A Tutorial for G@RCH 2.3, a Complete Ox Package for Estimating and Forecasting ARCH Models

Sébastien Laurent\textsuperscript{a,\textbf{b}} and Jean-Philippe Peters\textsuperscript{a}

\textsuperscript{a} Département d’Économie, de Gestion et de Sciences Sociales, Université de Liège, Belgium

\textsuperscript{b} Department of Quantitative Economics, Maastricht University, Netherlands.

E-mails: S.Laurent@ulg.ac.be and jp.peters@ulg.ac.be

April 26, 2002

Abstract

This tutorial documents G@RCH 2.3, an Ox package dedicated to the estimation and forecast of various univariate ARCH-type models in the conditional variance and an AR(FI)MA specification in the conditional mean. These ARCH processes include ARCH, GARCH, EGARCH, GJR, APARCH, IGARCH, FIGARCH, FIEGARCH, FIAPARCH and HYGARCH.

These models can be estimated by Approximate (Quasi-) Maximum Likelihood under four assumptions: normal, Student-$t$, GED or skewed Student-$t$ errors. Explanatory variables can enter both the conditional mean and the conditional variance equations. One-step-ahead (density) forecasts of both the conditional mean and variance are available as well as some miss-specification tests and several graphical features.

After a brief introduction of the package, we present the G@RCH class member functions. We then propose a theoretical overview of all the specifications both in the conditional mean and the conditional variance where the usefulness of asymmetric and fractionally integrated models is highlighted. Next, further explanations are given about the estimation methods. A concrete application of G@RCH 2.3 is then provided with the daily CAC40 French index. Numerous print-screens are given with this application so that this document can be considered as an user guide. That way, the interface of the program is illustrated and its ease of use is highlighted. Finally, a note on the history of the releases of the G@RCH package as well as a brief list of possible future improvements of the software is given.
Contents

1 Introduction ........................................... 1
   1.1 The G@RCH Package ........................................... 1
       1.1.1 Definition ........................................... 1
       1.1.2 Program Versions ........................................... 2
   1.2 Disclaimer ........................................... 2
   1.3 Availability and Citation ........................................... 2
   1.4 Installing and Running G@RCH 2.3 ........................................... 3
   1.5 What’s new in this version ? ........................................... 3

2 Structure of the Program ........................................... 4
   2.1 Classes and Functions ........................................... 4
   2.2 G@RCH Member Functions List ........................................... 4
   2.3 G@RCH Members Functions ........................................... 8

3 Features of the package ........................................... 27
   3.1 Mean equation ........................................... 27
   3.2 Variance equation ........................................... 28
       3.2.1 ARCH Model ........................................... 28
       3.2.2 GARCH Model ........................................... 29
       3.2.3 EGARCH Model ........................................... 30
       3.2.4 GJR Model ........................................... 31
       3.2.5 APARCH Model ........................................... 31
       3.2.6 IGARCH Model ........................................... 32
       3.2.7 Fractionally Integrated Models ........................................... 33

4 Estimation Methods ........................................... 36
   4.1 Parameters Constraints ........................................... 36
   4.2 Distributions ........................................... 36
   4.3 Tests ........................................... 38
   4.4 Forecasts ........................................... 39
       4.4.1 Forecasting the conditional mean ........................................... 39
       4.4.2 Forecasting the conditional variance ........................................... 40
   4.5 Accuracy ........................................... 42
   4.6 Features Comparison ........................................... 44
5 Application

5.1 Data and Methodology ............................................. 47
5.2 Using the “Full Version” ........................................... 47
5.3 Using the “Light Version” ......................................... 51

6 Versions and Future Improvements

6.1 Releases History ..................................................... 57
6.2 Future Improvements .............................................. 57
1 Introduction

Well known statistical packages such as Eviews 4.0, Rats 5.0, Microfit 4.0, Matlab 12 or S-Plus 6.0 provide various options to estimate sophisticated econometric models, in completely different areas such as cointegration, panel data, etc.

This paper documents G@RCH 2.3, an Ox package dedicated to the estimation and forecast of various univariate models. Contrary to the softwares mentioned above, G@RCH 2.3 is only concerned with ARCH-type models (Engle, 1982), including some of the most recent contributions in this field: GARCH (Bollerslev, 1986), EGARCH (Nelson, 1991), GJR (Glosten, Jagannathan, and Runkle, 1993), APARCH (Ding, Granger, and Engle, 1993), IGARCH (Engle and Bollerslev, 1986), FIGARCH (Baillie, Bollerslev, and Mikkelsen, 1996a and Chung, 1999), FIEGARCH (Bollerslev and Mikkelsen, 1996), FIAPARCH (Tse, 1998) and HYGARCH (Davidson, 2001) specifications of the conditional variance. Moreover an AR(FI)MA process can be specified in the conditional mean (see Baillie, Chung, and Tieslau, 1996, Tschernig, 1995, Teyssiële, 1997, or Lecourt, 2000, for further details about ARFIMA models).

Our package has been developed with the Ox 3.10 matrix programming language of Doornik (2001). G@RCH 2.3 should be compatible with a lot of platforms, including Windows, Linux, Unix and Solaris. For most of the specifications, it is generally very fast and one of its main characteristic is its ease of use.

This tutorial is structured as follows: the structure and the functions of the package are detailed in Section 2 while we propose an overview of the package features in Section 3, with the presentation of the models (in the mean and in the variance). Comments over estimation procedures (parameters constraints, distributions, standard deviation estimation methods, tests, forecasting procedures and accuracy of the package) are introduced in Section 4. Then a user guide is provided for both versions of G@RCH 2.3 in Section 5 with an application using the CAC40 French index. Section 6 proposes an history of the releases of G@RCH and future improvements of the package.

1.1 The G@RCH Package

1.1.1 Definition

G@RCH 2.3 is an Ox package dedicated to the estimation and the forecasting of GARCH models and many of its extensions. It can be used via OxPack (with a dialog-oriented interface) or via the classic programming way.


For a comprehensive review of this language, see Cribari-Neto and Zarkos (2001).
1993), IGARCH (Engle and Bollerslev, 1986), FIGARCH (Baillie, Bollerslev, and Mikkelsen, 1996 and Chung, 1999), FIEGARCH (Bollerslev and Mikkelsen, 1996), FIAPARCH (Tse, 1998) and HYGRACH (Davidson, 2001). This package provides a lot of features unavailable in most traditional econometric softwares, including various model specifications, two standard errors estimation methods (Approximate Maximum Likelihood and Approximate Quasi-Maximum Likelihood) and four distributions (normal, Student-t, GED or skewed Student-t). Moreover, explanatory variables can enter the mean and/or the variance equations. Finally, one-step-ahead (density) forecasts of both the conditional mean and variance are available as well as many miss-specification tests (Nyblom, SBT, Pearson goodness-of-fit, Box-Pierce,...).

1.1.2 Program Versions

Two versions of our program are available and called the “Light Version” and the “Full Version”.

The “Light Version” can be launched from the Ox file GarchEstim.ox. This version requires some experience with the program and its structure. The “Light Version” is therefore dedicated to advanced users. This is also dedicated to users who do not possess the GiveWin software. Indeed, since the OxPack module is only available for registered Ox users, users who have a free (console) Ox version cannot use our dialogs-oriented (or “Full”) version. Hence, the “Light Version” is the solution, since its use just requires an Ox executable and a text editor.

The “Full Version” provides the same features as the other version, but also offers a friendly interface and some graphical features. This version needs to be launched from OxPack.

1.2 Disclaimer

This package is functional but no warranty is given whatsoever. The most appropriate way to discuss about problems and issues of the G@RCH package is the Ox-users forum (go to http://www.mailbase.ac.uk/lists/ox-users for registration and archives). Suggestions, mistakes, typos and possible improvements can be reported to the authors via e-mail: S.Laurent@ulg.ac.be for Sébastien and jp.peters@ulg.ac.be for Jean-Philippe.

1.3 Availability and Citation

The G@RCH package is available for downloading at the following address:

http://www.egss.ulg.ac.be/garch

This package is free of charge for academic and research purposes. For a commercial use of the program, please contact the authors. Moreover, for easier validation and replication of empirical findings, please cite this documentation (or Laurent and Peters, 2002) in all reports and publications involving the use of G@RCH 2.3.
1.4 Installing and Running G@RCH 2.3

To run G@RCH 2.3, unzip the garch23.zip (or garch23.tut.zip) file in the ox/packages directory (using folders names). A new Garch23 folder should be automatically created. The files contained in this garch23.zip file are listed in readme.txt.

If you want to use the "Light Version", you need garch.oxo and garch.h. The example of section 5 is based on GarchEstim.ox. To run it, type

```
oxl GarchEstim
```

at the command prompt. Alternatively, you may use oxrun. OxRun can easily be launched from OxEdit, a free text editor2. GiveWin is needed to display the graphics but is not mandatory for running the estimation.

To run the "Full Version" of the program, garch.oxo and garch.h must also be installed. Moreover, you have to possess a registered version of Ox 3.10 Professional, and you need to be sure that GiveWin and OxPack are correctly installed on your computer. See Section 5.2 for more details.

1.5 What’s new in this version?

- Ox 3.10 compatible.
- Improved OxPack interface.
- Allows to run tests on the raw series from OxPack.
- The Hyperbolic GARCH (HYGARCH) of Davidson (2001) is available.
- Improved forecasts, with new graphical options.
- Stationarity and positivity constraints are checked for most the models.
- All the GARCH models and all the tests can be called from external code. See Section 2.3 for further details.
- Possibility to fix some parameters at any defined value. It allows thus to estimate, for instance, an AR(2) model with only lag 2 estimated (by fixing the AR(1) parameter to 0).
- Possibility to save graphics when using the "Light Version" without GiveWin.
- Several minor bugs corrected.

---

2 You can download OxEdit at http://www.oxedit.com
2 Structure of the Program

2.1 Classes and Functions

Ox provides support for object-oriented programming. An interesting concept is therefore the “Classes”. Indeed, one can create new classes based on other existing parent-classes and use the functions of these parents, therefore avoiding to rewrite these procedures for derived classes. In our case, the Garch class is defined as a Modelbase type of class. This Modelbase class derives itself from the Database class.

The Database class is dedicated to the handling of the database, the sample, the names of the variables, the selection of the variables... The Modelbase class implements model estimation features. It is not intended to be used directly but as a base for a more specialized class, such as our Garch class or already available classes such as ARFIMA, DPD (Panel Data estimation), SVPack (Stochastic Volatility models) or SsfPack (State space forms).

See Doornik (2001) for more details about the notion of “Classes”.

2.2 G@RCH Member Functions List

Here is the list of the Garch member functions and a brief description for each of them. Our program also uses functions from other classes (Modelbase and Database). New and significantly modified functions for this version are indicated with the † symbol.

**Constructor**

Garch Constructor

**Model Formulation (used in the “Light Version”)**

- ARCHLAGS: Specifies the desired lags for the Engle’s LM test for ARCH
- ARFIMA: Specifies if ARFIMA is wanted in the mean
- ARMA_ORDERS: Specifies the AR and MA orders in the mean
- BOUNDS: Specifies if estimated parameters are bounded with the low and up bounds entered in startingvalues.txt
- BOXPIERCE: Specifies the desired lags for the Box-Pierce test
- COVAR: Specifies if the Variance-Covariance matrix of the estimated parameters is printed in the output
- CSTS: Specifies if constants are wanted in the mean and in the variance
- DISTRI: Specifies the desired distribution
- FIXPARAM(†): Allows to fix some parameters to their starting values.
- FOREGRAPHS: Plots and saves various forecast-related graphs in GiveWin
- FORECASTS: Specifies if forecasts are wanted and the number of forecasts
GARCH_ORDERS Specifies the \( p \) and \( q \) orders of the GARCH\((p, q)\)

GRAPHS Plots and saves various estimation related graphs in GiveWin

ITER Specifies the number of iterations between prints of intermediary results

MLE Specifies the estimation method of the standard errors

MODEL(†) Specifies the GARCH-type of models in the conditional variance.

NYBLOM Specifies if the Nyblom stability test is wanted

PEARSON Specifies the desired lags for the adjusted Pearson goodness-of-fit test

SAVEPAR Saves the parameters estimates and their standard errors in an Excel spreadsheet

STORE Allows storing estimated \( \varepsilon_t \), \( \varepsilon_t^2 \) and \( \sigma_t^2 \) series

TESTS(†) Allows to run tests either on raw data (prior to estimation) or on the estimated series.

TRUNC(†) Truncation order for the F.I. models using the method of BBM (96).

Initialization

CheckPara Check initial values

FixBounds Fixes the values of the lower and upper bounds of the estimated parameters

InitData Initializes the characteristics of the model (sample, regressors...)

InitStartValues Initializes the starting values of the parameters to estimate

Parameters related functions

Dialogs Parameter starting values (only “Full Version”)

GetPara Constructs the parameters vector

PAR(†) Creates a matrix with the parameters estimates, their standard errors and their robust s.e.

SplitPara Allocates the value of each element of the parameters vector to the correct variable

Transform Computes the transformation of Equation 28

Filters

AParch APARCH filter

EGarch EGARCH filter

Garch_Filter GARCH filter

GJR_Filter GJR filter

FIEGarch FIEGARCH filter (the Fractional Integration process uses Chung’s method)

Figarch_BBMM FIGARCH filter with the Baillie et al. (1996) method (BBM)

Figarch_Chung FIGARCH filter with the Chung (1999) method

Distributions Related

CD Computes the Cumulative Distribution Function of the Gaussian distribution
CDFGED Computes the Cumulative Distribution Function of the GED
CDFTA Computes the Cumulative Distribution Function of the (skewed) Student-t distribution
GaussLik Computes the log-likelihood for the Gaussian distribution
GEDLik Computes the log-likelihood for the GED
INVCDFGED Computes the Inverse CDF of the GED
INVCDTFA Computes the Inverse CDF of the (skewed) Student-t distribution
KiAparch Computes the stationary condition of Equation 18
SkStudentLik Computes the log-likelihood for the skewed Student distribution
StudentLik Computes the log-likelihood for the Student distribution

**Forecasting**

Fig.FOR Allocates the right forecasts filter depending on the specification
FOR_APARCH(†) Forecasts filter of the APARCH process
FOR_ARFIMA Forecasts filter of the ARFIMA process
FOR_ARMA Forecasts filter of the ARMA process
FOR_EGARCH(†) Forecasts filter of the EGARCH process
FOR_FIAPARCH_BBMS(†) Forecasts filter of the FIAPARCH process (BBM method)
FOR_FIAPARCH_Chung(†) Forecasts filter of the FIAPARCH process (Chung method)
FOR_FIEGARCH(†) Forecasts filter of the FIEGARCH process.
FOR_FIGARCH_BBMS(†) Forecasts filter of the FIGARCH process (BBM method)
FOR_FIGARCH_Chung(†) Forecasts filter of the FIGARCH process (Chung method)
FOR_GARCH Forecasts filter of the GARCH process
FOR_GJR Forecasts filter of the GJR process
FOR_GRAPHSC(†) Draws the forecasts graphics
FORECASTING(†) Launches the one-step ahead forecast process

**General**

GetCovar Gets the covariance matrix of the estimated parameters
GetT(†) Gets the number of observations
GetForeData Gets all the post-estimation data (if any)
GetForErrors Gets the forecasts errors
GetINames Gets the name of the realized volatility series (if it exists)
GetNbPar(†) Gets the number of parameters
GetParNames Gets the names of the parameters
GetSeries Returns a matrix with the Y series, the mean residuals and the conditional variance
GetXB Gets $\mu_t$ of Equation 2
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetXBetaForc</td>
<td>Gets the post-estimation values of the regressor(s) in the mean (if any)</td>
</tr>
<tr>
<td>GetXNames</td>
<td>Gets the names of the regressors in the mean equation</td>
</tr>
<tr>
<td>GetYNames(†)</td>
<td>Gets the name of the dependent variable</td>
</tr>
<tr>
<td>GetZB</td>
<td>Gets $\alpha_0$ of Section 3.2.1</td>
</tr>
<tr>
<td>GetZBetaForc</td>
<td>Gets the post-estimation values of the regressor(s) in the variance (if any)</td>
</tr>
<tr>
<td>GetZNames</td>
<td>Gets the names of the regressors in the variance equation</td>
</tr>
</tbody>
</table>

