GARCH Toolbox Release Notes

The "GARCH Toolbox 2.1 Release Notes" on page 1-1 describe the changes included in Version 2.1 of the GARCH Toolbox since Version 2.0.1. The following topics are discussed in these Release Notes:

- "New Features" on page 1-2
- "Upgrading from an Earlier Release" on page 1-3

If you are upgrading from a release earlier than Release 14SP2, you should also see "GARCH Toolbox 2.0.1 Release Notes" on page 2-1.

Printing the Release Notes

If you would like to print the Release Notes, you can link to a PDF version.

Contents

GARCH Toolbox 2.1 Release Notes

1

New Features 1-2
User-Specified Noise Process 1-2

Upgrading from an Earlier Release 1-3
garchsim Argument Is Renamed 1-3
garchsim Defaults to Current Random Number Generator State 1-3

GARCH Toolbox 2.0.1 Release Notes

2

New Features 2-2
garchfit Returns Exit Flags 2-2
NaN Output for Calculation Failures 2-2

Upgrading from an Earlier Release 2-3

GARCH Toolbox 2.0 Release Notes

3

New Features 3-2
GJR and EGARCH Conditional Variance Models 3-2
Student's t Distribution 3-3
User-Specified Presample Data 3-3
Improved Estimation Engine 3-3
Function Summary 3-3

Upgrading from an Earlier Release **3-5**
Automatic Setting of Series Presample Data **3-5**
Inequality Constraints Now Linked to TolCon **3-5**
Transient Decay Limits for Simulation **3-6**

1

GARCH Toolbox 2.1 Release Notes

New Features 1-2
User-Specified Noise Process 1-2

Upgrading from an Earlier Release 1-3
garchsim Argument Is Renamed 1-3
garchsim Defaults to Current Random Number Generator State 1-3

New Features

This section introduces the new features and enhancements added in the GARCH Toolbox since Version 2.0.1.

User-Specified Noise Process

The `garchsim` function now allows you to provide a time series matrix of standardized, i.i.d. disturbances to drive the output `Innovations` time series process. In previous versions, you could only provide a state that was used to generate a random noise process. See the `State` input argument on the `garchsim` reference page for more information.

Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving to the GARCH Toolbox 2.1 from an earlier version of the GARCH Toolbox.

garchsim Argument Is Renamed

In Version 2.1, the `garchsim` argument `Seed` is renamed to `State` for consistency with the MATLAB `rand` and `randn` functions. The name change, in itself, introduces no backward incompatibilities. The following topic explains a related change.

garchsim Defaults to Current Random Number Generator State

In Version 2.0.1 of the GARCH Toolbox, if you did not specify the `Seed` argument, `garchsim` used the initial random number generator state, 0. The `Seed` argument corresponded to the `rand` and `randn` state value.

In Version 2.1, if you do not specify the `State` (formerly `Seed`) argument, `garchsim` uses the current state of `rand` and `randn`, rather than the initial state. You can use the commands `s = rand('state')` and `s = randn('state')` to determine the current state of these random number generators. See the `rand` and `randn` reference pages for more information.

2

GARCH Toolbox 2.0.1 Release Notes

New Features 2-2
garchfit Returns Exit Flags 2-2
NaN Output for Calculation Failures 2-2

Upgrading from an Earlier Release 2-3

New Features

This section introduces the new features and enhancements added in the GARCH Toolbox since Version 2.0.

garchfit Returns Exit Flags

The `Summary` output of the `garchfit` function now includes a field, `exitFlag`, that describes the exit condition of the optimization. The possible values are

- `>0` Log-likelihood objective function converged to a solution.
- `0` Maximum number of function evaluations or iterations was exceeded.
- `<0` Function did not converge to a solution.

NaN Output for Calculation Failures

In the event an error occurs in the calculation of some `garchfit` outputs, the outputs are set to `NaN`. For example, the error could result from a lack of convergence or the violation of boundary constraints. Specifically,

- For errors in the calculation of the standard errors, all fields of `Errors` associated with estimated coefficients are set to `NaN`.
- For errors in the calculation of the innovations, `Innovations` is a vector of `NaN`s.
- For errors in the calculation of the conditional standard deviations, `Sigmas` is a vector of `NaN`s.

