Package 'gEcon'

November 7, 2013

Type Package

Title General Equilibrium Economic Modelling Language and Solution Framework (gEcon)

Version 0.4.0

Date 2013-11-07

Author Grzegorz Klima, Karol Podemski, Kaja Retkiewicz-Wijtiwiak

Copyright Chancellery of the Prime Minister of the Republic of Poland

Maintainer Grzegorz Klima <gklima@users.sourceforge.net>

Description Package for developing and solving dynamic general equilibrium models

Depends R(>= 3.0.0)

Imports methods, MASS, Matrix, nleqslv, Rcpp

License_restricts_use yes

License file LICENCE

R topics documented:

`gecon-package` *General Equilibrium Economic Modelling Language and Solution Framework (gEcon)*

Description

Package for developing and solving dynamic general equilibrium models.

Details

Package:	gecon
Type:	Package
Version:	0.4.0
Date:	2013-11-07
License:	file LICENCE
Depends:	R(>= 3.0.0), Matrix, MASS, nleqslv, Rcpp, methods

`gEcon` is a framework for developing and solving large scale dynamic stochastic general equilibrium models. It consists of model description language and an interface with a set of solvers in R. It was developed at the Department for Strategic Analyses at the Chancellery of the Prime Minister of the Republic of Poland as a part of a project aiming at construction of large scale DSGE models of Polish economy.

Publicly available toolboxes/toolkits used in RBC/DSGE modelling require users to derive first order conditions and linearisation equations by pen & pencil (e.g. Uhlig's toolkit) or at least require manual derivation of FOCs (e.g. Dynare). Owing to the development and implementation of an

algorithm for automatic derivation of first order conditions, gEcon allows users to describe their models in terms of optimisation problems of agents. To authors' best knowledge there is no other publicly available framework for writing and solving DSGE in this natural way. Writing models in terms of optimisation problems instead of FOCs is far more natural to an economist, takes off the burden of tedious differentiation and reduces the risk of making a mistake. gEcon allows users to focus on economic aspects of the model and makes it possible to design large-scale (100+ variables) models. As a bonus gEcon can automatically produce a draft of Latex documentation for a model.

The model description language is simple and intuitive. Given optimisation problems, constraints and identities computer automatically derives FOCs, steady state equations and linearisation matrices. Numerical solvers can then be employed to determine steady state and approximate equilibrium laws of motion around it.

Author(s)

Grzegorz Klima <gklima@users.sourceforge.net>
Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

Maintainer: Grzegorz Klima <gklima@users.sourceforge.net>

References

Cf. gEcon manual distributed with the package.

Examples

```
# copy the example model to current working directory
file.copy(from = file.path(system.file("examples", package=gEcon),
          rbc.gcn), to = getwd())

# make and load model
rbc <- make_model(rbc.gcn)

# solve model
rbc <- steady_state(rbc)
rbc <- solve_pert(rbc)

# compute and print correlations
rbc <- set_shocks(rbc, matrix(0.01, 1, 1))
rbc <- compute_corr(rbc, ref_var = Y)
get_moments(rbc, moments=TRUE, correlations=TRUE,
            autocorrelations=TRUE, var_dec=TRUE)
irf_rbc <- compute_irf(rbc)
plot_simulation(irf_rbc)
summary(rbc)
```

check_bk *Blanchard Kahn conditions and eigenvalues*

Description

Checks Blanchard Kahn conditions and prints information about eigenvalues.

Usage

```
check_bk(model)
```

Arguments

model	an object of class gecon_model.

Details

Eigenvalues are computed when gEcon attempts to solve the perturbation (solver uses Lapack function zgges to compute eigenvalues). solve_pert must be called before the eigenvalues can be retrived.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

References

Blanchard, O., Kahn C. M. (1980), "The Solution of Linear Difference Models under Rational Expectations", *Econometrica*

Examples

```
file.copy(from = file.path(system.file("examples", package=gEcon),
          rbc.gcn), to = getwd())
rbc <- make_model(rbc.gcn)
rbc <- steady_state(rbc)

# solve in loglinearized form
rbc <- solve_pert(rbc)
# check eigenvalues
check_bk(rbc)
```

compute_corr *Computation of correlations*

Description

This function computes the statistics of the model using spectral and simulation methods.

Usage

```
compute_corr(model, ngrid = (64 * 16), filter = TRUE,
             sim = FALSE, nrun = 1e+05, lambda = 1600,
             ref_var = NULL, n_leadlags = 6)
```

Arguments

model	an object of class gecon_model.
ngrid	density of grid used by the Fourier transform. It has to be a multiplicity of 8.
filter	logical. If TRUE, the statistics are computed for the HP-filtered series, otherwise non-filtered series are used for the statistics computation.
sim	logical. If TRUE, simulation methods are used for correlations computations, otherwise spectral methods are used.
nrun	the number of MC simulation runs.
lambda	the lambda parameter for the HP filter.
ref_var	the name or the number of a variable in relation to which correlations are computed. When not specified, the first variable in variables list is treated as the reference value.
n_leadlags	the number of leads/lags for computing correlation tables.

Details

Cf. gEcon manual, chapter "Model analysis".

Value

An object of class gecon_model representing a model. Generic functions such as print and summary allow to show model elements. The function get_moments returns various statistics of the model (both absolute and relative).

