y_t = \epsilon_t,
  \]

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

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

  \(\text{nreg} = 0\), and
\( \theta \) is a \((p + q + 1) \times 1\) vector.

**No Regression**
\[
y_t = b_0 + \epsilon_t, \quad \text{isym} = 0, \quad \text{mn} = 1, \quad \text{nreg} = 0, \quad \text{and} \quad \theta \text{ is a } (p + q + 2) \times 1 \text{ vector.}
\]

**Note:** if the \( y_t = \mu + \epsilon_t \), where \( \mu \) is known (not to be estimated by nag_estimate_agarchI (g13fac)) then (1) can be written as \( y'_t = \epsilon_t \), where \( y'_t = y_t - \mu \). This corresponds to the case **No Regression or Mean**, with \( y_t \) replaced by \( y_t - \mu \).

**No Mean**
\[
y_t = x_t^T b + \epsilon_t, \quad \text{isym} = 0, \quad \text{mn} = 0, \quad \text{nreg} = k \quad \text{and} \quad \theta \text{ is a } (p + q + k + 1) \times 1 \text{ vector.}
\]

### 4 References


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

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

2. \( x[t:\text{num} \times \text{tdx}] \) – const double
   
   *On entry:* the \( i \)th element of the \( j \)th vector \( X \) is stored in \( x[(i - 1) \times \text{tdx} + j - 1] \).

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

   *Constraint:* \( \text{tdx} \geq \text{nreg} \).
4:  **num** – Integer

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

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

5:  **p** – Integer

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

   Constraint: \( p \geq 0 \).

6:  **q** – Integer

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

   Constraint: \( q \geq 1 \).

7:  **nreg** – Integer

   On entry: \( k \), the number of regression coefficients.

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

8:  **mn** – Integer

   On entry: if \( \text{mn} = 1 \), the mean term \( b_0 \) will be included in the model.

   Constraint: \( \text{mn} = 0 \) or 1.

9:  **isym** – Integer

   On entry: if \( \text{isym} = 1 \), the asymmetry term \( \gamma \) will be included in the model.

   Constraint: \( \text{isym} = 0 \) or 1.

10: **theta**\([\text{npar}]\) – double

    Input/Output

    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 \( \text{est\_opt} = \text{Nag\_Garch\_Est\_Initial\_False} \) (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 \( i = 1,2,\ldots,k \).

    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 \( b_o \).

    The final \( \text{nreg} \) elements are the estimated linear regression coefficients \( b_i \), for \( i = 1,2,\ldots,k \).

11: **se**\([\text{npar}]\) – double

    Output

    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 $j = 1, 2, \ldots, p$.
If $\text{isym} = 1$, the next element contains the standard error for $\gamma$.
If $\text{mn} = 1$, the next element contains the standard error for $b_o$.
The final $nreg$ elements are the standard errors for $b_j$, for $j = 1, 2, \ldots, k$.

12: $\mathbf{sc}[\text{npars}]$ – double
\text{Output}
\text{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 $j = 1, 2, \ldots, p$.
If $\text{isym} = 1$, the next element contains the score for $\gamma$.
If $\text{mn} = 1$, the next element contains the score for $b_o$.
The final $nreg$ elements are the scores for $b_j$, for $j = 1, 2, \ldots, k$.

13: $\mathbf{covar}[\text{npars} \times \text{tdc}]$ – double
\text{Output}
\text{Note:} the $(i, j)$th element of the matrix is stored in $\mathbf{covar}[(i-1) \times \text{tdc} + j - 1]$.
\text{On exit:} the covariance matrix of the parameter estimates $\hat{\theta}$, that is the inverse of the Fisher Information Matrix.

14: $\text{tdc}$ – Integer
\text{Input}
\text{On entry:} the stride separating matrix column elements in the array $\mathbf{covar}$.
\text{Constraint:} $\text{tdc} \geq \text{npars}$.