**Model Estimation**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DoEstimation</td>
<td>Estimates the model (“Light version”)</td>
</tr>
<tr>
<td>Estimate</td>
<td>Estimates the model (“Full version”)</td>
</tr>
<tr>
<td>FigLL</td>
<td>This is the function optimized by BFGS</td>
</tr>
<tr>
<td>Filter</td>
<td>Allocates the filter number depending on the model specification</td>
</tr>
<tr>
<td>GetRes</td>
<td>Gets the mean residuals</td>
</tr>
<tr>
<td>ResVar(†)</td>
<td>Gets the variance residuals</td>
</tr>
<tr>
<td>ScoreContributions</td>
<td>Computes the numerical derivatives</td>
</tr>
</tbody>
</table>

**Post Estimation**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>absha</td>
<td>Computes frequencies in an interval defined by upper and lower bounds</td>
</tr>
<tr>
<td>APGT</td>
<td>Computes adjusted Pearson Chi-square Goodness-of-fit test (Vlaar and Palm, 1993)</td>
</tr>
<tr>
<td>ArchTest</td>
<td>Computes and prints the Engle’s LM ARCH test (Engle, 1982)</td>
</tr>
<tr>
<td>AUTO</td>
<td>Computes and plots the autocorrelation functions</td>
</tr>
<tr>
<td>BoxPQ</td>
<td>Computes and prints the modified Box-Pierce Q-statistics and the associated p-values</td>
</tr>
<tr>
<td>confidence_limits</td>
<td>Computes the confidence bounds of a confidence interval from the vector of assumed uniform 0-1 “z series”</td>
</tr>
<tr>
<td>FEM(†)</td>
<td>Computes and prints 10 forecasts errors measures</td>
</tr>
<tr>
<td>ICriterion</td>
<td>Computes the four Information Criteria (Akaike, Hannan-Quinn, Schwarz and Shibata)</td>
</tr>
<tr>
<td>Normality</td>
<td>Computes the skewness, kurtosis and Jarque and Bera (1987) test,</td>
</tr>
<tr>
<td></td>
<td>with associated t-test and p-values</td>
</tr>
<tr>
<td>MLEMeth</td>
<td>Prints the estimated parameters, their standard deviations, t-tests and p-values</td>
</tr>
<tr>
<td>MZ(†)</td>
<td>Computes and prints the Mincer and Zarnowitz (1969) regression for the conditional variance</td>
</tr>
<tr>
<td>Nyblom</td>
<td>Computes and prints the Nyblom (1989) stability test</td>
</tr>
<tr>
<td>Output</td>
<td>Prints the model specification and launches other post-estimation procedures</td>
</tr>
<tr>
<td>Positivity(†)</td>
<td>Checks the positivity constraints with the estimated parameters</td>
</tr>
<tr>
<td>QuantileGraphics</td>
<td>Plots the quantiles of a distribution</td>
</tr>
<tr>
<td>SBT</td>
<td>Computes the sign bias test, the negative size bias test, the positive size bias test</td>
</tr>
</tbody>
</table>
and a joint test of the three
Stationarity(†) Checks the stationarity constraints with the estimated parameters
Tests Launches the selected tests and prints their results
TestGraphicAnalysis Draws the graphics of the estimation

2.3 G@RCH Members Functions

Garch::AParch, Garch::EGarch, Garch::Garch_Filtet, Garch::GJR_Filtet, Garch::Figarch_BBt, Garch::Figarch_Chung, Garch::FIEGarch, Garch::FIAParch,

AParch (const e, const level, const p, const q, const par) ;
EGarch (const e, const level, const p, const q, const par, const dist) ;
Garch_Filtet (const e, const level, const p, const q, const par) ;
GJR_Filtet (const e, const level, const p, const q, const par) ;
Figarch_BBt (const e, const level, const p, const q, const laglamb, const par) ;
Figarch_Chung (const e, const level, const p, const q, const par);
FIEGarch (const e, const level, const p, const q, const laglamb, const par, const dist) ;
FIAParch (const e, const level, const p, const q);

In: (m_cT x 1) matrix, residuals series (from the mean).
level in: (m_cT x 1) or (m_cT x m_cX) matrix, constant + independent variables.
p in: integer, GARCH order
q in: integer, ARCH order
laglamb in: integer, truncation order (BBM method)
par in: (# param x 1) matrix, parameters values (size of the vector depends on the model).
dist in: integer, distribution used (0: Normal, 1:Student-t, 2: GED, 3: Skewed Student-t)

Return value
A (m_cT x 1) matrix with the estimated conditional variance (σ²).

Description
These are the filters of the different models. Recall that two methods are available for the FIGARCH models: the Baillie, Bollerslev, and Mikkelsen (1996) method (BBM) that includes a truncation order or the Tse (1998) method that does not. Moreover, the HYGARCH model is launched with the FIGARCH function: the last element of par is ln(α). If it is equal to 0, then we have the traditional FIGARCH model. Note also that level is a (m_cT x 1) vector of ones if there is no independent variable and a (m_cT x m_cXV) matrix equals to b′X if there are explanatory variables in the variance equation. Finally, the arguments of the functions are sufficient to call them from an external Ox code. Indeed no class member is used in these procedures anymore.
Here is an example of an Ox code that creates randomly a series with 1000 observations, runs a GARCH(1,1), a GJR(1,1) and an APARCH(1,1) models on it (all with a normal distribution) and then displays the graphs of the estimated conditional variance series (in GiveWin):

```ox
decl garchobj; decl p, q, alpha, beta, apa, gjr ;
dcl parameters,res, level ;
dcl garch, gjr, aparch;
garchobj = new Garch();
res = rann(1000,1) ;
level = ones(1000,1);
p = 1 ;
q = 1 ;
alpha = 0.05 ;
beta = 0.90 ;
gjr = -0.15 ;
apa = <0.15;1.5> ;
parameters = alpha|beta ;
garch = garchobj.Garch_Filter(res, level, p, q, parameters) ;
gjr = garchobj.GJR_Filter(res, level, p, q, parameters|gjr) ;
aparch = garchobj.AParch(res, level, p, q, parameters|apa) ;
Draw(0, garch');
DrawTitle(0, "Garch");
Draw(1, gjr');
DrawTitle(1, "GJR");
Draw(2, figarch');
DrawTitle(2, "Aparch");
ShowDrawWindow() ;
delete garchobj ;
```

The elements of the `par` argument have to be entered in a specific order. See the `GetPara` function for more details on the order.

**Garch::APGT**

`APGT(const cd, const ng, const np)`;

- `cd` in: \((m \times T) \times 1\) matrix, values of the cumulative distribution function.
- `ng` in: integer, number of classification groups
- `np` in: integer, number of parameters

_Return value_

1 if the test is successfully run, 0 otherwise.

_Description_

Computes and prints adjusted Pearson \(\chi^2\) goodness-of-fit test (Vlaar and Palm, 1993). See Section 4.3 for more detail about this test.

**Garch::ARCHLAGS**

`ARCHLAGS(const lags)`;

_No return value_

_Description_

Fix the lags wanted when computing Engle’s LM test for ARCH processes. By default, `lags` is \(< 2; 5; 10 >\). This means that three Engle’s LM tests for ARCH are computed, with lags 2, 5 and 10. If `lags` is `<>`, the test will not be reported.
Garch::ARMA_ORDERS, Garch::GARCH_ORDERS
ARMA_ORDERS(const cAR, const cMA) ;
GARCH_ORDERS(const cP, const cQ) ;

cAR    in: integer, AR order, p
cMA    in: integer, MA order, q
cP     in: integer, GARCH order, p
cQ     in: integer, ARCH order, q

No return value
Description
   Fixes the ARMA and GARCH orders.

Garch::AUTO
AUTO(const z, const ncor, const min, const max, const plot) ;

z      in: \((n \times 1)\) matrix, series to be tested
ncor   in: integer, maximum lag of the autocorrelation
min, max in: integer, coordinates of the Y-axis (for the 4 graphs)
plot   in: integer, area wherein the first graph is plotted

No return value
Description
   Computes and plots the autocorrelations (with maximum lag = ncor).

Garch::BOUNDS
BOUNDS(const method) ;

method in: 1 if bounded parameters wanted, 0 otherwise.

No return value
Description
   If method is 1, the estimated parameters are bounded between (editable) lower and upper
   values defined in startingvalues.txt. If it is equal to 0, the parameters are not bounded.

Garch::BOXPIERCE
BOXPIERCE(const lags) ;

No return value
Description
   Fixes the lags wanted when computing Box-Pierce statistics. By default, lags is \(< 5; 10; 20 >\).
   This means that BP(5), BP(10) and BP(20) are computed for the standardized residuals and
   squared standardized residuals. If lags is \(< >\), the test will not be reported.

Garch::BoxPQ
BoxPQ(const eh, const e2h, const ncor, const adjM, const adjV, const sq) ;
eh in: \((m \times T \times 1)\) matrix, series to be tested.
e2h in: \((m \times T \times 1)\) matrix, squared series.
ncor in: \((m \times T \times 1)\) matrix, vector of lags.
adjM in: integer, number of degrees of freedom to be subtracted for eh
adjV in: integer, number of degrees of freedom to be subtracted for e2h
sq in: if 0, test on the residuals, if 1, test on the squared residuals

**Return value**
1 if the test is successfully run, 0 otherwise.

**Description**
Computes and prints Box-Pierce Q-statistics on standardized residuals and/or squared standardized residuals. See Section 4.3 for more details about this test.

**Garch::CD**

```c
CD (const e, const var, const dis, const par) ;
```
e in: \((m \times T \times 1)\) matrix, the residuals series.
var in: \((m \times T \times 1)\) matrix, the variance series.
dis in: integer, the distribution (0:normal, 1:Student, 2:GED or 3:skewed Student).
par in: \((2 \times 1)\) matrix, \(< \log(\xi) ; \nu >\), i.e. the asymmetry coefficient and the degree of freedom.
par is \(<>\) if dist = 0 and \(\log(\xi) = 0\) if dist = 1 or 2.

**Return value**
A \((m \times T \times 1)\) vector with the CD.

**Description**
Computes the CDF of \(\epsilon/\sigma\) for the selected distribution.

**Garch::CDFGED, Garch::CDFTA**

```c
CDFGED (const ee, const nu) ;
CDFTA (const ee, const logxi, const nu) ;
```
ee in: \((m \times T \times 1)\) matrix, the series.
logxi in: double, logarithm of the skewness parameter.
u in: double, degree of freedom (kurtosis parameter).

**Return value**
A \((\text{rows}(ee) \times 1)\) matrix with the cdf of the selected distribution.

**Description**
Computes the cdf of the GED (CDFGED) and the skewed Student (CDFTA).

**Garch::COVAR**

```c
COVAR(const p) ;
```
\( p \) in: 1 or 0, to print the variance-covariance matrix of the estimated parameters.

**No return value**

**Description**

If \( p \) is 1, the variance-covariance matrix of the estimated parameters is printed in the output.

**Garch::Dialogs**

Dialogs() ;

**No return value**

**Description**

Only used with OxPack. Launches dialogs related to the starting values. There are two possibilities: the "Parameter-by-parameter" dialog or the "Vector" dialog. The former is launched if the user has previously selected the "Manually (Individual Form)" option in OxPack while the latter is available when having chosen "Manually (Vector Form)".

**Garch::DISTRI**

DISTRI(const dist) ;

\( \text{dist} \) in: integer (0, 1, 2 or 3), distribution selection

**No return value**

**Description**

Selects the distribution. If \( \text{dist} \) is 0, the Gaussian (Normal) distribution is selected, if it is 1, it is the Student-\( t \) distribution, if it is 2, it is the Generalized Error distribution (GED) and if it is 3, it is the skewed Student-\( t \).

**Garch::DoEstimation, Garch::Estimate**

DoEstimation(const vStart) ;

Estimate () ;

\( \text{vStart} \) in: \((m_{cPar} \times 1)\) matrix, starting values of the parameters to be estimated.

**Return value**

1 if model successfully estimated, 0 if it failed.

**Description**

*DoEstimation* is launched from the “Light version” of the program while *Estimate* is launched from the “Full version”. These procedures are the core procedures of the program. They successively launch others procedures to initialize parameters, to estimate the formulated model, to print the results, to run the tests or to display the graphics.

**Garch::FEM**

FEM(const forc, const obs) ;

\( \text{forc} \) in: \((m_{cTforc} \times 2)\) matrix, mean forecast \( \sim \) variance forecast

\( \text{trunc} \) in: \((m_{cTforc} \times 2)\) matrix, observed series \((m_{Yfor}) \sim \) observed variance \((m_{Hfor})\)
**Return value**

1 if the tests are successfully run, 0 otherwise.

**Description**

Computes and prints 10 forecast error measures: Mean Squared Error (MSE), Median Squared Error (MedSE), Mean Error (ME), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Heteroskedastic Mean Squared Error (HMSE), Mean Absolute Percentage Error (MAPE), Adjusted Mean Absolute Percentage Error (AMAPE), Percentage Correct Sign (PCS), Theil Inequality Coefficient (THEIL) and Logarithmic Loss Function (LL). See Brooks, Burke, and Persand (1997) for more details about these measures.

**Garch::FigLL**

FigLL(const vP, const adFunc, const avScore, const amHessian);

 vP in: \((m \times cP \times 1)\) matrix, parameters to be estimated
 out: estimated parameters

 adFunc in: address
 out: double, log-likelihood function value at vP

 avScore in: 0, or an address
 out: if not 0 on input, \((m \times cP \times 1)\) matrix with first derivatives at vP

 amHessian in: 0, as MaxBFGS does not require the Hessian.

**Return value**

1 if successful, 0 if evaluation failed.

**Description**

This procedure is optimized by Ox with the MaxBFGS function. This function uses the Broyden, Fletcher, Goldfarb and Shanno (BFGS) quasi-Newton method (see Doornik, p.240, for further details).

**Garch::Filter**

Filter() ;

**Return value**

1 if successful, 0 if failure.

**Description**

Allocates the correct filter to the specified model.

**Garch::FixBounds**

FixBounds(const m);

 m in: \((18 \times 2)\) matrix, bounds values in startingvalues.txt.

**Return value**

1 if successful, 0 if allocation failed.
Description

This allocates the upper and lower bounds for the parameters to the class member \( m.mBound \) variable. Note the these bounds are editable by changing default values in `startingvalues.txt`. See Section 4.1 for more details.

Garch::FixParam

FixParam(const cfix, const fix);

cfix in: integer, 1 to enable this option, 0 to disable
fix in: \((k \times 1)\) vector of 1’s (fixed) or 0’s (estimated)

No return value

Description

When cfix=1, this procedure fixes several of the \( k \) parameters to their starting values. For instance, in an ARCH(1) model their are by default 3 parameters to be estimated. When using `DoEstimation(<0.01;0.01;0.5>)`, to launch the estimation, the ARCH parameter is initialized at 0.5. However, when setting `FixParam(1, <0;0;1>)`, the ARCH parameter is not estimated and fixed at 0.5. This option thus allows to estimate an AR(2)-ARCH(1) model with the estimation of only the lag 2 of the AR. 5 parameters are considered here (two constants, two AR parameters and the ARCH parameter). To do so, one has to select `FixParam(1, <0;1;0;0;0>)` for the AR(1) parameter to be fixed and `DoEstimation(<0.01;0.1;0.01;0.5>)` to fix it to 0.