Note

The grid has to be large enough (at least 64 * 8) for spectral methods to converge to simulation means.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

References

Hamilton. J.D. (1994), "Time Series Analysis", *Princeton University Press*

Examples

```
# copy model to current working directory
file.copy(from = file.path(system.file("examples", package=gEcon),
          rbc.gcn), to = getwd())

# make and load model
rbc <- make_model(rbc.gcn)

# solve model
rbc <- steady_state(rbc)
rbc <- solve_pert(rbc)

# compute and print correlations
rbc <- set_shocks(rbc, matrix(0.01, 1, 1))
rbc <- compute_corr(rbc, ref_var = Y)
get_moments(rbc, moments=TRUE, correlations=TRUE,
            autocorrelations=TRUE, var_dec=TRUE)
```

`compute_irf` *Compute impulse response functions (IRFs)*

Description

The function `compute_irf` computes the impulse response functions for the specified set of variables and shocks and returns an object of class `gecon_simulation`.

Usage

```
compute_irf(model, eps_list = NULL, var_list = NULL,
            path_length = 40, cholesky = TRUE)
```

Arguments

`model`	an object of class `gecon_model`.
`eps_list`	a list of shocks for which the IRFs are to computed. If this argument is missing, the IRFs are computed for all the shocks in model. By default, the impulse response functions are created for all the shocks in the model.
`var_list`	a list of variables, for which the impact of shocks has to be computed. By default, the impulse response functions are created for the state variables only.
`path_length`	the number of periods for which IRFs are to be computed.
`cholesky`	logical. If this option is set to FALSE, the function computes the IRFs based on uncorrelated shocks, otherwise variance-covariance matrix is orthogonalized using the Cholesky decomposition and the IRFs are computed using this matrix.

Details

Cf. gEcon manual, chapter "Model analysis".

Value

The function returns an object of `gecon_simulation` class. Generic functions such as `print` and `summary` provide information about the impulse response functions. The `plot_simulation` function allows to visualize the IRFs.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

Examples

```
# copy model to current working directory
file.copy(from = file.path(system.file("examples", package=gEcon),
          rbc.gcn), to = getwd())

# make and load model
rbc <- make_model(rbc.gcn)

# solve model
rbc <- steady_state(rbc)
rbc <- solve_pert(rbc)

# compute and print correlations
rbc <- set_shocks(rbc, matrix(0.01, 1, 1))
irf_rbc <- compute_irf(rbc, cholesky = TRUE,
                       var_list = c(K_s, C, Z, I, Y))
summary(irf_rbc)
plot_simulation(irf_rbc)
```

`find_calibr_eq` *Find a calibrated equation*

Description

The function `find_calibr_eq` finds calibrated equations with the specified indices.

Usage

```
find_calibr_eq(model, no_eq = NULL)
```

Arguments

`model`	an object of class `gecon_model`.
`no_eq`	a numeric variable, specifies the indices of requested equations.

Value

A character vector of requested equations.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

Examples

```
# copy model to current working directory
file.copy(from = file.path(system.file("examples", package=gEcon),
          rbc.gcn), to = getwd())

# make and load model
rbc <- make_model(rbc.gcn)
find_calibr_eq(rbc, c(1))
```

---

find_eq *Find an equation*

Description

The function find_eq finds equations with the specified indices.

Usage

```
find_eq(model, no_eq = NULL)
```

Arguments

model	an object of class gecon_model.
no_eq	a numeric variable, specifies the indices of requested equations.

Value

A character vector of requested equations.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

Examples

```
# copy model to current working directory
file.copy(from = file.path(system.file("examples", package=gEcon),
          rbc.gcn), to = getwd())

# make and load model
rbc <- make_model(rbc.gcn)
find_eq(rbc, c(3, 5))
```

---

gecon_model *Constructor of "gecon_model" class*

Description

The function gecon_model is a constructor of gecon_model class objects.

Usage

```
gecon_model(model_info, map_var, map_shocks, map_params, map_params_free,
            map_params_free_val, map_equations, map_calibr_equations,
            var_eq_map, shock_eq_map, ss_function, calibr_function,
            ss_calibr_function_jac, pert)
```

Arguments

model_info	a character vector of length 3, containing information about the model: the input file name, the input file path and the date of creation.
map_var	a character vector with the list of variable names.
map_shocks	a character vector with the list of shocks names.
map_params	a character vector with the list of all the parameter names.
map_params_free	a character vector with the list of all the free parameter names.
map_params_free_val	a numeric vector with the values of the free parameter names.
map_equations	a character vector with model equations.
map_calibr_equations	a character vector with model calibrating equations.
var_eq_map	a sparse matrix (object of class Matrix) with the mapping of variables into equations.
shock_eq_map	a sparse matrix (object of class Matrix) with the mapping of shocks into equations.
ss_function	a function defining steady state (equilibrium for the static models).
calibr_function	a function used for the calibration of parameters.

`ss_calibr_function_jac`
: a function returning a Jacobian of functions defining the steady state (equilibrium for steady state models) and calibration equations.

`pert`
: a function returning a list with the matrices representing canonical form of the model.

Value

An object of `gecon_model` class.

Note

The constructor `gecon_model` is used in R files created by `gEcon`.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

---

gecon_model-class *Class* "gecon_model"

Description

The class for storing models.

Objects from the Class

Objects can be created by calling `new("gecon_model", ...)` or preferably using `gecon_model` constructor.

Slots

`model_info`**:** a character vector of length 3, containing information about the model: the input file name, the input file path and the date of creation.

`map_params`**:** a character vector with the list of all parameter names.

`map_params_free`**:** a character vector with the list of free parameter names.

`map_free_into_params`**:** a numeric vector with the mapping of free parameters into parameters indices.