If you detect `NaN`s in any of these outputs, check the warning and convergence messages for a specific cause.

Upgrading from an Earlier Release

There are no upgrade issues if you are upgrading from GARCH Toolbox 2.0 the GARCH Toolbox 2.0.1.

However, if you are upgrading from version earlier than Version 2.0, see “Upgrading from an Earlier Release” on page 3-5 of the GARCH Toolbox 2.0 Release Notes.

3

GARCH Toolbox 2.0 Release Notes

New Features 3-2
GJR and EGARCH Conditional Variance Models 3-2
Student's t Distribution 3-3
User-Specified Presample Data 3-3
Improved Estimation Engine 3-3
Function Summary 3-3

Upgrading from an Earlier Release 3-5
Automatic Setting of Series Presample Data 3-5
Inequality Constraints Now Linked to TolCon 3-5
Transient Decay Limits for Simulation 3-6

New Features

This section introduces the new features and enhancements added in the GARCH Toolbox since Version 1.0.2 (Release 13).

- "GJR and EGARCH Conditional Variance Models" on page 3-2
- "Student's t Distribution" on page 3-3
- "User-Specified Presample Data" on page 3-3
- "Improved Estimation Engine" on page 3-3

A "Function Summary" on page 3-3 lists changes to individual functions.

GJR and EGARCH Conditional Variance Models

GARCH Toolbox Version 2.0 supports two new asymmetric conditional variance models

- **GJR, sometimes known as Threshold-GARCH or TGARCH.** GJR models are similar to GARCH models, but include a term to capture the leverage effect, or negative correlation, between asset returns and volatility. That is, for certain asset classes, most notably equities but excluding foreign exchange, volatility tends to rise in response to lower than expected returns and to fall in response to higher than expected returns. Such an effect suggests models that include an asymmetric response to positive and negative surprises.

  See Glosten, L.R., R. Jagannathan, and D.E. Runkle, "On the Relation Between Expected Value and the Volatility of the Nominal Excess Return on Stocks", *The Journal of Finance*, Vol.48, 1993.

- **EGARCH, or Exponential-GARCH.** EGARCH models also capture the leverage effect, but are treated as an ARMA model for the logarithm of the conditional variance. They are fundamentally different from GARCH and GJR models in that the standardized innovation serves as the forcing variable for both the conditional variance and the error.

  See Nelson, D.B., "Conditional Heteroskedasticity in Asset Returns: A New Approach", *Econometrica*, Vol. 59, 1991.

See "Conditional Mean and Variance Models" in the GARCH Toolbox documentation for more information.

Student's t Distribution

In addition to Gaussian distributions, the GARCH Toolbox now supports Student's t conditional probability distributions. These distributions explicitly capture fat-tails of asset returns.

User-Specified Presample Data

The functions `garchfit`, `garchinfer`, and `garchsim` now allow user-specified presample observations of asset returns, conditional standard deviations, and innovations. This new interface allows Monte Carlo simulation of forecasts and forecast error distributions, and tightly integrates the estimation, simulation, and forecasting engines. See "Presample Observations" in the GARCH Toolbox documentation for more information.

Improved Estimation Engine

The estimation engine, `garchfit`, now provides improved numerical robustness and run-time performance, as well as improved methods for applying initial parameter estimates and for handling strict inequality constraints. It now also accepts user-specified presample observations.

Function Summary

Version 2.0 of the GARCH Toolbox provides new or changed capabilities for the following functions.

Function	Enhancement or Change
`garchdisp`	Output now includes a model summary statement, and lists the selected conditional probability distribution.

Function	Enhancement or Change	

`garchfit`
`garchinfer`
`garchsim`
	Provide three new input arguments, `PreInnovations`, `PreSigmas`, and `PreSeries` that you can use to provide presample data. If you provide no presample data, these functions automatically derive any necessary presample data. If you provide your own presample data, you must provide all that are appropriate for your model.
	