Garch::FOR_APARCH, Garch::FOR_GARCH, Garch::FOR_GJR

FOR_APARCH(const e, const h, const p, const q, const alpha, const beta, const gamma, const delta, const level_forc, const Ki);
FOR_GARCH(const e, const h, const p, const q, const alpha, const beta, const gamma, const delta, const level_forc);
FOR_GJR(const e, const h, const p, const q, const alpha, const beta, const leverage, const level_forc, const prob_neg);
FOR_FIAPARCH_BBM(const e, const hh, const p, const q, const d, const alpha, const beta, const gamma, const delta, const level_forc, const laglamb, const Ki);
FOR_FIAPARCH_Chung(const e, const hh, const p, const q, const d, const alpha, const beta, const gamma, const delta, const level_forc, const Ki);
FOR_FIEGARCH(const e, const h, const p, const q, const d, const alpha, const beta, const gamma, const delta, const level_forc, const laglamb);
FOR_FIAPARCH_BBM(const e, const h, const p, const q, const d, const alpha, const beta, const gamma, const delta, const level_forc, const laglamb);
FOR_FIAPARCH_Chung(const e, const h, const p, const q, const d, const alpha, const beta, const gamma, const delta, const level_forc, const laglamb);
e in: \((m \times 1)\) matrix, pre-forecasts residual values
h in: \((m \times 1)\) matrix, pre-forecasts conditional variance
p in: integer, GARCH order.
q in: integer, ARCH order.
alpha in: \((q \times 1)\) matrix, ARCH coefficients.
beta in: \((p \times 1)\) matrix, GARCH coefficients.
leverage in: \((q \times 1)\) matrix, asymmetry coefficients of the GJR (\(\omega_i\) in Equation 16).
gamma in: \((q \times 1)\) matrix, asymmetry coefficients of the APARCH (\(\gamma_i\) in Equation 17).
delta in: double, standard deviation exponent (\(\delta\) in Equation 17).
theta1 in: \((q \times 1)\) matrix, sign effect of the EGARCH (\(\theta_1\) in Equation 12).
theta2 in: \((q \times 1)\) matrix, magnitude effect of the EGARCH (\(\theta_2\) in Equation 12).
Ki in: \((q \times 1)\) matrix, output of KiAparch(dist, q, par, delta, gamma)
prob_neg in: double, probability that \(\epsilon < 0\) (it equals 0.5 for the symmetric distributions and \(1/(1 + \xi^2)\) for the skewed Student).
dist in: integer, selected distribution (0: Normal, 1: Student, 2: GED, 3: Skewed Student)
level_forc in: \((m \times 1)\) or \((m \times m\) matrix, it is the level, computed as \(b'X\) if there are explanatory variables in the variance.
laglamb in: integer, truncation order (BBM method).

Return value
A \((m \times 1)\) matrix with the forecasts of the variance.

Description
These are the forecasting procedures. level_forc is a \((m \times 1)\) vector of ones if there is no independent variable and a \((m \times m)\) matrix equals to \(b'X\) if there are explanatory variables in the variance.

Garch::FOR_ARMA, Garch::FOR_ARFIMA
FOR_ARMA (const y, const p, const q, const arma, const level_forc, const e) ;
FOR_ARFIMA (const y, const p, const q, const d, const arma, const level_forc, const e) ;
y y in: \((m \times 1)\) matrix, with \(y = y - b'X\) (X is a matrix of explanatory variables and b the estimated parameters).
p in: integer, AR order.
q in: integer, MA order.
arma in: \((1 \times (p+q))\) matrix, AR coefficients followed by MA coefficients.
level_forc in: \((m \times 1)\) matrix, independent variables.
e in: \((m \times 1)\) matrix, residuals series.
d in: double, long memory coefficient.
Return value

A \((m_cTForc \times 1)\) matrix with the forecasts of the mean.

Description

These are the filters for the mean equation. They compute forecasts for ARMA and ARFIMA specifications (without explanatory variables).

Garch::FOR_GRAPHS

\begin{verbatim}
FOR_GRAPHS(const plot, const pre, const type, const valcrit);
\end{verbatim}

\begin{itemize}
  \item \textbf{plot} in: integer, area wherein the first graph is plotted
  \item \textbf{pre} in: integer, number of pre-observations
  \item \textbf{type} in: integer, type of confidence intervals (0: none, 1: bands, 2: bars, 3: fans).
  \item \textbf{valcrit} in: double, critical value for the confidence interval (forecasts \(\pm valcrit \times \text{std.err.}\)).
\end{itemize}

No return value

Description

Displays graphics of the mean forecasts (with or without confidence intervals) and/or the variance forecasts and/or the forecasted series and/or the observed series in the GiveWin frontend.

Garch::FORECAST

\begin{verbatim}
FORECAST(const i, const nbForc, const iprint);
\end{verbatim}

\begin{itemize}
  \item \textbf{i} in: 1 to compute forecasts, 0 otherwise.
  \item \textbf{nbForc} in: integer, number of forecasts.
  \item \textbf{iprint} in: 1 to print the forecasts, 0 otherwise.
\end{itemize}

No return value

Description

If \(i\) is 1, one-step ahead forecasting will be executed. The number of forecasts is given by the value of \(nbForc\). These forecasts will be printed in the output if \(iprint\) is 1.

Garch::FORCASTING

\begin{verbatim}
FORCASTING();
\end{verbatim}

Return value

\(m_mForc\), a \((m_cTForc \times 2)\) matrix containing the forecasts for the mean and for the variance.

Description

This procedure launches the forecasts for the mean, then for the variance and allocates different filters depending on the specification of the model.

Garch::FOREGRAPHS

\begin{verbatim}
FOREGRAPHS(const d, const s, const file);
\end{verbatim}
d in: 1 to draw forecasts graphics, 0 otherwise.

s in: 1 to save forecasts graphics, 0 otherwise.

file in: string, name of the EPS file containing the saved graphics.

No return value

Description

This function calls FOR_GRAPHS to draw forecasts graphics in GiveWin when using the “Light” version. It also allows to save these graphs in a EPS (Encapsulated PostScript) file.

Garch::Garch
Garch();

No return value

Description

Constructor.

Garch::GaussLik, Garch::GEDLik, Garch::StudentLik, Garch::SkStudentLik
GaussLik(const vE, const vSigma2);
GEDLik(const vE, const vSigma2, const a);
StudentLik(const vE, const vSigma2, const v);
SkStudentLik(const vE, const vSigma2, const s, const v);

vE in: (m_cT x 1) matrix, residuals.
vSigma2 in: (m_cT x 1) matrix, conditional variance.
a in: double, asymmetric coefficient.
s in: double, skewness parameter.
v in: double, degree of freedom (kurtosis parameter).

Return value

The log-likelihood function value associated with vE and vSigma2.

Description

Computes the log-likelihood functions of the four available distributions.

Garch::GetForcData, Garch::GetXBetaForc, Garch::GetZBetaForc;
GetForcData(const iGroup, const cTforc);
GetXBetaForc(const cTforc);
GetZBetaForc(const cTforc);

iGroup in: name of the variable group.
cTforc in: number of forecasts.

Return value

(m_cTforc x 1) matrix containing the realized values of the regressors (in the mean or in the variance).
Description

GetForcData collects the realized values matrix of all the regressor(s) for the forecasting period \([t + 1 : t + m_{cForc}]\). GetXBetaForc and GetZBetaForc do the same for the regressor(s) in the mean and the regressor(s) in the variance, respectively.

Garch::GetINames, Garch::GetParNames, Garch::GetXNames, Garch::GetYNames, Garch::GetZNames;
GetINames();
GetParNames();
GetXNames();
GetYNames();
GetZNames();

Return value

Array of strings with the names of the variables or parameters.

Description

These procedures collect the names of the series (Y), the realized volatility (I), the regressors in the mean equation (X) or in the variance equation (Z) or the estimated parameters (Par).

Garch:: GetPara;
GetPara();

Return value

1 if successful, 0 if failure.

Description

It constructs the parameters vector and allocates it to \(m_vPar\). The order of the parameters is the following: constant in mean (1 item), regressors in mean coefficients (\(m_cXM\) items), \(d\) coefficient (1 item), AR coefficients (\(p\) items), MA coefficients (\(q\) items), constant in var. (1 item), regressors in variance coefficients (\(m_cXV\) items), \(F.I.(d)\) coefficient (1 item), GARCH coefficients (\(p\) items), ARCH coefficients (\(q\) items), GJR coefficients (\(q\) items), EGARCH coefficients (\(q \times 2\) items), APARCH coefficients (\(q + 1\) items), GED degrees of freedom (1 item) and Student degrees of freedom (1 item).

Garch::GetRes
GetRes(const y, const x);

\(y\) in: \((m_cT \times 1)\) matrix, dependent variable

\(x\) in: \((m_cT \times m_cX)\) matrix, regressors

Return value

\((m_cT \times 1)\) matrix containing the residuals.

Description

Computes the residuals of the mean equation without taking into account the AR(FI)MA process.
Garch::GRAPHS
GRAPHS(const d, const s, const file) ;

  d in: 1 to draw graphics of the estimation, 0 otherwise.
  s in: 1 to save graphics of the estimation, 0 otherwise.
  file in: string, name of the EPS file with the saved graphics.

No return value

Description
This function calls Test_Graphic_Analysis to draw various graphics resulting from the estimation. These graphs are displayed in GiveWin. GRAPHS also allows to save these graphs in a EPS (Encapsulated PostScript) file.

Garch::ICriterion
ICriterion(const LogL, const n, const q) ;

  LogL in: double, the value of the log-likelihood function.
  n in: integer, number of observations.
  q in: integer, number of parameters.

Return value
1 if the test is successfully run, 0 otherwise.

Description
Computes and prints four information criteria : the Akaike, Schwarz, Shibata and Hannan-Quinn tests. See Section 4.3 for more details about this test.

Garch::InitData
InitData() ;

Return value
1 if successful, 0 if failure.

Description
Initializes the model by allocating the Y series and the regressors to class members, computing the number of observations of the sample and the number parameters to be estimated.

Garch::InitStartValues
InitStartValues(const init_par, const init_bounds) ;

  init_par in: 1 if default starting values used, 0 otherwise.
  init_bound in: 1 if bounded parameters used, 0 otherwise.

No return value

Description
Initializes the starting values when the user do not enter any specific starting values. These values are:
• Constant in the mean: 0.05
• Regressors in the mean: 0.01
• ARFIMA(p,d,q): \( p_1 = 0.2, p_{>1} = 0.05, d = 0.1, q_1 = 0.15, q_{>1} = 0.02 \)
• Constant term in the variance equation: 0.01
• GARCH: \( \beta_1 = 0.7 \) (if GARCH) or 0.45 (if FIGARCH), \( \beta_{>1} = 0.1 \).
• ARCH: \( \alpha_i = 0.1 \)
• FIGARCH: \( d = 0.5 \)
• GJR: \( \omega_i = 0.01 \)
• EGARCH: \( \phi_1 = -0.1 \) and \( \phi_2 = 0.2 \)
• APARCH: \( \delta = 1.2, \gamma_1 = 0.15, \gamma_{>1} = 0.05 \)
• skewed Student distribution: \( \xi = 0.01 \) (asymmetry coefficient).
• Student distribution: \( \nu = 6 \) (degrees of freedom).
• GED distribution: \( \nu = 2 \).

All these values can easily be modified by the user just by editing the `startingvalues.txt` file. This file is automatically installed in the `..//ox/packages/garch` directory when unzipping `garch22.zip`. In this `startingvalues.txt` file, one may also change lower and upper bounds of all the parameters.

**Garch::INVCDFGED, Garch::INVCDFTA**

INVCDFGED (const p, const nu);
INVCDFTA (const p, const logxi, const nu);

\[ p \quad \text{in: double, probability.} \]
\[ \logxi \quad \text{in: double, logarithm of the skewness parameter.} \]
\[ \nu \quad \text{in: double, degree of freedom (kurtosis parameter).} \]

*Return value*

The solution of the integral equation \( p = F(x|\logxi, \nu) \).

*Description*

Computes the the inverse cdf of the GED (INVCDFGED) and the skewed Student (INVCDFTA).

**Garch::ITER**

Integrate(const i);

\[ i \quad \text{in: integer, number of iterations between intermediary prints.} \]
No return value

Description
With G@RCH, it is possible to print intermediary results of the estimation. This function allows the user to easily select the number of iteration between printing results. For instance, if \( i \) is 10, intermediary values of the parameters and the log-likelihood function will be printed every 10 iterations. When \( i \) is 0, no intermediary result is printed.

**Garch::MLE**
MLE(const method);

method in: integer, method selection.

No return value

Description
Selection of the estimation method. If method = 0, both (Approximate) Maximum Likelihood and Quasi-Maximum Likelihood Estimates will be computed. If method = 1, only MLE is selected and if it is equal to 2, only QMLE are computed.

**Garch::MLEMeth**
MLEMeth(const par, const parnames, const title, const nbpar);

par in: (\( m \cdot cPar \) x 1) matrix, estimated parameters
parnames in: array of \( m \cdot cPar \) strings, estimated parameters names
title in: string, selected method name
nbpar in: integer, number of parameters (\( m \cdot cPar \))

No return value

Description
Prints the estimated parameters, their standard deviations, t-tests and p-values with their names. Depending on the user’s choice, ML estimates, QML estimates or both will be printed.

**Garch::MODEL**
MODEL(const mod);

mod in: integer, used to select the Garch model.

No return value

Description
The argument takes a value between 1 and 11 to select the ARCH model to be used in the conditional variance:

1: GARCH
2: EGARCH
3: GJR
4: FIGARCH (BBM)
5: FIGARCH (Chung)
6: FIEGARCH
7: FIEGARCH
4: APARCH 9: FIAPARCH (BBM)
5: IGARCH 10: FIAPARCH (Chung)
11: HYGARCH

Garch::MZ
MZ(const HFor, const MatFor, const nbFor);

HFor in: blabla
MatFor in: blabla
nbFor in: integer, number of forecasts

Return value
1 if the tests are successfully run, 0 otherwise.

Description
Computes and prints the Mincer-Zarnowitz regression on the forecasted volatility. See Section 4.3 for more details about this regression.

Garch::Normality
Normality(const e);

e in: \((m \times T \times 1)\) matrix, series to be tested.

Return value
1 if the test is successfully run, 0 otherwise.

Description
Computes and prints skewness, excess kurtosis and Jarque-Bera normality test with the associated adjusted t-statistics and p-values. See Section 4.3 for more details about this test.

Garch::NYBLOM
NYBLOM(const i);

i in: 1 or 0, to compute the Nyblom test.

No return value

Description
If \(i\) is 1, the parameters stability test of Nyblom (1989) is computed.

Garch::Nyblom
Nyblom(const eh, const grad);

eh in: \((m \times T \times 1)\) matrix, parameters to be tested.
grad in: \((m \times Parx1)\) matrix, gradients.
Return value
1 if the test is successfully run, 0 otherwise.

Description
Computes and prints the Nyblom test. This checks the constancy of parameters over time. See Nyblom (1989) and Lee and Hansen (1994) for more details.

Garch::Output
Output();
No return value
Description
Prints the specification of the formulated model and launches the standard errors computations.

Garch::PAR
PAR();
Return value
A \( (m_cPar \times 3) \) matrix structured as \( (m_cPar \sim m_cVar \sim m_cVStdErrors \sim m_cVRobStdErrors) \)
Description
Returns a \( (m_cPar \times 3) \) matrix with the parameters estimates, their standard errors and their robust standard errors. It is used together with SAVEPAR to store estimation results of a model in an external file (Microsoft Excel spreadsheet).

Garch::PEARSON
PEARSON(const lags);

lags in: \( (l \times 1) \) matrix, vector containing the \( l \) wanted lags for the test.
It must have the following form: \(< lag_1; lag_2; \ldots; lag_l >\)
No return value
Description
Fixes the lags wanted when computing the adjusted Pearson goodness-of-fit test (see Section 4.3 for more details). By default, \( lags \) is \( <40; 50; 60> \).

Garch::SAVEPAR
SAVEPAR(const i, const file);
i in: 0: store nothing, 1: stores parameters estimates, 2: stores estimates and std.errors, 3: stores estimates, std.errors and robust std. errors.
file in: string, name of the Excel file wherein the values will be stored.
Return value
1 if successful, 0 otherwise.
Description
Allows to store optimized parameters estimates, their standard errors and their robust standard errors in a .xls file (Excel spreadsheet) for further analysis.

**Garch::SBT**

\[
\text{SBT}(\text{const res, const cvar}) ;
\]

- res in: \((m_c T \times 1)\) matrix, residuals.
- cvar in: \((m_c T \times 1)\) matrix, conditional variance.

**Return value**

1 if the test is successfully run, 0 otherwise.

**Description**

Computes and prints Sign Bias Test, negative Size Bias Test, positive Size Bias Test and joint Test for the three effects described in Engle and Ng (1993). See Section 4.3 for more details about this test.

**Garch::SplitPara**

\[
\text{SplitPara}(\text{vP}) ;
\]

- vP in: \((m_c Par \times 1)\) matrix, parameters vector.

**Return value**

1 if successful, 0 if failure.

**Description**

Splits the parameters vector and allocates each one to the corresponding class member.

**Garch::STORE**

\[
\text{STORE}(\text{const res, const res2, const condv, const mfor, const vfor, const name, const file});
\]

- res in: 1 or 0, to store the residuals.
- res2 in: 1 or 0, to store the squared residuals.
- condv in: 1 or 0, to store the conditional variance.
- mfor in: 1 or 0, to store the mean forecasts.
- vfor in: 1 or 0, to store the variance forecasts.
- name in: string, suffix added to “Res\_”, “SqRes\_”, “CondV\_”, “MeanFor\_” or “VarFor\_” to name the saved series.
- file in: if 0, saves as a new .xls file. If 1, saves as a new .in7 file.