`map_params_free_mod_flag`**:** a logical vector with flags denoting if particular free parameter has been modified with respect to the .gcn file calibration

`map_params_calibr`**:** a character vector with the names of calibrated parameters.

`map_params_calibr_mod_flag`**:** a logical vector with the flags denoting if any non-default value has been given for calibrated parameter.

`map_calibr_into_params`**:** a numeric vector with the mapping of free parameters into parameters indices.

`map_var`: a character vector with the names of variables.

`map_shocks`: a character vector with the names of shocks.

`map_equations`: a character vector with the names of equations.

`map_calibr_equations`: a character vector with the names of calibrating equations.

`var_eq_map`: a sparse matrix (`Matrix` class object) with the mapping of variables into equations.

`shock_eq_map`: a sparse matrix (`Matrix` class object) with the mapping of shocks into equations.

`init_calib_pars_supplied`: logical. It informs if calibrated parameters values have been supplied.

`init_vals_supplied`: logical. It informs if values of variables have been supplied.

`loglin`: logical. It informs if the model has to be loglinearized. The default value is TRUE.

`loglin_var`: logical. Flags are set to TRUE for loglinearized variables.

`re_solved`: logical. It is set to TRUE if the model has been solved. The default value is FALSE.

`corr_computed`: logical. If TRUE, indicates that the correlations and other statistics of variables have been computed. The default value is FALSE.

`is_stochastic`: logical. If TRUE, the model has stochastic shocks.

`is_dynamic`: logical. If TRUE, the model has any lead or lagged variables.

`is_calibrated`: logical. If TRUE, the model takes into account calibrating equations when solving for the steady state for a dynamic model (equilibrium in case of static model).

`map_params_free_init_val`: a vector of free parameters values, which have been declared in .gcn file.

`map_params_free_val`: a vector of current free parameter values.

`map_params_calibr_val`: a vector of current values of calibrated parameters.

`params`: a vector of the model parameters.

`steady`: a vector of the steady state values of variables for dynamic models or equilibrium for static models.

`ss_function`: a function defining the steady state(equilibrium for static models).

`ss_function_jac`: a function computing the Jacobian of steady state function(equilibrium for static models).

`ss_calibr_function_jac`: a function computing the Jacobian of both steady state (equilibrium) and calibrating functions.

`calibr_function`: calibrating functions

`init_residual_vector`: a numeric vector of residuals of the steady state (equilibrium) function computed for intial values and calibrating parameters.

`residual_vector`: a numeric vector of residuals of the steady state (equilibrium) function computed for the values of variables and calibrating parameters after the nonlinear solver has exited.

`solver_status`: a character describing the steady state (equilibrium) solver status.

`ss_solved`: logical. If TRUE, steady state(equilibrium for static models) has been found.

`pert`: functions defining perturbation of first order (returning a list of matrices).

`eig_vals`: a matrix of system eigenvalues.

`solution`: a list with elements P, Q, R, S storing solution of the model.

`state_var_indices`: a numeric vector containing the indices of state variables.

`solver_exit_info`: a character containing information about perturbation solver exit information.

`solution_resid`: residuals of checking equations, veryfing if the model has been solved.

`shock_mat`: a variance-covariance matrix of model shocks.

`corr_mat`: a matrix of the model variables correlations.

`autocorr_mat`: a matrix of the model variables autocorrelations.

`corr_variable_mat`: a matrix of correlations of variables with the reference variable lead and lagged values.

`var_position`: a numeric value indicating position of reference variable for the computation of statistics.

`var_dec`: a matrix of variance decomposition of shocks.

`sdev`: a vector of standard deviations of variables.

Methods

print `signature(x = "gecon_model")`: prints information about the model solution status.

show `signature(object = "gecon_model")`: prints short information about the model solution status.

summary `signature(object = "gecon_model")`: prints detailed results of the model.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

Examples

```
# copy model to current working directory
file.copy(from = file.path(system.file("examples", package=gEcon),
          rbc.gcn), to = getwd())

# make and load model
rbc <- make_model(rbc.gcn)
print(rbc)
class(rbc)
```

gecon_simulation *Constructor for "gecon_simulation" class object*

Description

This function creates an object of class "gecon_simulation".

Usage

```
gecon_simulation(sim, eps_list, var_list, sim_type, time_n, model_info, model_variable_name)
```

Arguments

sim	the array of simulation results (three dimensional when the impulse response functions have been computed for more than one shocks).
eps_list	a list of shocks for which the simulations have been computed.
var_list	a list of variables used.
sim_type	a type of simulation.
time_n	the number of periods for which the simulation has been performed.
model_info	a character vector of length 3, containing information about the model: the input file name, the input file path and the date of creation.
model_variable_name	a string denoting the name of the model for which the simulation has been performed.

Value

An object of gecon_simulation class.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

gecon_simulation-class *Class "gecon_simulation"*

Description

The class storing simulation results.

Objects from the Class

Objects can be created by calling the form `new("gecon_simulation", ...)` or (preferably) using `gecon_simulation` constructor.

Slots

`sim`: a three-dimensional array with impulse response functions (the dimensions are variables, time, shocks) or two-dimensional array when storing the results of user defined path of shocks or random path of shocks.

`eps_list`: a vector of shocks for which simulations have been computed.

`var_list`: a vector of simulated variables names.

`sim_type`: a type of simulation.

`time_n`: the number of simulation periods.

`model_info`: a character vector of length 3, containing information about the model: the input file name, the input file path and the date of creation.