	See "User-Specified Presample Data" on page 3-3 above, and "Presample Observations" in the GARCH Toolbox documentation for more information.	

`garchset`	Provides the following new options:	
	`Distribution`	Conditional distribution of innovations. Specify `'Gaussian'` for Gaussian, and `'T'` for Student's t. The default is `'Gaussian'`.
	`DoF`	Degrees of freedom parameter for t distributions (must be > 2). The default is `[]`.
	`FixLeverage`	Equality constraint indicator for `Leverage` coefficients. The default is `[]`.
	`Leverage`	Leverage coefficients for asymmetric EGARCH(P,Q) and GJR(P,Q) models. The default is `[]`.
	`VarianceModel`	Conditional variance model. Specify `'GARCH'`, `'EGARCH'`, `'GJR'`, or `'Constant'`. The default is `'GARCH'`.
	Note that typing the name of a specification structure at the command line now displays only those fields that are applicable to the specified model. See "Examples of Specification Structures" in the GARCH Toolbox documentation.	

Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving to the GARCH Toolbox 2.0 from an earlier version of the GARCH Toolbox.

- "Automatic Setting of Series Presample Data" on page 3-5
- "Inequality Constraints Now Linked to TolCon" on page 3-5
- "Transient Decay Limits for Simulation" on page 3-6

Automatic Setting of Series Presample Data

When inferring the innovations, ε_t, from the mean equation

$$\varepsilon_t = -C + y_t - \sum_{i=1}^{R} \phi_i y_{t-i} - \sum_{j=1}^{M} \theta_j \varepsilon_{t-j} - \sum_{k=1}^{Nx} \beta_k X(t,k)$$

Version 1.0 of the GARCH Toolbox began at the $(R+1)$ th observation of y_t, and used the actual observations $\{y_1, y_2, \ldots, y_R\}$ as presample data to initiate the inverse filter. Because of this, the first R innovations were not truly inferred, but were simply assigned an expected value of zero. Although this method is correct, it was inconsistent with the method for setting presample observations used to infer the conditional variances, σ_t^2, from the variance equation.

In Version 2.0, if you let `garchfit` or `garchinfer` automatically generate the presample data, these functions assign the sample mean of `Series` to presample observations of y_t, and begin the inference process at the first observation of y_t. As a result, the first R innovations are actually inferred, and are no longer assigned a default value of zero. This method is consistent with the method for setting presample observations used to infer the conditional variances, σ_t^2.

This change should make little difference in the estimation results for time series of even moderate length, and then only for conditional mean models with an AR component, i.e., mean models with $R > 0$.

Inequality Constraints Now Linked to TolCon

In Version 1.0 of the GARCH Toolbox, parameter constraints were largely unrelated to the value of the termination tolerance on constraint violation,

`TolCon`. Additionally, parameters for which theoretical lower and upper bounds did not exist were often unconstrained.

Version 2.0 now explicitly sets appropriate lower and upper bound constraints for all parameters. More importantly, all strict inequality constraints, including lower and upper bounds, are linked to `TolCon`. All Version 1.0 estimation results may not be reproducible using the default value `TolCon = 1e-7`. In some situations, you may need to decrease `TolCon` to reproduce a particular result to within termination tolerances. However, these situations should occur infrequently, and, in most cases, you should see little if any significant differences. The example "Active Lower Bound Constraint" in the GARCH Toolbox documentation illustrates the most common case.

Because of this change, the `garchfit` constraint handling mechanism is now highly cohesive and more tightly integrated with the underlying optimizer `fmincon` in the Optimization Toolbox. The estimation process is now much more robust, and you should see significant runtime performance improvements. See "Constraint Violation Tolerance" in the GARCH Toolbox documentation for more information.

Transient Decay Limits for Simulation

In the absence of explicit presample data, `garchsim` places a 10000 observation maximum on the number of observations allowed for transients to decay to some arbitrarily small value, but which are later discarded.

If you suspect transients still exist, try increasing the response tolerance via the `Tolerance` argument. See "Automatically Generated Presample Data" and "Specifying a Scalar Response Tolerance" in the GARCH Toolbox documentation for more information.