**No return value**

**Description**

Stores the residuals, the squared residuals and the conditional variance of the estimated models, but also the forecasted mean and variance. Argument 6 provides a default suffix (“01”) that can be modified. If argument 7 equals 0 (default value), the series will be stored in a new .xls file.
(Microsoft Excel spreadsheet). If it is equal to 1, the series will be stored in a new .in7 file (GiveWin database).

**Garch::Tests**

Tests();

*No return value*

*Description*

Runs the selected tests.

**Garch::TESTS**

TESTSONLY(const p, const a);  

- **p** in: 0 or 1.
- **a** in: 0 or 1.

*No return value*

*Description*

Allows to run the test either for the raw series, prior to any estimation ($p = 1$) or for the estimated series, after the optimization ($a = 1$).

**Garch::TestGraphicAnalysis**

TestGraphicAnalysis(const ser, const res, const sqres, const h, const plot);  

- **ser** in: 1 or 0; 1 if raw series graph wanted.
- **res** in: 1 or 0; 1 if residuals graph wanted.
- **sqres** in: 1 or 0; 1 if squared residuals graph wanted.
- **h** in: 1 or 0; 1 if cond.variance graph wanted.
- **plot** in: integer, area wherein the first graph is plotted

*No return value*

*Description*

Displays graphics of the series and/or the residuals and/or the squared residuals and/or the conditional variance in the GiveWin front-end.

**Garch::Transform**

Transform(const cpar, const bornes, const meth);

- **cpar** in: $(m_{cPar} \times 1)$ matrix, estimated parameters
- **bornes** in: $(m_{cPar} \times 2)$ matrix, upper and lower bounds
- **meth** in: 1, 2 or 3.
**Return value**

\((m.cPar \times 1)\) matrix containing the transformed parameters.

**Description**

Transforms the values of the estimated parameters to take account of the lower and upper bounds of these parameters (see Section 4.1 for details). When \(meth\) is 0, the constrained parameters are transformed into unconstrained parameters. The opposite is obtained by equalling \(meth\) to 1. If \(meth\) is 2, unconstrained standard errors of the estimated parameters are transformed into constrained ones.

**Garch::TRUNC**

TRUNC(const t);

\(trunc\) in: integer, truncation order (this argument is only used with BBM’s approach)

**No return value**

**Description**

It is related to the fractionally integrated (FI) model selection. If the estimation method follows BBM’s (1996) specification, the value of \(t\) will be used as the truncation order. If the estimation method follows Chung’s (1999) specification, \(t\) will equal all the previous observations. One notable difference (among others) between these two methods is indeed that BBM uses a fix number of lags to compute the binomial expansion (Taylor’s theorem) while Chung proposes an increasing number of lags (it includes thus all previous observations). The FIAPARCH model is estimated with the FIGARCH procedure. Here is an example for a FIAPARCH(1,d,1) with the method of BBM:

\[
\text{Figarch\_BBM} (\sqrt{G}, \text{level}, p, q, \text{laglamb}, \alpha|\beta|d|\gamma) \cdot (2/\delta) \\
\text{with } G = |\text{residuals}| - \gamma \cdot (\text{residuals}^{\delta})
\]

**OxPack Functions**

OxPack related functions are described in Doornik (2001).
3 Features of the package

Our attention will be first devoted to review the specifications of the conditional mean equation. Then, some recent contributions in the ARCH modelling framework will be presented.

3.1 Mean equation

Let us consider an univariate time series $y_t$. If $\Omega_{t-1}$ is the information set at time $t-1$, we can define its functional form as:

$$y_t = E(y_t|\Omega_{t-1}) + \varepsilon_t,$$

where $E(.|.)$ denotes the conditional expectation operator and $\varepsilon_t$ is the disturbance term (or unpredictable part), with $E(\varepsilon_t) = 0$ and $E(\varepsilon_t \varepsilon_s) = 0, \forall \ t \neq s$.

This is the mean equation which has been studied and modelled in many ways. Two of the most famous specifications are the Autoregressive (AR) and Moving Average (MA) models. Mixing these two processes and introducing $n_1$ explanatory variables in the equation, we obtain this ARMAX($n, s$) process,

$$\Psi (L) (y_t - \mu_t) = \Theta (L) \varepsilon_t,$$

where $L$ is the lag operator$^3$, $\Psi (L) = 1 - \sum_{i=1}^{n_1} \psi_i L^i$ and $\Theta (L) = 1 + \sum_{j=1}^{s} \theta_j L^j$. To start the recursion, it is convenient to set the initial conditions as $\varepsilon_t = 0$ for all $t \leq \max\{p, q\}$.

Several studies have shown that the dependent variable (interest rate returns, exchange rate returns, etc.) may exhibit significant autocorrelation between observations widely separated in time. In such a case, we can say that $y_t$ displays long memory, or long-term dependence and is best modelled by a fractionally integrated ARMA process (so called ARFIMA process) initially developed in Granger (1980) and Granger and Joyeux (1980) among others.$^4$ The ARFIMA($n, \zeta, s$) is given by:

$$\Psi (L) (1 - L)\zeta (y_t - \mu_t) = \Theta (L) \varepsilon_t,$$

where the operator $(1 - L)\zeta$ accounts for the long memory of the process and is defined as:

$$(1 - L)\zeta = \sum_{k=0}^{\infty} \frac{\Gamma(\zeta + 1)}{\Gamma(k + 1) \Gamma(\zeta - k + 1)} L^k$$

$$= 1 - \zeta L - \frac{1}{2} \zeta(1 - \zeta)L^2 - \frac{1}{6} \zeta(1 - \zeta)(2 - \zeta)L^3 - \ldots$$

$\quad = 1 - \sum_{k=1}^{\infty} c_k(\zeta)L^k,$

$^3$Recall that $L^k y_t = y_{t-k}$.

with $0 < \zeta < 1$, $c_1(\zeta) = \zeta, c_2(\zeta) = \frac{1}{2} \zeta(1 - \zeta)$, \ldots and $\Gamma(.)$ denoting the Gamma function (see Baillie, 1996, for a survey on this topic). The truncation order of the infinite summation is set to $t - 1$.

It is worth noting that Doornik and Ooms (1999) recently provided an Ox package for estimating, forecasting and simulating ARFIMA models. However, in opposition to our package, they assume that the conditional variance is constant over time.

3.2 Variance equation

The $\varepsilon_t$ term in Eq. (1)-(3) is the innovation of the process. Two decades ago, Engle (1982) defined as an Autoregressive Conditional Heteroscedastic (ARCH) process, all $\varepsilon_t$ of the form:

$$
\varepsilon_t = z_t \sigma_t,
$$

where $z_t$ is an independently and identically distributed (i.i.d.) process with $E(z_t) = 0$ and $Var(z_t) = 1$. By definition, $\varepsilon_t$ is serially uncorrelated with a mean equal to zero, but its conditional variance equals $\sigma_t^2$ and, therefore, may change over time, contrary to what is assumed in the standard regression model.

The models provided by our program are all ARCH-type. They differ on the functional form of $\sigma_t^2$ but the basic principles are the same. Besides the traditional ARCH and GARCH models, we focus mainly on two kinds of models: the asymmetric models and the fractionally integrated models. The former are defined to take account of the so-called “leverage effect” observed in many stock returns, while the latter allows for long-memory in the variance. Early evidence of the “leverage effect” can be found in Black (1976), while persistence in volatility is a common finding of many empirical studies; see for instance Bera and Higgins (1993), Bollerslev, Chou, and Kroner (1992) or Palm (1996) for excellent surveys on ARCH models.

3.2.1 ARCH Model

The ARCH ($q$) model can be expressed as:

$$
\varepsilon_t = z_t \sigma_t,
$$

$$
z_t \sim i.i.d. \ D(0,1)
$$

$$
\sigma_t^2 = \omega + \sum_{i=1}^{q} \alpha_i \varepsilon_{t-i}^2,
$$

where $D(.)$ is a probability density function with mean 0 and unit variance (it will be defined in Section 4.2).

The ARCH model can describe volatility clustering. The conditional variance of $\varepsilon_t$ is indeed an increasing function of the square of the shock that occurred in $t - 1$. Consequently, if $\varepsilon_{t-1}$ was large in absolute value, $\sigma_t^2$ and thus $\varepsilon_t$ is expected to be large (in absolute value) as well. Notice

---

5For stochastic volatility models, see Koopman, Shepard, and Doornik (1998).
that even if the conditional variance of an ARCH model is time-varying \( \sigma_t^2 = E(\varepsilon_t^2|\psi_{t-1}) \), the unconditional variance of \( \varepsilon_t \) is constant and, provided that \( \omega > 0 \) and \( \sum_{i=1}^{q} \alpha_i < 1 \), we have:

\[
\sigma^2 = E(\varepsilon_t^2) = \frac{\omega}{1 - \sum_{i=1}^{q} \alpha_i}.
\]

(7)

Note also that the ARCH model can explain part of the excess kurtosis that we observe in financial time series. As shown by Engle (1982) for the ARCH(1) case under the normality assumption, the kurtosis of \( \varepsilon_t \) is equal to \( \frac{3(1-\alpha_1^2)}{1-3\alpha_1} \). The kurtosis is thus finite if \( \alpha_1 < \frac{1}{3} \) and larger than 3 (the kurtosis of a standard normal distribution) if \( \alpha_1 > 0 \).

The computation of \( \sigma_t^2 \) in Eq. (6) depends on past (squared) residuals \( \varepsilon_t^2 \), that are not observed for \( t = 0, -1, \ldots, -q + 1 \). To initialize the process, the unobserved squared residuals have been set to their sample mean.

In the rest of the paper, \( \omega \) is assumed fixed. If \( n_2 \) explanatory variables are introduced in the model, \( \omega_t = \omega + \sum_{i=1}^{n_2} \omega_i x_{i,t} \) with an exception for the exponential models (EGARCH and FIEGARCH) where \( \omega_t = \omega + \ln \left( 1 + \sum_{i=1}^{n_2} \omega_i x_{i,t} \right) \).

Finally, \( \sigma_t^2 \) has obviously to be positive for all \( t \). Sufficient conditions to ensure that the conditional variance in Eq. (6) is positive are given by \( \omega > 0 \) and \( \alpha_i \geq 0 \). However, these conditions are not necessary as shown by Nelson and Cao (1992). Furthermore, when explanatory variables enter the ARCH equation, these positivity constraints are not valid anymore (even if the conditional variance still has to be non-negative).

### 3.2.2 GARCH Model

Early empirical evidence has shown that a high ARCH order has to be selected to catch the dynamics of the conditional variance (thus involving the estimation of numerous parameters). The Generalized ARCH (GARCH) model of Bollerslev (1986) is an answer to this issue. It is based on an infinite ARCH specification and it allows to reduce the number of estimated parameters by imposing non-linear restrictions on them. The GARCH \((p,q)\) model can be expressed as:

\[
\sigma_t^2 = \omega + \sum_{i=1}^{q} \alpha_i \varepsilon_{t-i}^2 + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2.
\]

(8)

Using the lag or backshift operator \( L \), the GARCH \((p,q)\) model is:

\[
\sigma_t^2 = \omega + \alpha(L)\varepsilon_t^2 + \beta(L)\sigma_t^2,
\]

(9)

with \( \alpha(L) = \alpha_1 L + \alpha_2 L^2 + \ldots + \alpha_q L^q \) and \( \beta(L) = \beta_1 L + \beta_2 L^2 + \ldots + \beta_p L^p \).

If all the roots of the polynomial \( |1 - \beta(L)| = 0 \) lie outside the unit circle, we have:

\[
\sigma_t^2 = \omega [1 - \beta(L)]^{-1} + \alpha(L) [1 - \beta(L)]^{-1} \varepsilon_t^2,
\]

(10)

29
which may be seen as an ARCH($\infty$) process since the conditional variance linearly depends on all previous squared residuals. In this case, the conditional variance of $y_t$ can become larger than the unconditional variance given by:

$$\sigma^2 \equiv E(\varepsilon_t^2) = \frac{\omega}{1 - \sum_{i=1}^{q} \alpha_i - \sum_{j=1}^{p} \beta_j},$$

if past realizations of $\varepsilon_t^2$ are larger than $\sigma^2$ (Palm, 1996).

As in the ARCH case, some restrictions are needed to ensure $\sigma_t^2$ to be positive for all $t$. Bollerslev (1986) shows that imposing $\omega > 0$, $\alpha_i \geq 0$ (for $i = 1, \ldots, q$) and $\beta_j \geq 0$ (for $j = 1, \ldots, p$) is sufficient for the conditional variance to be positive. In practice, the GARCH parameters are often estimated without the positivity restrictions. Nelson and Cao (1992) argued that imposing all coefficients to be nonnegative is too restrictive and that some of these coefficients are found to be negative in practice while the conditional variance remains positive (by checking on a case-by-case basis). Consequently, they relaxed this constraint and gave sufficient conditions for the GARCH(1, q) and GARCH(2, q) cases based on the infinite representation given in Eq. (10). Indeed, the conditional variance is strictly positive provided $\omega [1 - \beta(1)]^{-1} > 0$ is positive and all the coefficients of the infinite polynomial $\alpha(L) [1 - \beta(L)]^{-1}$ in Eq. (10) are nonnegative. The positivity constraints proposed by Bollerslev (1986) can be imposed during the estimation (see 4.1). If not, these constraints, as well as the ones implied by the ARCH($\infty$) representation, will be tested a posteriori and reported in the output.

### 3.2.3 EGARCH Model

The Exponential GARCH (EGARCH) model is introduced by Nelson (1991). Bollerslev and Mikkelsen (1996) propose to re-express the EGARCH model as follows:

$$\ln \sigma_t^2 = \omega + [1 - \beta(L)]^{-1} [1 + \alpha(L)] g(z_{t-1}).$$

(11)

The value of $g(z_t)$ depends on several elements. Nelson (1991) notes that, “to accommodate the asymmetric relation between stock returns and volatility changes (...) the value of $g(z_t)$ must be a function of both the magnitude and the sign of $z_t$.” That is why he suggests to express the function $g(.)$ as

$$g(z_t) \equiv \gamma_1 z_t + \gamma_2 [||z_t| - E|z_t||].$$

(12)

$E|z_t|$ depends on the assumption made on the unconditional density of $z_t$. Indeed, for the normal distribution,

$$E(|z_t|) = \sqrt{\frac{2}{\pi}}.$$  

(13)

Note that with the EGARCH parameterization of Bollerslev and Mikkelsen (1996), it is possible to estimate an EGARCH $(p, 0)$ since $\ln \sigma_t^2$ depends on $g(z_{t-1})$, even when $q = 0.$
For the skewed Student distribution,
\[
E(|z_t|) = \frac{4\xi^2}{\xi + \frac{x}{2}} \frac{\Gamma \left(\frac{1+v}{2}\right) \sqrt{v-2}}{\Gamma \left(\frac{v}{2}\right)}
\]
where \(\xi = 1\) for the symmetric Student.

For the GED, we have
\[
E(|z_t|) = \lambda \upsilon \frac{2}{\Gamma \left(\frac{2}{v}\right)} \frac{\Gamma \left(\frac{1}{2}\right)}{\Gamma \left(\frac{v}{2}\right)}.
\]
\(\xi, \upsilon\) and \(\lambda\) concern the shape of the non-normal densities and will be defined in Section 4.2.

Note that the use of a ln transformation of the conditional variance ensures that \(\sigma_t^2\) is always positive.

### 3.2.4 GJR Model

This popular model is proposed by Glosten, Jagannathan, and Runkle (1993). Its generalized version is given by:
\[
\sigma_t^2 = \omega + \sum_{i=1}^{q} (\alpha_i |\varepsilon_{t-i}| + \gamma_i S_{t-i} \varepsilon_{t-i}) + \sum_{j=1}^{p} \beta_j \sigma_t^2,
\]
where \(S_{t-i}\) is a dummy variable.

In this model, it is assumed that the impact of \(\varepsilon_t^2\) on the conditional variance \(\sigma_t^2\) is different when \(\varepsilon_t\) is positive or negative. The TGARCH model of Zakoian (1994) is very similar to the GJR but models the conditional standard deviation instead of the conditional variance. Finally, Ling and McAleer (2002) has proposed, among other stationarity conditions for GARCH models, the conditions of existence of the second and fourth moment of the GJR.