`model_variable_name`: a character denoting the name of variable storing model for which the simulations have been created.

Methods

print `signature(x = "gecon_simulation")`: prints diagnostic information about the simulation performed.

show `signature(object = "gecon_simulation")`: prints short information about the simulation.

summary `signature(object = "gecon_simulation")`: prints and returns the simulation results in the form of list.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

See Also

`get_simulation_results` to retrive the simulated series from `sim` slot.

Examples

```
# copy model to current working directory
file.copy(from = file.path(system.file("examples", package=gEcon),
          rbc.gcn), to = getwd())

# make and load model
rbc <- make_model(rbc.gcn)

# solve model
rbc <- steady_state(rbc)
```

```
rbc <- solve_pert(rbc)

# compute and print correlations
rbc <- set_shocks(rbc, matrix(0.01, 1, 1))
irf_rbc <- compute_irf(rbc)
summary(irf_rbc)
class(irf_rbc)
```

`get_moments` *The statistics of model*

Description

The function `get_moments` prints and returns the statistics of the model (absolute and relative to the reference variable).

Usage

```
get_moments(model, var_names, relative_to = FALSE, moments = TRUE,
            correlations = TRUE, autocorrelations = TRUE, var_dec = TRUE,
            to_tex = FALSE)
```

Arguments

`model`	an object of class `gecon_model`.
`var_names`	the names of the variables of interest.
`relative_to`	logical. The default value is FALSE. If TRUE, the function returns moments relative to one of the variables in accordance with relevant options chosen in `compute_corr` (then only 'moments' and 'correalations' are active options).
`moments`	logical. If TRUE, the moments of variables: steady state values, standard deviations and variances are returned with the information about which variables have been loglinearized. If 'relative_to' is set to TRUE then moments and steady state values relative to the reference variable are returned.
`correlations`	logical. The default value is TRUE. If TRUE, a correlation matrix is returned. If relative_to is set to TRUE, then corraletions of variables with lagged and leading values of a chosen variable are returned.
`autocorrelations`	logical. The default value is TRUE. If TRUE then the autocorrelations of variables are returned. If `relative_to` is set to TRUE, this option is inactive.
`var_dec`	logical. The default value is TRUE. If TRUE then the variance decomposition (of shocks) is returned. If `relative_to` is set to TRUE, the option is inactive.
`to_tex`	logical. The default value is FALSE. If TRUE, the output is written to a .tex file.

Value

The function returns a list of absolute or relative moments of variables depending on the value of `relative_to` argument.

When `relative_to` is set to FALSE, the list may consist of the following elements:

- `moments` - means, standard deviations and variances of variables,
- `correlation_matrix` - a matrix of correlation of variables,
- `autocorrelations` - a matrix of correlation of variables with their own lagged values (auto-correlations),
- `variance_decomposition` - the variance decomposition, describing the amount of variable variability that can be ascribed to each of shocks.

When `relative_to` is set to TRUE, the list may consist of two elements:

- `relative_moments` - means, standard deviations and variance of variables with respect to reference variable specified in the function `compute_corr`,
- `correlations_variable` - a matrix of correlation of variables with lead and lagged values of a reference variable (usually GDP).

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

See Also

`compute_corr` to see how the statistics are computed.

Examples

```
# copy model to current working directory
file.copy(from = file.path(system.file("examples", package=gEcon),
          rbc.gcn), to = getwd())

# make and load model
rbc <- make_model(rbc.gcn)

# compute steady state and calibrate alpha
rbc <- steady_state(rbc)
rbc <- solve_pert(rbc)
rbc <- set_shocks(rbc, matrix(0.01, 1, 1))
rbc <- compute_corr(rbc,  ref_var=Y)
get_moments(model = rbc, relative_to = FALSE, moments = TRUE, correlations = TRUE,
            autocorrelations = TRUE, var_dec = TRUE)
get_moments(model = rbc, relative_to = TRUE, moments = TRUE, correlations = TRUE)
```

get_parameter_vals *Parameters of the model*

Description

The function get_parameter_vals prints and returns the values of parameters.

Usage

```
get_parameter_vals(model, var_names, to_tex)
```

Arguments

model	an object of gecon_model class.
var_names	a list of requested parameters names.
to_tex	logical. The default value is FALSE. If TRUE, the output is written to a .tex file.

Value

This function returns both free and calibrated parameter values.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

Examples

```
# copy model to current working directory
file.copy(from = file.path(system.file("examples", package=gEcon),
          rbc.gcn), to = getwd())

# make and load model
rbc <- make_model(rbc.gcn)

# solve model
rbc <- steady_state(rbc)
get_parameter_vals(rbc)
```

get_pert_solution *Recursive laws of motion for the model variables*

Description

The function get_pert_solution prints and returns the recursive laws of motion for the model's variables.

Usage

```
# getting recursive laws of motion
get_pert_solution(model, to_tex = FALSE)
```

Arguments

model	an object of gecon_model class.
to_tex	logical. The default value is FALSE. If TRUE, the output is written to a .tex file.

Value

A list with P, Q, R, S elements. P and Q matrices denote the impact of lagged state variables and current values of shocks variables on current values of state variables. R and S matrices denote the impact of lagged state variables and current values of shocks variables on current values of non-state variables.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

See Also

solve_pert for the description of solution procedure and description of output.