15: $\text{hp}$ – double *
\text{Input/Output}
\text{On entry:} if $\text{est\_opt} = \text{Nag\_Garch\_Est\_Initial\_False}$, $\mathbf{hp}$ is the value to be used for the pre-observed conditional variance.
If $\text{est\_opt} = \text{Nag\_Garch\_Est\_Initial\_True}$, $\mathbf{hp}$ is not referenced.
\text{On exit:} if $\text{est\_opt} = \text{Nag\_Garch\_Est\_Initial\_True}$, $\mathbf{hp}$ is the estimated value of the pre-observed of the conditional variance.

16: $\mathbf{et}[\text{num}]$ – double
\text{Output}
\text{On exit:} the estimated residuals, $\epsilon_t$, for $t = 1, 2, \ldots, T$.

17: $\mathbf{ht}[\text{num}]$ – double
\text{Output}
\text{On exit:} the estimated conditional variances, $h_t$, for $t = 1, 2, \ldots, T$.

18: $\text{lgf}$ – double *
\text{Output}
\text{On exit:} the value of the log likelihood function at $\hat{\theta}$.

19: $\text{stat\_opt}$ – Nag\_Garch\_Stationary\_Type
\text{Input}
\text{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.
\text{Constraint:} $\text{stat\_opt} = \text{Nag\_Garch\_Stationary\_True}$ or $\text{Nag\_Garch\_Stationary\_False}$.
20:  estopt – Nag_Garch_Est_Initial_Type

On entry: if estopt = Nag_Garch_Est_Initial_True, the function provides initial parameter estimates of the regression terms \(b_0, b_T\).

If estopt = Nag_Garch_Est_Initial_False, you must supply the initial estimations of the regression parameters \(b_0, b_T\).

Constraint: estopt = Nag_Garch_Est_Initial_True or Nag_Garch_Est_Initial_False.

21:  max_iter – Integer

On entry: the maximum number of iterations to be used by the optimization function when estimating the GARCH\(p,q\) parameters. 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 \(\geq 0\).

22:  tol – double

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

23:  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 = \langle value\rangle while 1 + q + p + isym + mn + nreg = \langle value\rangle. These parameters must satisfy num \(\geq 1 + q + p + isym + mn + nreg\).

On entry, tdc = \langle value\rangle while 1 + q + p + isym + mn + nreg = \langle value\rangle. These parameters must satisfy tdc \(\geq 1 + q + p + isym + mn + nreg\).

On entry, tdx = \langle value\rangle while nreg = \langle value\rangle. These parameters must satisfy tdx \(\geq nreg\).

NE_ALLOC_FAIL
Dynamic memory allocation failed.

NE_BAD_PARAM
On entry, argument estopt had an illegal value.
On entry, argument statopt 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: nreg \(\geq 0\).
On entry, p = \langle value\rangle.
Constraint: p \(\geq 0\).
On entry, q = \langle value\rangle.
Constraint: q \(\geq 1\).
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.

Value \( \text{value} \) given to \text{isym} is not valid. Correct range is 0 to 1.

Value \( \text{value} \) given to \text{mn} is not valid. Correct range is 0 to 1.

Matrix \( X \) does not give a model of full rank.

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.

Not applicable.

Not applicable.

None.

This example program illustrates the use of \text{nag_estimate_agarchI} (g13fac) to model a GARCH(1,1) sequence generated by \text{nag_rand_agarchI} (g05pdc), a three step forecast is then calculated using \text{nag_forecast_agarchI} (g13fbc).