### 3.2.5 APARCH Model

This model has been introduced by Ding, Granger, and Engle (1993). The APARCH \((p, q)\) model can be expressed as:
\[
\sigma_t^{\delta} = \omega + \sum_{i=1}^{q} \alpha_i (|\varepsilon_{t-i}| - \gamma_i \varepsilon_{t-i})^{\delta} + \sum_{j=1}^{p} \beta_j \sigma_t^{\delta},
\]
where \(\delta > 0\) and \(-1 < \gamma_i < 1\) \((i = 1, ..., q)\).

This model couples the flexibility of a varying exponent with the asymmetry coefficient (to take the “leverage effect” into account). The APARCH includes seven other ARCH extensions as special cases:\footnote{Complete developments leading to these conclusions are available in Ding, Granger, and Engle (1993).}

- The ARCH of Engle (1982) when \(\delta = 2\), \(\gamma_i = 0\) \((i = 1, \ldots, p)\) and \(\beta_j = 0\) \((j = 1, \ldots, p)\).
- The GARCH of Bollerslev (1986) when \(\delta = 2\) and \(\gamma_i = 0\) \((i = 1, \ldots, p)\).
- Taylor (1986)/Schwert (1990)’s GARCH when \(\delta = 1\), and \(\gamma_i = 0\) \((i = 1, \ldots, p)\).
• The GJR of Glosten, Jagannathan, and Runkle (1993) when $\delta = 2$.
• The TARCH of Zakoian (1994) when $\delta = 1$.
• The NARCH of Higgins and Bera (1992) when $\gamma_i = 0$ ($i = 1, \ldots, p$) and $\beta_j = 0$ ($j = 1, \ldots, p$).
• The Log-ARCH of Geweke (1986) and Pentula (1986), when $\delta \to 0$.

Following Ding, Granger, and Engle (1993), if $\omega > 0$ and $\sum_{i=1}^{q} \alpha_i E(|z| - \gamma_i z)^{\delta} + \sum_{j=1}^{p} \beta_j < 1$, a stationary solution for Eq. (17) exists and is:

$$E(\sigma_t^2) = \frac{\alpha_0}{1 - \sum_{i=1}^{q} \alpha_i (|z| - \gamma_i z)^{\delta} - \sum_{j=1}^{p} \beta_j}.$$  (18)

Notice that if we set $\gamma = 0$, $\delta = 2$ and $z_t$ has zero mean and unit variance, we have the usual stationarity condition of the GARCH(1,1) model ($\alpha_1 + \beta_1 < 1$). However, if $\gamma \neq 0$ and/or $\delta \neq 2$, this condition depends on the assumption made on the innovation process.

Ding, Granger, and Engle (1993) derived a closed form solution to $\kappa_i = E(|z| - \gamma_i z)^{\delta}$ in the Gaussian case. Lambert and Laurent (2001) show that for the standardized skewed Student:

$$\kappa_i = \left\{ \xi^{-(1+\delta)}(1+\gamma_i)^{\delta} + \xi^{1+\delta}(1-\gamma_i)^{\delta} \right\} \frac{\Gamma\left(\frac{\xi+2}{\xi}\right)\Gamma\left(\frac{\nu/2}{2}(\nu-2)\right)\Gamma\left(\frac{\nu}{2}\right)}{\Gamma\left(\frac{\xi+\nu}{2}\right)\Gamma\left(\frac{\nu}{2}\right)}.$$

For the GED, we can show that:

$$\kappa_i = \frac{[\xi(1+\gamma_i) + (1-\gamma_i)]^2 \xi^{\nu/2} \Gamma\left(\frac{\xi+4}{\xi}\right)\lambda_{\nu,\upsilon}}{\Gamma\left(\frac{\xi+\nu}{2}\right)}.$$

Note that $\xi$, $\nu$, and $\lambda_{\nu,\upsilon}$ concern the shape of the non-normal densities and will be defined in Section 4.2.

### 3.2.6 IGARCH Model

In many high-frequency time-series applications, the conditional variance estimated using a GARCH($p$, $q$) process exhibits a strong persistence, that is:

$$\sum_{j=1}^{p} \beta_j + \sum_{i=1}^{q} \alpha_i \approx 1.$$

If $\sum_{j=1}^{p} \beta_j + \sum_{i=1}^{q} \alpha_i < 1$, the process $(\varepsilon_t)$ is second order stationary, and a shock to the conditional variance $\sigma_t^2$ has a decaying impact on $\sigma_{t+h}^2$, when $h$ increases, and is asymptotically negligible.

However, if $\sum_{j=1}^{p} \beta_j + \sum_{i=1}^{q} \alpha_i \geq 1$, the effect on $\sigma_{t+h}^2$ does not die out asymptotically. This property is called persistence in the literature.

---

*For the symmetric Student density, $\xi = 1$. 

32
When this sum is equal to one, we are confronted to an Integrated GARCH (IGARCH) model, meaning that current information remains of importance when forecasting the volatility for all horizons. A similar concept is present in the mean equation: when the sum of all AR coefficients and MA coefficients is equal to one, the ARMA process in integrated (ARIMA).

Recall that the GARCH($p,q$) model can be expressed as an ARMA process. Using the lag operator $L$, we can rearrange Eq. (8) as:

$$[1 - \alpha(L) - \beta(L)]\varepsilon_t^2 = \omega + [1 - \beta(L)](\varepsilon_t^2 - \sigma_t^2).$$

When the $[1 - \alpha(L) - \beta(L)]$ polynomial contains a unit root, i.e. the sum of all the $\alpha_i$ and the $\beta_j$ is one, we have the IGARCH($p,q$) model of Engle and Bollerslev (1986). It can then be written as:

$$\phi(L)(1 - L)\varepsilon_t^2 = \omega + [1 - \beta(L)][1 - \beta(L)]^{-1}\varepsilon_t^2,$$

(19)

where $\phi(L) = [1 - \alpha(L) - \beta(L)](1 - L)^{-1}$ is of order $[\max\{p,q\}-1]$.

We can rearrange Eq. (19) to express the conditional variance as a function of the squared residuals. After some manipulations, we have:

$$\sigma_t^2 = \frac{\omega}{1 - \beta(L)} + \left\{1 - \phi(L)(1 - L)[1 - \beta(L)]^{-1}\right\}\varepsilon_t^2.$$  

(20)

### 3.2.7 Fractionally Integrated Models

Volatility tends to change quite slowly over time, and, as shown in Ding, Granger, and Engle (1993) among others, the effects of a shock can take a considerable time to decay. Therefore, the distinction between I(0) and I(1) processes seems to be far too restrictive. Indeed, the propagation of shocks in an I(0) process occurs at an exponential rate of decay (so that it only captures the short-memory), while for an I(1) process the persistence of shocks is infinite. In the conditional mean, the ARFIMA specification has been proposed to fill the gap between short and complete persistence, so that the short-run behavior of the time-series is captured by the ARMA parameters, while the fractional differencing parameter allows for modelling the long-run dependence.

To mimic the behavior of the correlogram of the observed volatility, Baillie, Bollerslev, and Mikkelsen (1996) (hereafter denoted BBM) introduce the Fractionally Integrated GARCH (FIGARCH) model by replacing the first difference operator of Eq. (20) by $(1 - L)^d$.

The conditional variance of the FIGARCH ($p, d, q$) is given by:

$$\sigma_t^2 = \omega \frac{1 - \beta(L)}{[1 - \beta(L)]^{-1}} + \left\{1 - [1 - \beta(L)]^{-1}\phi(L)(1 - L)^d\right\}\varepsilon_t^2,$$

(21)

or $\sigma_t^2 = \omega^* + \sum_{i=1}^\infty \lambda_i L^i \varepsilon_t^2 = \omega^* + \lambda(L)\varepsilon_t^2$, with $0 \leq d \leq 1$. It is fairly easy to show that $\omega > 0, \beta_1 - d \leq \phi_1 \leq \frac{2 - d}{2}$ and $d \left(\phi_1 - \frac{1 - d}{2}\right) \leq \beta_1 (\phi_1 - \beta_1 + d)$ are sufficient to ensure that the

---

9 In their study of the daily S&P500 index, they find that the squared returns series has positive autocorrelations over more than 2,500 lags (or more than 10 years!).

10 See Bollerslev and Mikkelsen (1996, p.158) for a discussion on the importance of non-integer values of integration when modelling long-run dependencies in the conditional mean of economic time series.
The conditional variance of the FIGARCH \((1, d, 1)\) is positive almost surely for all \(t\). Setting \(\phi_1 = 0\) gives the condition for the FIGARCH \((1, d, 0)\).

Davidson (2001) notes the interesting and counterintuitive fact that the memory parameter of this process is \(-d\), and is increasing as \(d\) approaches zero, while in the ARFIMA model the memory increases when \(\zeta\) increases. According to Davidson (2001), the unexpected behavior of the FIGARCH model may be due less to any inherent paradoxes than to the fact that, embodying restrictions appropriate to a model in levels, it has been transplanted into a model of volatility. The main characteristic of this model is that it is not stationary when \(d > 0\). Indeed,

\[
(1 - L)^d = \sum_{k=0}^{\infty} \frac{\Gamma(d + 1)}{\Gamma(k + 1) \Gamma(d - k + 1)} L^k
\]

\[
= 1 - dL - \frac{1}{2} d(1 - d)L^2 - \frac{1}{6} d(1 - d)(2 - d)L^3 - \ldots
\]

\[
= 1 - \sum_{k=1}^{\infty} c_k(d) L^k,
\]

where \(c_1(d) = d, c_2(d) = \frac{1}{2} d(1 - d), \ldots\). By construction, \(\sum_{k=1}^{\infty} c_k(d) = 1\) for any value of \(d\), and consequently, the FIGARCH belongs to the same “knife-edge-nonstationary” class represented by the IGARCH. To test whether this nonstationarity feature holds, Davidson (2001) proposes a generalized version of the FIGARCH and calls it the HYperbolic GARCH. The HYGARCH is given by Eq. (21), when \(\lambda(L)\) is replaced by \(1 - [1 - \beta(L)]^{-1} \phi(L) \{1 + \alpha [(1 - L)^d]\}\). Note that we report \(\ln(\alpha)\) and not \(\alpha\). The \(c_k(d)\) coefficients are thus weighted by \(\alpha\). Interestingly, the HYGARCH nests the FIGARCH when \(\alpha = 1\) (or equivalently when \(\ln(\alpha) = 0\)) and if the GARCH component observes the usual covariance stationarity restrictions, then this process is stationary with \(\alpha < 1\) (or equivalently when \(\ln(\alpha) < 0\)) (see Davidson, 2001 for more details).

Chung (1999) underscores some little drawbacks in the BBM model: there is a structural problem in the BBM specification since the parallel with the ARFIMA framework of the conditional mean equation is not perfect, leading to difficult interpretations of the estimated parameters. Indeed the fractional differencing operator applies to the constant term in the mean equation (ARFIMA) while it does not in the variance equation (FIGARCH). Chung (1999) proposes a slightly different process:

\[
\phi(L)(1 - L)^d (\varepsilon_t^2 - \sigma^2) = [1 - \beta(L)](\varepsilon_t^2 - \sigma_t^2),
\]

where \(\sigma^2\) is the unconditional variance of \(\varepsilon_t\).

If we keep the same definition of \(\lambda(L)\) as in Eq. (21), we can formulate the conditional variance as:

\[
\sigma_t^2 = \sigma^2 + \left\{1 - [1 - \beta(L)]^{-1} \phi(L)(1 - L)^d\right\} (\varepsilon_t^2 - \sigma^2)
\]

or

\[
\sigma_t^2 = \sigma^2 + \lambda(L) (\varepsilon_t^2 - \sigma^2).
\]

\(\lambda(L)\) is an infinite summation which, in practice, has to be truncated. BBM propose to truncate \(\lambda(L)\) at 1000 lags (this truncation order has been implemented as the default value
in our package, but it may be changed by the user) and initialize the unobserved $\varepsilon_t^2$ at their unconditional moment. Contrary to BBM, Chung (1999) proposes to truncate $\lambda(L)$ at the size of the information set $(t-1)$ and to initialize the unobserved $(\varepsilon_t^2 - \sigma_t^2)$ at 0 (this quantity is small in absolute values and has a zero mean).\textsuperscript{11}

The idea of fractional integration has been extended to other GARCH types of models, including the Fractionally Integrated EGARCH (FIEGARCH) of Bollerslev and Mikkelsen (1996) and the Fractionally Integrated APARCH (FIAPARCH) of Tse (1998).\textsuperscript{12}

Similarly to the GARCH$(p, q)$ process, the EGARCH$(p, q)$ of Eq. (11) can be extended to account for long memory by factorizing the autoregressive polynomial $[1 - \beta(L)] = \phi(L)(1 - L)^d$ where all the roots of $\phi(z) = 0$ lie outside the unit circle. The FIEGARCH $(p, d, q)$ is specified as follows:

$$
\ln(\sigma_t^2) = \omega + \phi(L)^{-1} (1 - L)^{-d} [1 + \alpha(L)]g(z_{t-1}).
$$

Finally, the FIAPARCH $(p, d, q)$ model can be written as:\textsuperscript{13}

$$
\sigma_t^\delta = \omega + \left\{1 - [1 - \beta(L)]^{-1} \phi(L) (1 - L)^d \right\} (|\varepsilon_t| - \gamma \varepsilon_t)^\delta.
$$

\textsuperscript{11}See Chung (1999) for more details.

\textsuperscript{12}Notice that the GJR has not been extended to the long-memory framework. It is however nested in the FIAPARCH class of models.

\textsuperscript{13}When using the BBM option in G@ARCH for the FIEGARCH and FIAPARCH, $(1-L)^d$ and $(1-L)^{-d}$ are truncated at some predefined value (see above). It is also possible to truncate this polynomial at the information size at time $t$, i.e. $t-1$. 

35
4 Estimation Methods

4.1 Parameters Constraints

When numerical optimization is used to maximize the log-likelihood function with respect to the vector of parameters $\Psi$, the inspected range of the parameter space is $[-\infty; \infty[$. The problem is that some parameters might have to be constrained in a smaller interval. For instance, the leverage effect parameter $\gamma$ of the APARCH model must lie between -1 and 1. To impose these constraints one could estimate $\Psi^*$ (which ranges from $-\infty$ to $+\infty$) instead of $\Psi$ where $\Psi$ is recovered using the non-linear function: $\Psi = x(\Psi^*)$. In our package, $x(.)$ is defined as:

$$x(\Psi^*) = \text{Low} + \frac{\text{Up} - \text{Low}}{1 + e^{-\Psi^*}},$$

(27)

where Low is the lower bound and Up the upper bound (i.e. in our example, Low = -1 and Up = 1).

So, applying unconstrained optimization of the log-likelihood function with respect to $\Psi$ is equivalent to applying constrained optimization with respect to $\Psi^*$. Therefore, the optimization process of the program results in $\hat{\Psi}^*$ with the covariance matrix being noted $\text{Cov}(\hat{\Psi}^*)$. The estimated covariance of the parameters of interest $\hat{\Psi}$ is:

$$\text{Cov}(\hat{\Psi}) = \left( \frac{\partial x(\hat{\Psi}^*)}{\partial \Psi^*} \right) \text{Cov}(\hat{\Psi}^*) \left( \frac{\partial x(\hat{\Psi}^*)}{\partial \Psi^*} \right)^T.$$  

(28)

In our case, we have $\text{Cov}(\hat{\Psi}) = \text{Cov}(\hat{\Psi}^*) \exp(-\Psi^*)(\text{Up} - \text{Low})/\left[1 + \exp(-\Psi^*)\right]^2$. Note that, in G@RCH 2.3, lower and upper bounds of the parameters can be easily modified by the user in the file startingvalues.txt.

4.2 Distributions

Weiss (1986) and Bollerslev and Wooldridge (1992) show that under the normality assumption, the quasi maximum likelihood estimator is consistent if the conditional mean and the conditional variance are correctly specified. This estimator is, however, inefficient with the degree of inefficiency increasing with the degree of departure from normality (Engle and González-Rivera, 1991). Since the issue of fat-tails is an important one in empirical finance, it may be expected that using a more appropriate distribution would reduce the excess kurtosis displayed by the residuals of conditional heteroscedasticity models. As reported by Palm (1996), Pagan (1996) and Bollerslev, Chou, and Kroner (1992), the use of a fat-tailed distributions is widespread in the literature. In particular, Bollerslev (1987), Hsieh (1989), Baillie and Bollerslev (1989) and Palm and Vlaar (1997) among others show that these distributions perform better in order to capture the higher observed kurtosis.