Examples

```
# prepare perturbation
file.copy(from = file.path(system.file("examples", package=gEcon),
          rbc.gcn), to = getwd())
rbc <- make_model(rbc.gcn)
rbc <- steady_state(rbc)

# solve in loglinearized form
rbc <- solve_pert(rbc)
get_pert_solution(rbc)
```

get_residuals *Retriving residuals*

Description

The function get_residuals allows to check the residuals of the steady state equations (equations characterizing equilibrium in case of static models) and identify equations with the highest errors. This information may help to assign better initial values to variables when the solver cannot find the steady state (equilibrium).

Usage

```
get_residuals(model, highest = 5)
```

Arguments

model	an object of class gecon.
highest	the number of equations with the highest error to be printed.

Value

This function returns a list with elements initial and final. The initial residuals are residuals computed using the intial values. The final residuals are residuals computed after the solver has exited. The function prints the indices of equations with the highest initial and final errors. The equations can be investigated using function find_eq.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

Examples

```
file.copy(from = file.path(system.file("examples", package=gEcon),
          rbc.gcn), to = getwd())
rbc <- make_model(rbc.gcn)
# we deliberately give an initial value extremely far from the solution
rbc <- initval_var(rbc, list(Y=10000))
rbc <- steady_state(rbc, use_jac=TRUE,
                        options_list=list(method=Newton,
                                          global=gline,
                                          max_iter = 300,
                                          tol = 1e-7))
get_residuals(rbc)
# residuals are much closer to zero then during initial evaluation.
# One can infer that to solve model with this starting value more iterations
# or change of global search strategy are necessary.
# One can also change starting value of variables in equations 4, 11 and 13.
rbc <- initval_var(rbc, list(Y=1000))
```

```
rbc <- steady_state(rbc, use_jac=TRUE,
                        options_list=list(method=Newton,
                                          global=qline,
                                          max_iter = 500,
                                          tol = 1e-7))
get_residuals(rbc)
```

`get_simulation_results` *Retrive series of simulated variables*

Description

The function `get_simulation_results` retrives the series of simulated variables from an object of class `gecon_simulation`.

Usage

```
get_simulation_results(sim_obj)
```

Arguments

`sim_obj`	An object of class `gecon_simulation`.

Value

The results are returned as one element list when the simulation has been invoked by `random_path` or `simulate_model` functions or a list of more elements corresponding to the number of shocks when the simulation has been performed with the function `compute_irf`.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

Examples

```
# copy model to current working directory
file.copy(from = file.path(system.file("examples", package=gEcon),
          rbc.gcn), to = getwd())

# make and load model
rbc <- make_model(rbc.gcn)

# solve model
rbc <- steady_state(rbc)
rbc <- solve_pert(rbc)

# compute and print correlations
```

```
rbc <- set_shocks(rbc, matrix(0.01, 1, 1))
irf_rbc <- compute_irf(rbc, cholesky = TRUE,
                           var_list = c(K_s, C, Z, I, Y))
get_simulation_results(irf_rbc)
```

`get_ss_values` *Return steady state (equilibrium) values*

Description

The function `get_ss_values` returns (and prints) the steady state of the model for dynamic models (equilibrium for static models).

Usage

```
get_ss_values(model, var_names, to_tex = FALSE)
```

Arguments

`model`	an object of class `gecon_model`.
`var_names`	the names or the indices of the variables, whose steady state values (equilibrium values) are to be returned. The default option is a vector containing all the variable names.
`to_tex`	logical. The default value is FALSE. If TRUE, the output is written to a .tex file.

Value

A numeric vector with the steady state (equilibrium for static models).

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

Examples

```
# copy model to current working directory
file.copy(from = file.path(system.file("examples", package=gEcon),
          rbc.gcn), to = getwd())

# make and load model
rbc <- make_model(rbc.gcn)

# solve model
rbc <- steady_state(rbc)
get_ss_values(rbc)
get_ss_values(rbc, var_names=c(K_d, L_d))
```

`initval_calibr_par` *Setting initial values of calibrated parameters*

Description

This function enables setting the initial values of calibrated parameters for the nonlinear solver searching for the steady state of dynamic models (equilibrium for static models) and the values of calibrated parameters. If not set by this function, the default values of parameters are assumed to be 0.5.

Usage

```
initval_calibr_par(model, calibr_par)
```

Arguments

`model`	an object of class `gecon_model`.
`calibr_par`	a list or a vector of parameters (with or without their names).

Details

The values of parameters passed to the `gecon_model` are treated as initial values for the steady state solver when the user specifies calibrating equations in a .gcn file and requests that `steady_state` function shall use it. If the calibration is omitted, the initial values of calibrated parameters are treated as their final values, so one has to specify the right set of calibrated parameters values when one decides to omit calibrating equations.

Value

An object of class `gecon_model` representing the model. Generic functions such as `print` and `summary` allow to show the model's elements. The function `get_parameter_vals` return parameter values.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

Examples

```
file.copy(from = file.path(system.file("examples", package=gEcon),
          rbc.gcn), to = getwd())

# make and load model
rbc <- make_model(rbc.gcn)
rbc <- initval_calibr_par(rbc, calibr_par=list(alpha = 0.4))
rbc <- steady_state(rbc, use_jac=TRUE,
                        options_list=list(method=Broyden, global=gline,
```

```
                                          max_iter = 300, tol = 1e-7))
get_parameter_vals(rbc, c(alpha))
```

`initval_var` *Setting initial values of variables.*

Description

The function `initval_var` sets the initial values of the model's variables to values specified by the user. The initial values close to solution will help the nonlinear equations solver to find the solution.