/* \text{nag_estimate_agarchI} (g13fac) Example Program. */
* Copyright 2014 Numerical Algorithms Group.
* Mark 6, 2000.
* Mark 8 revised, 2004
* */
#include <nag.h>
#include <nag_stdlib.h>
#include <stdio.h>
#include <ctype.h>
#include <math.h>
#include <nag05.h>
#include <naggl3.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 *covar = 0, *cvar = 0, *et = 0, *ht = 0, *r = 0;
double *sc = 0, *se = 0, *theta = 0, *x = 0, *yt = 0;
double fac1, hp, lgf, xterm;

/* 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[] = { 0.15, 0.1, 0.8, 0.1 };

/* Asymmetry parameter */
double gamma = -0.3;

/* Regression parameters */
Integer nreg = 2;
double mean = 3.0;
double bx[] = { 1.5, 2.5 };

/* When fitting a model to the time series ... */
/* Include asymmetry parameter in the model */
Integer isym = 1;

/* 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 */

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

printf("nag_estimate_agarchI (g13fac) Example Program Results \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;
tdc = npar + mn + isym + nreg;
tdx = nreg;

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

/* Allocate arrays */

if (!(covar = NAG_ALLOC((npar + mn + isym + nreg) * tdc, double)) ||
    !(et = NAG_ALLOC(num, double)) ||
    !(ht = NAG_ALLOC(num, 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)) ||
    !(state = NAG_ALLOC(lstate, Integer)) ||
    !(r = NAG_ALLOC(lr, 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;
}

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

/* Generate a realization of a random AGARCH I time series and discard it */

fcall = Nag_TRUE;
nag_rand_agarchI(dist, num, ip, iq, param, gamma, df, ht, yt, fcall, r, lr,
                          state, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_rand_agarchI (g05pdc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Generate a realization of a random AGARCH I time series to use */

fcall = Nag_FALSE;
nag_rand_agarchI(dist, num, ip, iq, param, gamma, df, ht, yt, fcall, r, lr,
                          state, &fail);
if (fail.code != NE_NOERROR)
printf("Error from nag_rand_agarchI (g05pdc).\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;
if (isym == 1)
theta[npar + isym - 1] = gamma * 0.5;
if (mn == 1)
theta[npar + isym] = mean * 0.5;
for (i = 0; i < nreg; ++i)
theta[npar + isym + mn + i] = bx[i] * 0.5;

} /* nag_estimate_agarchI (gl3fac).
* Univariate time series, parameter estimation for either a
* symmetric GARCH process or a GARCH process with asymmetry
* of the form (epsilon_(t-1)+gamma)^2
*/
ag_estimate_agarchI(yt, x, tdx, num, ip, iq, nreg, mn, isym, 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_agarchI (gl3fac).\n%s\n",
fail.message);
exit_status = 1;
goto END;
}

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

} /* Now forecast nt steps ahead */
if (isym)
{
    gamma = theta[npar + isym - 1];
}
else

Mark 25


{  
gamma = 0.0;  
}

/* nag_forecast_agarchI (g13fbc).
* Univariate time series, forecast function for either a
* symmetric GARCH process or a GARCH process with asymmetry
* of the form (epsilon_(t-1)+gamma)^2
*/
nag_forecast_agarchI(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(sc);
NAG_FREE(se);
NAG_FREE(theta);
NAG_FREE(x);
NAG_FREE(cvar);
NAG_FREE(yt);
NAG_FREE(state);
NAG_FREE(r);

return exit_status;
}

10.2 Program Data

None.

10.3 Program Results

nag_estimate_agarchI (g13fac) Example Program Results

<table>
<thead>
<tr>
<th>Parameter estimates</th>
<th>Standard errors</th>
<th>Correct values</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1045</td>
<td>(0.0486)</td>
<td>0.1500</td>
</tr>
<tr>
<td>0.0884</td>
<td>(0.0235)</td>
<td>0.1000</td>
</tr>
<tr>
<td>0.8196</td>
<td>(0.0433)</td>
<td>0.8000</td>
</tr>
<tr>
<td>-0.7658</td>
<td>(0.2296)</td>
<td>-0.3000</td>
</tr>
<tr>
<td>3.0051</td>
<td>(0.1309)</td>
<td>3.0000</td>
</tr>
<tr>
<td>1.4478</td>
<td>(0.0779)</td>
<td>1.5000</td>
</tr>
<tr>
<td>2.4681</td>
<td>(0.1229)</td>
<td>2.5000</td>
</tr>
</tbody>
</table>

3 step forecast = 1.1019