Four distributions are available in our program: the usual Gaussian (normal) distribution, the Student-t distribution, the Generalized Error Distribution (GED) and the skewed Student-t distribution.
The GARCH models are estimated using a maximum likelihood (ML) approach. The logic of
ML is to interpret the density as a function of the parameters set, conditional on a set of sample
outcomes. This function is called the likelihood function. It is quite evident from equation (6) (and
all the following equations of Section 3) that the recursive evaluation of this function is conditional
on unobserved values. The ML estimation is therefore not perfectly exact. To solve the problem
of unobserved values, we have set these quantities to their unconditional expected values.

If we express the mean equation as in Eq. (1) and $\varepsilon_t = z_t \sigma_t$, the log-likelihood function of
the standard normal distribution is given by:

$$L_{\text{norm}} = -\frac{1}{2} \sum_{t=1}^{T} \left[ \ln (2\pi) + \ln (\sigma_t^2) + z_t^2 \right],$$

(29)

where $T$ is the number of observations.

For a Student-$t$ distribution, the log-likelihood is:

$$L_{\text{Stud}} = T \left\{ \ln \Gamma \left( \frac{v+1}{2} \right) - \ln \Gamma \left( \frac{v}{2} \right) - \frac{1}{2} \ln \left[ \pi (v-2) \right] \right\} - \frac{1}{2} \sum_{t=1}^{T} \left[ \ln (\sigma_t^2) + (1+v) \ln \left( 1 + \frac{z_t^2}{v-2} \right) \right],$$

(30)

where $v$ is the degrees of freedom, $2 < v \leq \infty$ and $\Gamma(.)$ is the gamma function.

The GED log-likelihood function of a normalized random variable is given by:

$$L_{\text{GED}} = \sum_{t=1}^{T} \left[ \ln (v/\lambda_v) - 0.5 \left| \frac{z_t}{\lambda_v} \right|^v - (1+v) \ln (1/v) - 0.5 \ln (\sigma_t^2) \right],$$

(31)

where $0 < v < \infty$ and

$$\lambda_v = \sqrt{\frac{\Gamma \left( \frac{1}{v} \right) 2^{-\frac{2}{v}}}{\Gamma \left( \frac{2}{v} \right)}}.$$

(32)

The main drawback of the last two densities is that even if they may account for fat tails, they
are symmetric. Skewness and kurtosis are important in financial applications in many respects
(in asset pricing models, portfolio selection, option pricing theory or Value-at-Risk among others).
Quite recently, Lambert and Laurent (2000, 2001) applied and extended the skewed Student
density proposed by Fernández and Steel (1998) to the GARCH framework.

If $\Gamma(.)$ denotes the gamma function, the log-likelihood of a standardized (zero mean and unit
variance) skewed Student is:

$$L_{\text{SkSt}} = T \left\{ \ln \Gamma \left( \frac{v+1}{2} \right) - \ln \Gamma \left( \frac{v}{2} \right) - 0.5 \ln \left[ \pi (v-2) \right] + \ln \left( \frac{2}{\xi + \xi} \right) + \ln (s) \right\}$$

$$- 0.5 \sum_{t=1}^{T} \left[ \ln \sigma_t^2 + (1+v) \ln \left[ 1 + \frac{(sz_t + m)^2}{v-2} \xi - 2\xi \right] \right],$$

(33)

where $I_t = \begin{cases} 1 & \text{if } z_t \geq -\frac{m}{s} \\ -1 & \text{if } z_t < -\frac{m}{s} \end{cases}$, $\xi$ is the asymmetry parameter, $v$ is the degree of freedom of the
distribution, \( m = \frac{\Gamma\left(\frac{\nu+1}{2}\right) \sqrt{n-2}}{\sqrt{\pi (\nu/2)}} \left( \xi - \frac{1}{\xi} \right) \) and \( s = \sqrt{\left( \xi^2 + \frac{1}{\xi^2} - 1 \right) - m^2} \). See Lambert and Laurent (2001) for more details.

In principal, the gradient vector and the hessian matrix can be obtained numerically or by evaluating its analytic expressions. Due to the high number of possible models and distributions, we use numerical techniques to approximate the derivatives of the log-likelihood function with respect to the parameter vector.

4.3 Tests

In addition to the possibilities offered by GiveWin (ACF, PACF, QQ-plots...), several tests are provided in the G@RCH package:

- Four Information Criteria (divided by the number of observations): \(^{14}\)
  - Akaike = \(-2 \frac{\text{Log}L}{n} + 2 \frac{k}{n}\)
  - Hannan-Quinn = \(-2 \frac{\text{Log}L}{n} + 2 \frac{k \log \log(n)}{n}\)
  - Schwartz = \(-2 \frac{\text{Log}L}{n} + 2 \frac{\log(k)}{n}\)
  - Shibata = \(-2 \frac{\text{Log}L}{n} + \log \left( \frac{n+2k}{n} \right)\)

- The value of the skewness and the kurtosis of the standardized residuals (\(\hat{z}_t\)) of the estimated model, their \(t\)-tests and \(p\)-values. Moreover, the Jarque-Bera normality test (Jarque and Bera, 1987) is also reported.

- The Box-Pierce statistics at lag \(l^*\) for both standardized, i.e. \(BP (l^*)\), and squared standardized, i.e. \(BP^2 (l^*)\), residuals. Under the null hypothesis of no autocorrelation, the statistics \(BP (l^*)\) and \(BP^2 (l^*)\) should be evaluated against the \(\chi^2 (l^* - m - l)\) and \(\chi^2 (l^* - p - q)\), respectively (see McLeod and Li, 1983).

- The Engle LM ARCH test (Engle, 1982) to test the presence of ARCH effects in a series.

- The diagnostic test of Engle and Ng (1993) that investigate possible misspecification of the conditional variance equation. The Sign Bias Test (SBT) examines the impact of positive and negative return shocks on volatility not predicted by the model under construction. The negative Size Bias Test (resp. positive Size Bias Test) focuses on the different effects that large and small negative (resp. positive) return shocks have on volatility, which is not predicted by the volatility model. Finally, a joint test for these three tests is also provided.

- The adjusted Pearson goodness-of-fit test that compares the empirical distribution of the innovations with the theoretical one. In order to carry out this testing procedure, it is necessary to first classify the residuals in cells according to their magnitude.\(^{15}\) Let \(n\) be the number of observations, \(r\) the number of categories we consider, \(p_i\) (\(i = 1, \ldots, r\)) the observed

\(^{14}\) \(\text{Log}L = \) log likelihood value, \(n = \# \) observations and \(k\) the number of estimated parameters.

\(^{15}\) See Palm and Vlaar (1997) for more details.
proportion of observations being in the \( i^{th} \) category and \( p_i \) \((i = 1,...,r)\) the theoretical probability for an observation to be in the \( i^{th} \) category. The Pearson goodness-of-fit test has the null \( H_0: p_1 = p_{t1}, p_2 = p_{t2}, \ldots, p_r = p_{tr}. \) The statistic is computed as

$$P(g) = \sum_{i=1}^{r} \frac{(n_i - En_i)^2}{En_i}$$

(34)

where \( n_i \) is the observed number in the sample that fall into the \( i^{th} \) category and \( En_i \) is the number of observations expected to be in this \( i^{th} \) category when \( H_0 \) is true. The Pearson statistic is therefore “small” when all of the observed counts (proportions) are close to the expected counts (proportions) and it is “large” when one or more observed counts (proportions) differs noticeably from what is expected when \( H_0 \) is true. For \( i.i.d. \) observations, Palm and Vlaar (1997) show that under the null of a correct distribution the asymptotic distribution of \( P(g) \) is bounded between a \( \chi^2(r - 1) \) and a \( \chi^2(r - k - 1) \) where \( k \) is the number of estimated parameters. As explained by Palm and Vlaar (1997), the choice of \( r \) is far from being obvious. For \( T = 2252 \), these authors set \( r \) equal to 50. According to König and Gaab (1982), the number of cells must increase at a rate equal to \( T^{0.4} \).

- The Nyblom test (Nyblom, 1989 and Lee and Hansen, 1994) to check the constancy of parameters over time. See also Hansen (1994) for an overview of this test.

### 4.4 Forecasts

Estimating a model can be useful to try to understand the mechanism that produces the series of interest. It can also suggest a solution to an economic problem. Is it the only game in town? Certainly not. Indeed, the main purpose of building and estimating a model with financial data is to produce a forecast. G@RCH 2.3 also provides forecasting tools. Indeed, forecasts of both the conditional mean and the conditional variance are available as well as several forecast error measures.

#### 4.4.1 Forecasting the conditional mean

Our first goal is to give the optimal \( h \)-step-ahead predictor of \( y_{t+h} \) given the information we have up to time \( t \).

For instance, for the following AR(1) process,

$$y_t = \mu + \psi_1 (y_{t-1} - \mu) + \varepsilon_t.$$  

(35)

The optimal\(^{17}\) \( h \)-step-ahead predictor of \( y_{t+h} \), i.e. \( \hat{y}_{t+h|t} \), is its conditional expectation at time \( t \) (given the estimated parameters \( \hat{\mu} \) and \( \hat{\psi}_1 \)):

$$\hat{y}_{t+h|t} = \hat{\mu} + \hat{\psi}_1 (\hat{y}_{t+h-1|t} - \hat{\mu}),$$

(36)

---

\(^{16}\)Large values of GoF suggest therefore that \( H_0 \) is false.

\(^{17}\)By optimal, we mean optimal under expected quadratic loss, or in a mean square error sense.
where \( \hat{y}_{t+i|t} = y_{t+i} \) for \( i \leq 0 \).

For the AR(1), the optimal 1-step-ahead forecast equals \( \hat{\mu} + \hat{\psi}_1(\hat{y}_t - \hat{\mu}) \). For \( h > 1 \), the optimal forecast can be obtained recursively or directly as \( \hat{y}_{t+h|t} = \hat{\mu} + \hat{\psi}_1^h(\hat{y}_t - \hat{\mu}) \).

In the general case of an ARFIMA\((n, \zeta, s)\) as given in Eq. (3), the optimal \( h \)-step-ahead predictor of \( y_{t+h} \) is:

\[
\hat{y}_{t+h|t} = \left[ \hat{\mu}_{t+h|t} + \sum_{k=1}^{\infty} \hat{c}_k (\hat{y}_{t+h-k} - \hat{\mu}_{t+h|t}) \right] + \sum_{i=1}^n \hat{\psi}_i \left[ \hat{y}_{t+h-i} - \left[ \hat{\mu}_{t+h|t} + \sum_{k=1}^{\infty} \hat{c}_k (\hat{y}_{t+h-i-k} - \hat{\mu}_{t+h|t}) \right] \right] + \sum_{j=1}^s \hat{\theta}_j (\hat{y}_{t+h-j} - \hat{y}_{t+h-j|t}).
\] (37)

Recall that when exogenous variables enter the conditional mean equation, \( \mu \) becomes \( \mu_t = \mu + \sum_{i=1}^{n_1} \delta_i x_{it} \) and consequently, provided that the information \( x_{it+h} \) is available at time \( t \) (which is the case for instance if \( x_{it} \) is a “day-of-the-week” dummy variable), \( \hat{\mu}_{t+h|t} \) is also available at time \( t \).

When there is no exogenous variable in the ARFIMA model and \( n = 1, s = 0 \) and \( \zeta = 0 \) (\( c_k = 0 \)), the forecast of the AR(1) process given in Eq. (36) can be recovered.

### 4.4.2 Forecasting the conditional variance

Independently from the conditional mean, one can forecast the conditional variance. In the simple GARCH\((p, q)\) case, the optimal \( h \)-step-ahead forecast of the conditional variance, i.e. \( \hat{\sigma}^2_{t+h|t} \) is given by:

\[
\hat{\sigma}^2_{t+h|t} = \hat{\omega} + \sum_{i=1}^q \hat{\alpha}_i \hat{\varepsilon}^2_{t+i-h-i|t} + \sum_{j=1}^p \hat{\beta}_j \hat{\sigma}^2_{t+h-j|h|t},
\] (38)

where \( \hat{\varepsilon}^2_{t+i|t} = \sigma^2_{t+i|t} \) for \( i > 0 \) while \( \hat{\varepsilon}^2_{t+i|t} = \hat{\varepsilon}_{t+i}^2 \) and \( \sigma^2_{t+i|t} = \sigma^2_{t+i} \) for \( i \leq 0 \). Eq. (38) is usually computed recursively, even if a closed form solution of \( \sigma^2_{t+h|t} \) can be obtained by recursive substitution in Eq. (38).

Similarly, one can easily obtain the \( h \)-step-ahead forecast of the conditional variance of an ARCH, IGARCH and FIGARCH model. By contrast, for thresholds models, the computation of the out-of-sample forecasts is more complicated. Indeed, for the EGARCH, GJR and APARCH models (as well as for their long-memory counterparts), the assumption made on the innovation process may have an effect on the forecast (especially for \( h > 1 \)).

For instance, for the GJR \((p, q)\) model,

\[
\hat{\sigma}^2_{t+h|t} = \hat{\omega} + \sum_{i=1}^q (\hat{\alpha}_i \hat{\varepsilon}^2_{t-i+h-i|t} + \hat{\gamma}_i S^+_{t-i+h|t} \hat{\varepsilon}^2_{t-i+h|t}) + \sum_{j=1}^p \hat{\beta}_j \hat{\sigma}^2_{t-j+h|t},
\] (39)

When all the \( \gamma_i \) parameters equal 0, one recovers the forecast of the GARCH model. Otherwise, one has to compute \( S^-_{t-i+h|t} \). Note first that \( S^-_{t+i|t} = S^-_{t+i} \) for \( i \leq 0 \). However, when \( i > 1 \), \( S^-_{t+i|t} \)
depends on the choice of the distribution of \( z_t \). When the distribution of \( z_t \) is symmetric around 0 (for the Gaussian, Student and GED density), the probability that \( \varepsilon_{t+i} \) will be negative is \( S_{t+i|t}^{-} \approx 0.5 \). If \( z_t \) is (standardized) skewed Student distributed with asymmetry parameter \( \xi \) and degree of freedom \( \nu \), \( S_{t+i|t}^{-} = \frac{1}{1+\nu} \) since \( \xi^2 \) is the ratio of probability masses above and below the mode.

For the APARCH \((p,q)\) model,

\[
\sigma^\delta_{t+h|t} = E \left( \sigma^\delta_{t+h|t} \mid \Omega_t \right) = E \left( \tilde{\omega} + \sum_{i=1}^{q} \tilde{\alpha}_i (|\varepsilon_{t+h-i} - \tilde{\gamma}_i \varepsilon_{t+h-i}|^{\delta} + \sum_{j=1}^{p} \tilde{\beta}_j \sigma^\delta_{t+h-j} \mid \Omega_t \right)
\]

\[
= \tilde{\omega} + \sum_{i=1}^{q} \tilde{\alpha}_i E \left[ (|\varepsilon_{t+h-i} - \tilde{\gamma}_i \varepsilon_{t+h-i}|^{\delta} \mid \Omega_t \right] + \sum_{j=1}^{p} \tilde{\beta}_j \sigma^\delta_{t+h-j|t}, \tag{40}
\]

where \( E \left[ (|\varepsilon_{t+k} - \tilde{\gamma}_i \varepsilon_{t+k}|^{\delta} \mid \Omega_t \right] = \kappa_i \sigma^\delta_{i|t} \), for \( k > 1 \) and \( \kappa_i = E (|z| - \gamma_i z)^{\delta} \) (see Section 4.2).

For the EGARCH \((p,q)\) model,

\[
\ln \sigma^2_{t+h|t} = E \left( \ln \sigma^2_{t+h} \mid \Omega_t \right) = E \left\{ \tilde{\omega} + \left[ 1 - \tilde{\beta}(L) \right]^{-1} \right\} \ln \sigma^2_{t+h|t} + \left[ 1 - \tilde{\beta}(L) \right] \tilde{\omega} + \tilde{\beta}(L) \ln \sigma^2_{t+h|t} + \left[ 1 + \tilde{\alpha}(L) \right] \hat{g}(z_{t+h-1|t}), \tag{41}
\]

where \( \hat{g}(z_{t+h|t}) = \hat{g}(z_{t+k}) \) for \( k \leq 0 \) and 0 for \( k > 0 \).

Finally, the \( h \)-step-ahead forecast of the FIAPARCH and FIEGARCH models are obtained in a similar way.