Usage

```
initval_var(model, init_var)
```

Arguments

`model`	an object of class `gecon_model`.
`init_var`	a list or vector of the initial values of variables.

Value

An object of class `gecon_model` representing the model. Generic functions such as `print` and `summary` allow to show model elements. The function `get_ss_values` returns the steady state (equilibrium) values of the model variables.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

Examples

```
file.copy(from = file.path(system.file("examples", package=gEcon),
          rbc.gcn), to = getwd())
rbc <- make_model(rbc.gcn)
rbc <- initval_var(rbc, list(K_d = 10, C = 2, I = 0.5, Y = 2.5))
rbc <- initval_var(rbc, c(pi=0))
```

load_model *Load model from R file*

Description

The function load_model loads the already generated R file with the model and creates an object of class gecon_model.

Usage

```
load_model(model_file)
```

Arguments

model_file	the name of the .R file containing the model's functions and variables. It can be a name of file or a name of file ending with a .R extension.

Details

The R file with the model specification has to be created first. It can be done using the make_model command and the gcn file model specification or manually.

Value

An object of class gecon_model representing the model. Generic functions such as print and summary allow to show the model's elements.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

See Also

make_model in order to create an R file with the model elements based on the model specification.

Examples

```
# copy model to current working directory
file.copy(from = file.path(system.file("examples", package=gEcon),
          rbc.gcn), to = getwd())

# make and load model
rbc <- make_model(rbc.gcn)
rbc2 <- load_model(rbc.R)
print(rbc2)
```

make_model *Make model from .gcn file*

Description

This function calls the dynamic library, parses the .gcn model file, generates an R file and loads it into a gecon_model class object.

Usage

```
make_model(model_file)
```

Arguments

model_file	the name of the .gcn file containing model formulation. It must be ended with a .gcn extension.

Details

Cf. gEcon manual, chapters "Model description language" and "Derivation of First Order Conditions".

Value

An object of class gecon_model representing the model. Generic functions such as print and summary allow to show the model elements.

Note

When the function is called, an R file with the same name as the .gcn file is created in the the .gcn file directory. Additional files such as a Latex documentation or a log may be created when such an option is set to TRUE in the .gcn file.

Author(s)

Grzegorz Klima <gklima@users.sourceforge.net>
Karol Podemski <karol.podemski@gmail.com>
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

See Also

load_model to load already created R file with model.

Examples

```
# copy model to current working directory
file.copy(from = file.path(system.file("examples", package=gEcon),
          rbc.gcn), to = getwd())

# make and load model
rbc <- make_model(rbc.gcn)
print(rbc)
```

`plot_simulation` *Plot a gecon_simulation object*

Description

The function plots the simulations or saves them as `.eps` files in the model's subdirectory `/plots`.

Usage

```
plot_simulation(sim_obj, to_eps = FALSE)
```

Arguments

`sim_obj`	an object of class `gecon_simulation`.
`to_eps`	if TRUE, plot(s) shall be saved as `.eps` file(s) in the model's subdirectory `/plots`.

Value

If the number of variables of interest is greater then five, more then one plots for each impulse are created (max. 5 variables on each plot). Separate plots are created for all the impulses, if the function `compute_irf` has been used for generating simulations.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

Examples

```
# copy model to current working directory
file.copy(from = file.path(system.file("examples", package=gEcon),
          rbc.gcn), to = getwd())

# make and load model
rbc <- make_model(rbc.gcn)

# solve model
rbc <- steady_state(rbc)
rbc <- solve_pert(rbc)
```

```
# compute and print correlations
rbc <- set_shocks(rbc, matrix(0.01, 1, 1))
irf_rbc <- compute_irf(rbc)
plot_simulation(irf_rbc)
```

---

print-methods *Methods for Function* print *in gEcon package*

Description

Prints information about objects of class gecon_simulation and gecon_model.

Methods

signature(x = "gecon_simulation") Prints the name of the model for which the simulations have been created, information about time span, shock and variables used.

signature(x = "gecon_model") Shows the type of the model, the date of creation, the solving status and more detailed information about the number of variables and parameters then the show generic.

---

random_path *Simulation of the model using a random path of shocks*

Description

This function draws random shocks from distribution with user specified covariance matrix and then simulates the behaviour of the system.

Usage

```
random_path(model, eps_list = NULL, var_list = NULL, path_length = 100)
```

Arguments

model	an object of class gecon_model.
eps_list	a list of shock names that should be taken into account. If not specified the system of all the shocks is simulated.
var_list	a list of variables on which the impact of shocks is to be computed. By default, the impact of random path is evaluated for the state variables only.
path_length	the length of stochastic path, default value = 100.

Details

Cf. gEcon manual, chapter "Model analysis".

Value

An object of class gecon_simulation with simulated paths of variables.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

See Also

simulate_model enables user to specify her own path of shocks and simulate the impact. The function returns an object of gecon_simulation class. Generic functions such as print and summary provide information about the simulations. The plot_simulation function allows to visualize the impact on variables.

Examples

```
# copy model to current working directory
file.copy(from = file.path(system.file("examples", package=gEcon),
          rbc.gcn), to = getwd())

# make and load model
rbc <- make_model(rbc.gcn)

# solve model
rbc <- steady_state(rbc)
rbc <- solve_pert(rbc)

# compute and print correlations
rbc <- set_shocks(rbc, matrix(0.01, 1, 1))
irf_rbc <- random_path(rbc, path_length = 100,
                           var_list = c(K_s, C, Z, I, Y))
plot_simulation(irf_rbc)
```

---

set_free_par *Setting free parameters of model*

Description

The function set_free_par defines parameters of a gecon_model class object.