One of the most popular measures to check the forecasting performance of the ARCH-type models is the Mincer-Zarnowitz regression, i.e. ex-post volatility regression:

\[
\hat{\sigma}^2_t = a_0 + a_1 \hat{\sigma}^2_t + u_t, \tag{42}
\]

where \( \hat{\sigma}^2_t \) is the ex-post volatility, \( \hat{\sigma}^2_t \) is the forecasted volatility and \( a_0, a_1 \) are parameters to be estimated. If the model for the conditional variance is correctly specified (and the parameters are known) and \( E(\hat{\sigma}^2_t) = \hat{\sigma}^2_t \), it follows that \( a_0 = 0 \) and \( a_1 = 1 \). The \( R^2 \) of this regression is often used as a simple measure of the degree of predictability of the ARCH-type model.

However, \( \hat{\sigma}^2_t \) is never observed. By default, G@RCH 2.3 uses \( \hat{\sigma}^2_t = (y_t - \bar{y})^2 \), where \( \bar{y} \) is the sample mean of \( y_t \). The \( R^2 \) of this regression is often lower than 5% and this could lead to the conclusion that GARCH models produce poor forecasts of the volatility (see, among others, Schwert, 1990, or Jorion, 1996). But, as described in Andersen and Bollerslev (1998), the reason of these poor results is the choice of what is considered as the “true” volatility. G@RCH 2.3 allows to select any series as the “observed” volatility (Obs.-Var., see Figure 1). The user may then compute the daily realized volatility as the sum of squared intraday returns and use it as the “true” volatility. Actually, Andersen and Bollerslev (1998) show that this measure is a more
proper one than squared daily returns. Therefore, using 5-minute returns for instance, the realized volatility can be expressed as:

$$\sigma_t^2 = \sum_{k=1}^{K} y_{k,t}^2,$$

where $y_{k,t}$ is the return of the $k^{th}$ 5-minutes interval of the $t^{th}$ day and $K$ is the number of 5-minutes intervals per day.

Finally, to compare the adequacy of the different distributions, GARCH 2.3 also allows the computation of density forecasts tests developed in Diebold, Gunther, and Tay (1998). The idea of density forecasts is quite simple.\(^{18}\) Let $f_i(y_i|\Omega_i)_{i=1}^{m}$ be a sequence of $m$ one-step-ahead density forecasts produced by a given model, where $\Omega_i$ is the conditioning information set, and $p_i(y_i|\Omega_i)_{i=1}^{m}$ the sequence of densities defining the Data Generating Process $y_i$ (which is never observed). Testing whether this density is a good approximation of the true density $p(.)$ is equivalent to testing:

$$H_0 : f_i(y_i|\Omega_i)_{i=1}^{m} = p_i(y_i|\Omega_i)_{i=1}^{m} \quad (44)$$

Diebold, Gunther, and Tay (1998) use the fact that, under Eq. (44), the probability integral transform $\hat{\zeta}_i = \int_{-\infty}^{y_i} f_i(t)dt$ is i.i.d. $U(0, 1)$, i.e. independent and identically distributed uniform. To check $H_0$, they propose to use goodness-of-fit test and independence test for i.i.d. $U(0, 1)$. The i.i.d.-ness property of $\hat{\zeta}_i$ can be evaluated by plotting the correlograms of $(\hat{\zeta}_i - \overline{\zeta})^j$, for $j = 1, 2, 3, 4, ...$, to detect potential dependence in the conditional mean, variance, skewness, kurtosis, etc. Departure from uniformity can also be evaluated by plotting an histogram of $\hat{\zeta}_i$.

According to Bauwens, Giot, Grammig, and Veredas (2000), a humped shape of the $\hat{\zeta}$-histogram would indicate that the issued forecasts are too narrow and that the tails of the true density are not accounted for. On the other hand, a U-shape of the histogram would suggest that the model issues forecasts that either under- or overestimate too frequently. Moreover, Lambert and Laurent (2001) show that an inverted S shape of the histogram would indicate that the errors are skewed, i.e. the true density is probably not symmetric.\(^{19}\) An illustration is provided in Section 5 with some formal tests and graphical tools.

4.5 Accuracy

McCullough and Vinod (1999) and Brooks, Burke, and Persand (2001) use the daily German mark/British pound exchange rate data of Bollerslev and Ghysels (1996) to compare the accuracy of GARCH model estimation among several econometric softwares. They choose the GARCH(1,1) model described in Fiorentini, Calzolari, and Panattoni (1996) (hereafter denoted FCP) as the benchmark. In this section, we use the same methodology with the same dataset to check the

\(^{18}\)For more details about density forecasts and applications in finance, see the special issue of Journal of Forecasting (Timmermann, 2000).

\(^{19}\)Confidence intervals for the $\hat{\zeta}$-histogram can be obtained by using the properties of the histogram under the null hypothesis of uniformity.
accuracy of our procedures. Coefficients and standard errors estimates of G@RCH 2.3 are reported in Table 1 together with the results of McCullough and Vinod (1999) (FCP in the table).

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Hessian</th>
<th>QMLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>G@RCH</td>
<td>FCP</td>
<td>G@RCH</td>
</tr>
<tr>
<td>$\mu$</td>
<td>-0.006184</td>
<td>-0.006190</td>
</tr>
<tr>
<td>$\omega$</td>
<td>0.010760</td>
<td>0.010761</td>
</tr>
<tr>
<td>$\alpha_1$</td>
<td>0.153407</td>
<td>0.153134</td>
</tr>
<tr>
<td>$\beta_1$</td>
<td>0.805879</td>
<td>0.805974</td>
</tr>
</tbody>
</table>

Table 1: Accuracy of the GARCH procedure

G@RCH gives very satisfactory results since the first four digits (at least) are the same as those of the benchmark for all but two estimations. In addition, it competes well compared to other well known econometric softwares: Table 2 gives indeed the coefficient estimates and the error percentage associated for 5 softwares. G@RCH, PcGive and TSP (which uses the analytical gradients for the GARCH(1,1) model) clearly outperform Eviews and S-Plus on this specification.

<table>
<thead>
<tr>
<th></th>
<th>FCP</th>
<th>G@RCH</th>
<th>Eviews</th>
<th>PcGive</th>
<th>TSP</th>
<th>S-Plus</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu$</td>
<td>-0.00619</td>
<td>-0.00618</td>
<td>-0.00541</td>
<td>-0.00625</td>
<td>-0.00619</td>
<td>-0.00919</td>
</tr>
<tr>
<td>$\omega$</td>
<td>0.010760</td>
<td>0.010760</td>
<td>0.009581</td>
<td>0.010760</td>
<td>0.010760</td>
<td>0.011696</td>
</tr>
<tr>
<td>$\alpha_1$</td>
<td>0.153134</td>
<td>0.153407</td>
<td>0.142284</td>
<td>0.153397</td>
<td>0.153134</td>
<td>0.154295</td>
</tr>
<tr>
<td>$\beta_1$</td>
<td>0.805974</td>
<td>0.805879</td>
<td>0.821336</td>
<td>0.805886</td>
<td>0.805974</td>
<td>0.800276</td>
</tr>
</tbody>
</table>

|            | -    | 0.10% | 12.58% | 0.91% | 0.00% | 48.41% |
| $\mu$      | -    | 0.01% | 10.96% | 0.01% | 0.00% | 8.69%  |
| $\omega$   | -    | 0.18% | 7.08%  | 0.17% | 0.00% | 0.76%  |
| $\alpha_1$ | -    | 0.01% | 1.91%  | 0.01% | 0.00% | 0.71%  |

Table 2: GARCH Accuracy Comparison

Moreover, to investigate the accuracy of our forecasting procedures, we have run a 8-step ahead
forecasts of the model, similar to Brooks, Burke, and Persand (2001). Table 4 in Brooks, Burke, and Persand (2001) reports the conditional variance forecasts given by six well-known softwares and the correct values. Contrary to E-VIEWS, Matlab and SAS, G@RCH hits the benchmarks for all steps to the third decimal (note that GAUSS, Microfit and Rats also do).

Finally, Lombardi and Gallo (2001) extends the work of Fiorentini, Calzolari, and Panattoni (1996) to the FIGARCH model of Baillie, Bollerslev, and Mikkelsen (1996) and develops the analytic Hessian matrices of this long memory process. For the same DEM/UKP database as in the previous example, Table 3 reports the coefficients estimates and their standard errors for our package (using numerical gradients and the BFGS optimization method) and for Lombardi and Gallo (2001) (using analytical gradients and the Newton-Raphson algorithm).

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Hessian</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>G@RCH</td>
</tr>
<tr>
<td>µ</td>
<td>0.003606</td>
</tr>
<tr>
<td>ω</td>
<td>0.015772</td>
</tr>
<tr>
<td>α₁</td>
<td>0.198134</td>
</tr>
<tr>
<td>β₁</td>
<td>0.675652</td>
</tr>
<tr>
<td>d</td>
<td>0.570702</td>
</tr>
</tbody>
</table>

Table 3: Accuracy of the FIGARCH procedure

Results show that G@RCH provides excellent numerical estimates that are quite close to the analytical ones, even for an advanced model such as the FIGARCH. As expected, it is however more time-consuming than the C code of Lombardi and Gallo (2001)\(^\text{20}\) (163 sec. vs 43 sec. using a Dell PC with a PIII processor).

4.6 Features Comparison

The goal of this section is to compare objectively, in a GARCH perspective, the features offered by G@RCH 2.3 and nine other econometric softwares, namely PcGive 10 (also programmed in Ox), GAUSS and its Fan pac extension, Eviews 4, S-Plus 6 and its GARCH module, Rats and its garch.src example\(^\text{21}\), TSP 4.5, Microfit 4, SAS 8.2 and Stata 7. It is thus not our intention to evaluate a program against another, but we will rather show an overview of what you can or cannot do with these softwares.

\(^{20}\)This C code is available at http://www.ds.unifi.it/~mj1/ in the ”software” section.

\(^{21}\)This file is available at http://www.estima.com/procindex.htm for download.
<table>
<thead>
<tr>
<th></th>
<th>G@RCH</th>
<th>PcGive</th>
<th>Fanpac</th>
<th>Eviews</th>
<th>S-Plus</th>
<th>Rats</th>
<th>TSP</th>
<th>Microfit</th>
<th>SAS</th>
<th>Stata</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Version</strong></td>
<td>2.3</td>
<td>10</td>
<td>-</td>
<td>4.0</td>
<td>6</td>
<td>5.0</td>
<td>4.5</td>
<td>4</td>
<td>8.2</td>
<td>7</td>
</tr>
<tr>
<td><strong>Conditional mean</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Explanatory variables</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>ARMA</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>ARFIMA</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ARCH-in-Mean</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td><strong>Conditional variance</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Explanatory variables</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>GARCH</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>IGARCH</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>EGARCH</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>GJR</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>APARCH</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>C-GARCH</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FIGARCH</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FIEGARCH</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FIAPARCH</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>HYGARCH</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Distributions</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normal</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Student-t</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>GED</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Skewed-t</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Double Exponential</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Estimation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MLE</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>QMLE</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

A “+” (resp. “-”) means that the corresponding option is (resp. is not) available for this software. C-GARCH corresponds to the Component GARCH of Engle and Lee (1999).
The proposed models and options differ widely from one program to the other as can be seen in Table 4. Regarding the range of different univariate models\(^{22}\), if many programs propose asymmetric models, very few (G@RCH, S-Plus with the FIGARCH BBM and the FIEGARCH and Fanpac with only the FIGARCH BBM) offer long memory models in the variance equation and none (except G@RCH) offers fractionally integrated specification in the mean. As for the distribution, G@RCH and S-Plus are the only softwares that permit the use of four densities. Finally, robust standard errors are proposed in 5 programs out of the 10 we have compared (G@RCH, PeGive, GAUSS Fanpac, Eviews and Stata).

In summary, on a GARCH point of view, G@RCH 2.3 offers many more possibilities than most of the reviewed programs. S-Plus has certainly the most complete range of options among G@RCH competitors, while TSP and Microfit have quite poor GARCH features.

---

\(^{22}\)For the multivariate models, only FANPAC and S-Plus currently provide such models. The inclusion of multivariate models in G@RCH is currently under way.
5 Application

5.1 Data and Methodology

To illustrate our G@RCH 2.3 package with a concrete application, we analyze the French CAC40 stock index for the years 1995-1999 (1249 daily observations). It is computed by the exchange as a weighted measure of the prices of its components and is available in the database on an intraday basis with the price index being computed every 15 minutes. For the time period under review, the opening hours of the French stock market were 10.00 am to 5.00 pm, thus 7 hours of trading per day. This translates into 28 intraday returns used to compute the daily realized volatility. Intraday prices are the outcomes of a linear interpolation between the closest recorded prices below and above the time set in the grid. Correspondingly, all returns are computed as the first difference in the regularly time-spaced log prices of the index. Because the exchange is closed from 5.00 pm to 10.00 am the next day, the first intraday return is the first difference between the log price at 10.15 am and the log price at 5.00 pm the day before. Then, the intraday data are used to compute the daily realized volatility using Eq. (43). Finally, daily returns in percentage are defined as 100 times the first difference of the log of the closing prices.\footnote{By definition and using the properties of the log distribution, the sum of the intraday returns is equal to the observed daily return based on the closing prices.}

The estimation of the parameters is carried out for the 800 observations while forecasting is computed for the last observations.

5.2 Using the “Full Version”

First, open the database you want to use in GiveWin and select the OxPack module. Then select Add/Remove Package... in the Package menu. Click on the Browse button, then find and select garch.oxo. Click on the Add button. The Garch class is now active.

Then, select Package/Garch. There are two model classes in G@RCH 2.3 : Garch Models and Dataset Statistics. This second option allows to run several tests (general statistics, ARCH test and Box-Pierce test) on the raw series of the dataset. It is thus similar to the TESTSONLY function of the ”Light Version”.

For this application, select Garch Models and Model/Estimate is automatically launched. The list of all the variables of the database appears in the Database section (see figure 1). There are four possible statuses for each variable: dependent variable (Y variable), regressor in the mean (Mean), regressor in the variance (Variance) or observed volatility (Obs. Var.). Note that, in the example, we include the observed volatility of the series. Our program provides estimations for univariate models\footnote{The extension of this package to multivariate GARCH models is currently under development.}, so only one Y variable per model is accepted. However one can include several regressors in the mean and the variance equations and the same variable can be a regressor in both equations.

[INSERT FIGURE 1]
Once the OK button is pressed, the Model Settings box automatically appears. This box allows to select the specification of the model: AR( FI)MA orders for the mean equation, GARCH orders, type of GARCH model for the variance equation and the distribution (figure 2). The default specification is an ARMA(0,0)-GARCH(1,1) with normal errors. In our application, we run a ARMA(1,0)-APARCH(1,1).

As explained in Section 4.1, it is possible to constrain the parameters to range between a lower and an upper bound by selecting the Bounded Parameters option. The defaults bounds can be changed in the startingvalues.txt file.

In the next window, we are asked to make a choice regarding the starting values (figure 3): we might (1) let the program choose the starting values, (2) enter them manually, element by element, or (3) enter the starting values in a vector form (the required form is “value1;value2;value3”).

The first method is obviously the easiest, and may be indicated for beginning users, since it prevents from entering aberrant values. If we want particular starting values for the estimation and if we do not know the sequence of the parameters in the parameter vector used in our program, the second method should be a solution. An advanced user knowing the program quite well may use the third option as it is faster to do than the previous one. Note that, in the output, the estimated parameters are notably given in a vector form, so that we can just copy the vector and paste it in this box for a subsequent estimation.

Then, the estimation method for standard deviations is selected: Maximum Likelihood (ML) or Quasi-Maximum Likelihood (QML) or both. In this box (see figure 4), one may also select the sample and some maximization options (such has the number of iterations between intermediary results prints) when clicking on the Options button.

When we click on the OK button, the estimation procedure is then launched and the program comes back to GiveWin. A new dialog box is launched if the starting values are entered manually. Let us assume that the element-by-element method has been selected. A new window appears (see figure 5) with all the possible parameters to be estimated. Depending on the specification, some parameters have a value, others have not. The user should replace only the former since they correspond to the parameters to be estimated for the specified model.

Once this step is completed, the program starts the iteration process. Depending on the options selected earlier, it prints intermediary iteration results or not. The final output is divided

---

25Note that these default values can be modified by the user. Indeed they are stored in the startingvalues.txt file installed with the package.

48
by default in two main parts: first, the model specification reminder; second, the estimated values and other useful statistics of the parameters. The output is given in the box “Output 1”.