Usage

```
set_free_par(model, free_par, reset = FALSE)
```

Arguments

model	an object of class gecon_model.
free_par	a list or a vector of parameters (with or without their names).
reset	logical value. If TRUE, allows to reset the free parameters to values specified in the .gcn file.

Value

An object of class gecon_model representing the model. If the option reset has been set to TRUE, the model's parameters will be set back to values from .gcn file. Generic functions such as print and summary allow to show model elements. The function get_parameter_vals returns parameter values currently in use.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

Examples

```
# copy model to current working directory
file.copy(from = file.path(system.file("examples", package=gEcon),
          rbc.gcn), to = getwd())

# make and load model
rbc <- make_model(rbc.gcn)

# set free parameters to different values than in .gcn file
rbc <- set_free_par(rbc, free_par = list(beta = 0.98, delta = 0.01))
rbc <- steady_state(rbc, options=list(method=Broyden, global=gline))
get_ss_values(rbc)

# reset values to .gcn file values
rbc <- set_free_par(rbc, reset = TRUE)
rbc <- steady_state(rbc)
get_ss_values(rbc)
```

---

set_shocks *Setting a variance-covariance matrix of stochastic shocks.*

Description

The function set_shocks defines a variance-covariance matrix for the model shocks.

Usage

```
set_shocks(model, shock_matrix)
```

Arguments

model	an object of class gecon_model.
shock_matrix	a positive definite matrix with the dimensions (n * n), where n is the number of shocks in the model.

Details

The rows and columns of shock matrix must agree with the order of shocks stored in a gecon_class object. This order can be checked using the function shock_info and the generic function print.

Value

An object of class gecon_model representing model. Generic functions such as print and summary allow to show the model elements. The function shock_info returns names of shocks, information about which equations they appear in and a current variance-covariance matrix.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

Examples

```
# copy model to current working directory
file.copy(from = file.path(system.file("examples", package=gEcon),
          rbc.gcn), to = getwd())

# make and load model
rbc <- make_model(rbc.gcn)
shock_info(rbc)
rbc <- set_shocks(rbc, matrix(0.01, 1, 1))
shock_info(rbc)
```

---

shock_info *Print information about shocks*

Description

The function shock_info displays the information about the model's shocks (occurence in equations and variance-covariance matrix)

Usage

```
shock_info(model, shock_names)
```

Arguments

model	an object of class gecon_model.
shock_names	a list of shock names.

Value

The function displays only those model's elements, which have already been set or computed. If a variance-covariance matrix has not been set, it will not appear in the `shock_info` output.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

Examples

```
# copy model to current working directory
file.copy(from = file.path(system.file("examples", package=gEcon),
          rbc.gcn), to = getwd())

# make and load model
rbc <- make_model(rbc.gcn)
rbc <- set_shocks(rbc, matrix(0.01, 1, 1))

# getting information about shocks
shock_info(rbc)
```

---

show-methods *Methods for Function* show *in gEcon package*

Description

Show objects of classes `gecon_simulation` and `gecon_model`

Methods

`signature(object = "gecon_simulation")` Shows the name of model for which the simulations have been created, information about time span, shocks and variables used.

`signature(object = "gecon_model")` Shows the type of the model, the date of creation, the solving status and the information about number of variables and parameters.

---

simulate_model *Simulation of the model*

Description

Simulates model based on user defined realisations of shock values. In particular enables to compute the impact of negative shocks.

Usage

```
simulate_model(model, eps_list = NULL, var_list = NULL, shock_m = NULL,
               periods = NULL, path_length = 40,
               sim_type = NULL, model_name = NULL)
```

Arguments

model	an object of class gecon_model.
eps_list	the shock names for the rows of shock_m specified by the user. The default names are the names of the first shocks from the list of shocks up to the number of shock_m matrix rows.
var_list	a list of variables for which the impact has to be computed. By default, the impact of shocks is evaluated for the state variables only.
shock_m	a matrix or vector of user defined shocks. Values for different shocks should be stored in rows and values for periods in columns.
periods	the number of periods for which, shocks in shock_m have been specified. The default values are from 1 to the number of columns of the shock matrix.
path_length	the number of periods for which the model is simulated. The default number is 40.
sim_type	the type of simulation performed on model.
model_name	the name of gecon_class object based on which simulations are created. The user does not have to specify the name explicitly (by default the variable is deparsed and name is retrived automatically).

Value

An object of class gecon_simulation with simulated paths of variables.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

See Also

random_path to use random path of shocks for the simulation of the model. The function returns an object of gecon_simulation class. Generic functions such as print and summary provide information about the simulations. The plot_simulation function allows to visualize the impact on variables.

Examples

```
# copy model to current working directory
file.copy(from = file.path(system.file("examples", package=gEcon),
          rbc.gcn), to = getwd())

# make and load model
```

```
rbc <- make_model(rbc.gcn)

# solve model
rbc <- steady_state(rbc)
rbc <- solve_pert(rbc)

# compute and print correlations
rbc <- set_shocks(rbc, matrix(0.01, 1, 1))
irf_rbc <- simulate_model(rbc, shock_m = matrix(c(-0.05, -0.05), 1, 2),
                          periods=c(1, 4), var_list = c(K_s, C, Z, I, Y))
plot_simulation(irf_rbc)
```

`solve_pert` *Solve the model in a linearised form (1st order perturbation)*

Description

This function solves the model in a linearised form using Christopher Sims' `gensys` solver.