**Output 1**

*SPECIFICATIONS*

Mean Equation: ARMA (1, 0) model.
No regressor in the mean.

Variance Equation: APARCH (1, 1) model.
No regressor in the variance.

The distribution is a Skewed Student distribution, with a tail coefficient of 15.72 and an asymmetry coefficient of -0.08751.

Strong convergence using numerical derivatives

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Std.Error</th>
<th>t-value</th>
<th>t-prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cst(M)</td>
<td>0.065337</td>
<td>0.037157</td>
<td>1.758</td>
</tr>
<tr>
<td>AR(1)</td>
<td>0.004704</td>
<td>0.037117</td>
<td>0.1267</td>
</tr>
<tr>
<td>Cst(V)</td>
<td>0.017498</td>
<td>0.013488</td>
<td>1.297</td>
</tr>
<tr>
<td>Beta1</td>
<td>0.947590</td>
<td>0.017776</td>
<td>46.93</td>
</tr>
<tr>
<td>Alpha1</td>
<td>0.038464</td>
<td>0.014188</td>
<td>2.64</td>
</tr>
<tr>
<td>Gammal</td>
<td>0.676364</td>
<td>0.348702</td>
<td>1.940</td>
</tr>
<tr>
<td>Delta</td>
<td>1.462837</td>
<td>0.535581</td>
<td>2.742</td>
</tr>
<tr>
<td>Asymmetry</td>
<td>-0.087512</td>
<td>0.054314</td>
<td>-1.611</td>
</tr>
<tr>
<td>Tail</td>
<td>15.718323</td>
<td>8.087414</td>
<td>1.944</td>
</tr>
</tbody>
</table>

No. Observations: 800

Mean (Y): 0.08103

Variance (Y): 1.27405

Log Likelihood: -1190.521

After the estimation of the model, new options are available in OxPack: Menu/Tests, Menu/Graphic Analysis, Menu/Forecasts, Menu/Exclusion Restrictions, Menu/Linear Restrictions and Menu/Store.

The Menu/Graphic Analysis option allows to plot different graphics (see Figure 6 for details). Just as any other graphs in the GiveWin environment, they can be easily edited (color, size,...) and exported in many formats (.eps, .ps, .wmf, .emf and .gwg). Figure 7 provides the graphs of the squared residuals and the conditional mean with a 95% confidence interval.

[INSERT FIGURES 6 and 7]

The Menu/Tests option allows to run different tests (see Section 4.2 for further explanations). It also allows to print the variance-covariance matrix of the estimated parameters (Figure 8). The results of these tests are printed in GiveWin. An example of output is reported in the next box (“Output 2”).

[INSERT FIGURE 8]

Recall that the estimations are based on the numerical evaluation of the gradients.
### TESTS:

<table>
<thead>
<tr>
<th>Statistic</th>
<th>t-value</th>
<th>t-prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skewness</td>
<td>-0.2135</td>
<td>2.47</td>
</tr>
<tr>
<td>Excess Kurtosis</td>
<td>0.4684</td>
<td>2.713</td>
</tr>
<tr>
<td>Jarque-Bera</td>
<td>13.39</td>
<td>13.39</td>
</tr>
</tbody>
</table>

Information Criterium (minimize)

| Akaike         | 2.998802 |
| Schwarz        | 3.051504 |

BOX-PIERCE:

- H0: No serial correlation ⇒ Accept H0 when prob. is High [Q < Chisq(lag)]
- Box-Pierce Q-statistics on residuals
  - P-values adjusted by 1 degree(s) of freedom
  - \( Q(10) = 14.47 \ [0.1064] \)
  - \( Q(20) = 21.67 \ [0.3012] \)
- Box-Pierce Q-statistics on squared residuals
  - P-values adjusted by 2 degree(s) of freedom
  - \( Q(10) = 9.887 \ [0.2731] \)
  - \( Q(20) = 16.13 \ [0.5838] \)

Diagnostic test based on the news impact curve (EGARCH vs. GARCH)

<table>
<thead>
<tr>
<th>Test</th>
<th>Prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sign Bias t-Test</td>
<td>0.98838</td>
</tr>
<tr>
<td>Negative Size Bias t-Test</td>
<td>0.14581</td>
</tr>
<tr>
<td>Positive Size Bias t-Test</td>
<td>0.62400</td>
</tr>
<tr>
<td>Joint Test for the Three Effects</td>
<td>5.13914</td>
</tr>
</tbody>
</table>

Joint Statistic of the Nyblom test of stability: 2.727

Individual Nyblom Statistics:

<table>
<thead>
<tr>
<th>Statistic</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Cst(M)</td>
<td>0.72438</td>
</tr>
<tr>
<td>AR(1)</td>
<td>0.68524</td>
</tr>
<tr>
<td>Cst(V)</td>
<td>0.51505</td>
</tr>
<tr>
<td>Beta1</td>
<td>0.42785</td>
</tr>
<tr>
<td>Alpha1</td>
<td>0.46229</td>
</tr>
<tr>
<td>Gamma1</td>
<td>0.43489</td>
</tr>
<tr>
<td>Delta</td>
<td>0.54130</td>
</tr>
<tr>
<td>Asymmetry</td>
<td>0.21342</td>
</tr>
<tr>
<td>Tail</td>
<td>0.08950</td>
</tr>
</tbody>
</table>

Rem: Asymptotic 1% critical value for individual statistics = 0.75.
Asymptotic 5% critical value for individual statistics = 0.47.

Adjusted Pearson Chi-square Goodness-of-fit test

<table>
<thead>
<tr>
<th>Lags</th>
<th>Statistic</th>
<th>P-Value(lag-1)</th>
<th>P-Value(lag-k-1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>24.9000</td>
<td>0.981261</td>
<td>0.729677</td>
</tr>
<tr>
<td>50</td>
<td>26.7500</td>
<td>0.995994</td>
<td>0.946240</td>
</tr>
<tr>
<td>60</td>
<td>32.6500</td>
<td>0.997893</td>
<td>0.972622</td>
</tr>
</tbody>
</table>

Rem.: \( k = \# \) estimated parameters

We do not intend to comment this application in details. However, looking at these results, one can briefly argue that the model seems to capture the dynamics of the first and second moments of the CAC40 (see the Box-Pierce statistics). Moreover, the Sign Bias tests show that there is no remaining leverage component in the innovations while the Nyblom stability test suggests that the estimated parameters are quite stable during the investigated period. Finally, our model specification is not rejected by the goodness-of-fit tests for various lag lengths.

To obtain the \( h \)-step-ahead forecasts, access the menu Test/Forecast and set the number of forecasts, pre-sample observations (to be plotted) as well as some other graphical options (Figure
Figure 10 shows 10 pre-sample observations and the forecasts up to horizon 10 of the conditional mean. The forecasted bands are $\pm 2\hat{\sigma}_{t+h|t}$ (note that the critical value 2 can be changed).

Finally, the residuals, the squared residuals, the conditional variance, the (forecasted) probability integral transform and the forecasted conditional mean and conditional variance series can be stored in the database as a new variable. When selecting this option, a first window appears and the user selects the series to be stored (figure 11). A default name is then proposed for this series.

**5.3 Using the “Light Version”**

First, to specify the model you want to estimate, you have to edit `GarchEstim.ox` with any text editor. Yet we recommend OxEdit. It is a shareware that highlights Ox syntax in color (see http://www.oxedit.com for more details). An example of the `GarchEstim.ox` file is displayed here below.
Let us study this file more in details. The `#import` statement indicates that this file is linked with the `Garch.oxo` and `Garch.h` files. In the body of the file (after the `main()` instruction), a
new Garch object is first created and a database is loaded. The user has to enter the correct path of the database, but also has to pay attention to the structure of the database he will use. For instance, to use a Microsoft Excel file, the format of the spreadsheet is of crucial importance. The following convention has to be adopted when loading an Excel spreadsheet: variables are in columns, columns with variables are labelled, there is an unlabelled column containing the dates (with the form Year-Period) and the data form a contiguous sample. Here is an example: 27

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>RET</td>
<td>MON</td>
<td>HOL</td>
</tr>
<tr>
<td>2</td>
<td>1990-1</td>
<td>0.0439</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1990-2</td>
<td>-0.0302</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1990-3</td>
<td>0.0845</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

We note then that the dependent variable (Y), the regressor(s) in the mean equation (X), the regressor(s) in the variance equation (Z) and the "realized" volatility (0) are selected with the Select function. Ox being case-sensitive, the exact name of the variable has to be entered. The second and third arguments denote the starting and ending observations to be considered. By default, "0" and "0" mean that all the observations are selected. From this selection, a sample can be extracted with the SetSelSample function. The arguments are ordered as (StartYear, StartPeriod, EndYear, EndPeriod2) and the default (-1, 1, -1, 1) means all the selected observations.

The GarchEstim.ox file consists in six parts: 28

- the “Data” part deals with the database, the sample and the variables selection;
- the “Specification” part is related to the choice of the model, the lag orders and the shape of the distribution;
- the “Parameters” part consists in two procedures. BOUNDS to constraint or not several parameters to range between a lower and an upper bound (see Section 4.1), FixParam to fix some parameters to their starting values;
- the “Output” part includes several options including MLE that refer to the computation method of the standard deviations of the estimated parameters, TESTONLY, useful when

---

27 See Doornik (2001) for the supported formats, the Load functions and other related information. Interested reader can also take a look at the DJIA.xls file included in the package for an example of Excel file ready to be loaded by Ox. Please be very careful when editing numbers in an Excel file, especially regarding the decimal separator ("," or "." depending on the language used).

28 All the functions cited here are described in details in section 2.3
you want to run some tests on the raw series, prior to any estimation and GRAPHS and FOREGRAPHS, to print graphs for the estimation and the forecasting, respectively;\footnote{Graphics will only be displayed when using GiveWin as front-end.}

- the "Tests & Forecasts" part allows to compute different tests and to parameterize the forecasting part. Note that BOXPIERCE, ARCHLAGS and PEARSON all require a vector of integers corresponding to the lags used in the computation of the statistics;

- DoEstimation launches the estimation of the model and the STORE function allowing to store some series. The argument of the DoEstimation procedure is a vector containing starting values of the parameters in a specified order (but the user can also let the program take defaults values).

Note that the "Light Version" is more than just a replication of the "Full Version" without the graphical interface. Indeed, G@RCH uses the object-oriented programming features of Ox and provides a new class called Garch. All the functions of this class can thus be used within an Ox program. To illustrate the potentiality of our package, we also provide Forecast.ox, an example that computes 448 one-step-ahead forecasts of the conditional mean and conditional variance (using the estimated parameters presented in the previous section), computes the Mincer-Zarnowitz regression and performs some out-of-sample density forecast tests as suggested by Diebold, Gunther, and Tay (1998).

The interesting part of Forecast.ox is printed in the next box. This code has been used to produce Figure 12 and the outputs associated with this forecasting experiment (see below).
Forecast.ox

```ox
#import <packages/garch/garch>
maint() {
    decl garchobj;
garchobj = new Garch();
    ...

garchobj.DoEstimation(<>);
decl number_of_forecasts=448; // number of h_step_ahead forecasts
decl step=1; // specify h (h-step-ahead forecasts)
decl T=garchobj.GetcT();
decl par=garchobj.PAR()[][0];
println("!!! Please Wait while computing the forecasts !!!");
decl forc=<>.
    h, yfor=<>.
    Hfor=<>;
decl RV=columns(garchobj.GetGroup(O_VAR));
decl shape=<>;
if (garchobj.GetDistri()==1 || garchobj.GetDistri()==2) // Except for the HYGARCH
    shape=par[rows(par)-1];
else if (garchobj.GetDistri()==3)
    shape=par[rows(par)-2:rows(par)-1];
for (h=0; h<number_of_forecasts; ++h) {
    garchobj.FORECAST(1,step,0);
    garchobj.SetSelSample(-1, 1, T+h, 1);
    garchobj.InitData();
yfor|=garchobj.GetForcData(Y_VAR, step);
    forc|=garchobj.FORECASTING();
    if (RV==1)
        Hfor|=garchobj.GetForcData(O_VAR, step); // If you use the realized volatility
}
decl cd=garchobj_CD(yfor-forc[0].forc[1].garchobj.GetDistri(),shape);
println("Density Forecast Test on Standardized Forecast Errors");
garchobj.APGT(cd,20|30,rows(par));
garchobj.AUTO(cd, number_of_forecasts, -0.1, 0.1, 0);
garchobj.confidence_limits_uniform(cd,30,0.95,1,4);
if (RV==0) {
    DrawTitle(5, "Conditional variance forecast and absolute returns");
    Hfor = (yfor - meanc(yfor)).^2;
}
else
    DrawTitle(5, "Conditional variance forecast and realized volatility");
    Draw(5, (Hfor*forc[1].1));
    ShowDrawWindow();
garchobj.MZ(Hfor, forc, number_of_forecasts);
garchobj.FEM(forc, yfor*Hfor);
garchobj.STORE(0,0,0,0,0,"01",0); // Arg.1,2,3,4,5 ...
    // Arg.6 : Suffix....
    // Arg.7 : if 0, ....
delete garchobj;
}
```

In the first four panels of Figure 12, we show the correlograms of $\left( \hat{\zeta} - \bar{\zeta} \right)^j$, for $j = 1, 2, 3, 4$. This graphical tool has been proposed by Diebold, Gunther, and Tay (1998) to detect potential remaining dependence in the conditional mean, variance, skewness, kurtosis. In our example, it seems that the probability integral transform is independently distributed.

[INSERT FIGURE 12]

Panel 5 of Figure 12 also shows the histogram (with 30 cells) of $\hat{\zeta}$ with the 95% confidence bands. From this figure, it is clear that the AR(1)-APARCH(1,1) model coupled with a skewed Student distribution for the innovations performs very well with the dataset we have investigated. This conclusion is reinforced by the Pearson Chi-square goodness-of-fit test printed hereafter that provides a statistical version of the graphical test presented in Figure 12. Finally, the program performs the Mincer-Zarnowitz regression given in Eq. (42) that regresses the observed volatility
(in our case the realized volatility) on a constant and a vector of 448 one-step-ahead forecasts of the conditional variance (produced by the APARCH model). 30 The results (reported in the next box) suggest that the APARCH model gives good forecasts of the conditional variance. Indeed, looking at the estimated parameters of this regression, one can hardly conclude that the APARCH model provides biases forecasts. Moreover, the $R^2$ of this regression is higher than 40 % (See Andersen and Bollerslev (1998) for more details).

| Density Forecast Test on Standardized Forecast Errors Adjusted Pearson Chi-square Goodness-of-fit test |
|-------------------------------------------------|-------------------------------|---------------------|
| Lags    | Statistic | P-Value(lag-1) | P-Value(lag-k-1) |
| 20      | 21.0179   | 0.335815       | 0.020969          |
| 30      | 26.5089   | 0.598181       | 0.149654          |

Rem.: $k =$ number of estimated parameters

Mincer-Zarnowitz regression on the forecasted volatility

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Std. Error</th>
<th>t-value</th>
<th>t-prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_0$</td>
<td>-0.225818</td>
<td>0.264837</td>
<td>-0.8527</td>
</tr>
<tr>
<td>$a_1$</td>
<td>1.370648</td>
<td>0.176086</td>
<td>7.784</td>
</tr>
</tbody>
</table>

$R^2$: 0.402914

Note: S.E. are Heteroskedastic Consistent (White, 80)

---

30The realized and one-step-ahead forecasts are plotted in the last panel of Figure 12. 
6 Versions and Future Improvements

6.1 Releases History

Here is the history of the G@RCH package releases:

v.2.30 : April, 22nd
v.2.10 : September, 4th, 2001.
v.2.00 : April, 23th, 2001.
v.1.11 : November, 18th, 2000.
v.1.10 : October, 30th, 2000.
v.1.00 : September, 4th, 2000.

6.2 Future Improvements

• Analytical gradients
• Multivariate GARCH models (under development)
• New "add-ons" with specific applications such as Value-at-Risk...
• ...

We Wish You A Productive Use Of G@RCH 2.3 !

References


Figure 1: Selecting the variables

Figure 2: Model Settings
Figure 3: Selecting the Starting Values Method

Figure 4: Standard Errors Estimation Methods
Figure 5: Entering the Starting Values

Figure 6: Graphics Menu
Figure 7: Graphical Analysis
Figure 8: Tests Dialog Box

Figure 9: Forecasting Menu
Figure 10: Forecasts from an AR(1)-APARCH(1,1).

Figure 11: Storing in the Database
Figure 12: Density Forecast Analysis