Usage

```
solve_pert(model, loglin = TRUE, not_loglin_var = NULL,
           norm_tol = 1e-08, solver = "sims_solver")
```

Arguments

`model`	an object of class `gecon_model`.
`loglin`	an option to log-linearise the perturbation. If FALSE then model is linearised only.
`not_loglin_var`	a vector of variables that will not be log-linearised.
`norm_tol`	the tolerance for residuals of model(default 1e-12).
`solver`	the name linear RE solver. The default solver is Christopher Sims' solver. Currently no other solvers are available.

Details

Cf. gEcon manual, chapter "Solving model in linearised form".

Value

an object of class `gecon_model` representing the model. Generic functions such as `print` and `summary` allow to show the model elements. Function `get_pert_solution` returns computed recursive laws of motion for the model's variables. The function `check_bk` displays the eigenvalues of the system and checks the Blanchard-Kahn conditions.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

References

Sims, Ch. A. (2002), "Solving Linear Rational Expectations Models.", *Computational Economics*

Examples

```
# create model object and find steady state
file.copy(from = file.path(system.file("examples", package=gEcon),
          rbc.gcn), to = getwd())
rbc <- make_model(rbc.gcn)
rbc <- steady_state(rbc)

# solve in loglinearized form
rbc <- solve_pert(rbc)
get_pert_solution(rbc)

# solve in linearized form
rbc <- solve_pert(rbc, loglin = FALSE)
get_pert_solution(rbc)

# solve with all variables except L_s loglinearized
rbc <- solve_pert(rbc, not_loglin_var=c(L_s))
get_pert_solution(rbc)
```

---

steady_state *Compute the steady state (equilibrium) of the dynamic (static) model*

Description

This function solves for the steady state of a dynamic model (equilibrium for static model) and calibrates the chosen parameters using a set of solvers from the package `nleqslv`.

Usage

```
steady_state(model, solver = "slv1_nleqslv", use_jac = TRUE,
             calibration = TRUE, options_list = NULL, solver_status = FALSE)
```

Arguments

`model`	an object of class gecon_model.
`solver`	the name of nonlinear equations solver. In the current version only an interface to `slv1_nleqslv` has been implemented.
`use_jac`	an option to use the Jacobian generated by the symbolic library. If FALSE, numerical derivatives are computed.

`calibration` if FALSE, calibrating equations will not be taken into account in the computation of the steady state (equilibrium in case of static model). The inital values of calibrated parameters will be then treated as their values.

`options_list` a list containing one or more of the following fields:

- `method` a character, can be set to "Newton" or "Broyden", the default option is "Newton".
- `global` a character, search strategy can be set to "dbldog", "pwldog", "qline", "gline", "none". The default option is "qline".
- `xscalm` a character, a method of scaling x. It can be set to "fixed", "auto". The default option is "fixed".
- `max_iter` a numeric value denoting max. number of iterations. The default value is 150.
- `tol` a numeric value setting the numeric tolerance for a solution. The default value is 1e-6.

`solver_status` the information about the solver exit code.

Details

Cf. gEcon Manual, chapter "Deterministic steady state".

Value

An object of class `gecon_model` representing the model. Generic functions such as `print` and `summary` allow to show model elements. The functions `get_ss_values` and `get_parameter_vals` return steady state (equilibrium) and parameter values respectively.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

See Also

`nleqslv` for the detailed description of `nleqslv` solver capabilities.

Examples

```
# copy model to current working directory
file.copy(from = file.path(system.file("examples", package=gEcon),
          rbc.gcn), to = getwd())

# make and load model
rbc <- make_model(rbc.gcn)

# compute steady state and calibrate alpha
rbc <- initval_calibr_par(rbc, list(alpha = 0.33))
rbc <- steady_state(rbc, use_jac=TRUE,
                        options_list=list(method=Broyden, global=gline,
                                          max_iter = 300, tol = 1e-7))
```

```
get_ss_values(rbc)

# compute steady state not taking alpha calibration into account
rbc <- initval_calibr_par(rbc, list(alpha = 0.4))
rbc <- steady_state(rbc, calibration=FALSE, use_jac=FALSE,
                        options_list=list(method=Newton, global=gline,
                                          max_iter = 100, tol = 1e-5))
get_ss_values(rbc)
```

---

summary-methods *Summary Method for* gecon_model *objects in gEcon package*

Description

This generic function prints a fairly complete summary for gecon_model and gecon_simulation objects.

Methods

signature(object = "gecon_simulation") Prints a summary of a gecon_simulation class object consisting of a shock matrix and the simulation for each shock.

signature(object = "gecon_model") Prints a summary of a gecon_model class object consisting of all the computed statistics and values.

---

var_info *Information about variables*

Description

The function var_info enables printing various information about model's variables on console. In particular it allows to check in which equations a given set of variables appear. In addition, this function prints the already computed statistics of given variables set.

Usage

```
var_info(model, var_names)
```

Arguments

model	an object of class gecon_model.
var_names	the names of the variables of interest.

Details

The function may be useful in debugging model and quick retrieval of information when the model is large. `Incidence info` element shows in which equations variables appear and their timing.

Other fields:

- steady state (equilibrium) values,
- variables info (which ones are loglinearized and which are state variables),
- state variables impact,
- shocks impact,
- moments,
- correlations

display characteristics of variables which have been computed up to the function call.

Note

The function displays only already set and computed elements of model, eg. if the model has been solved but the statistics have not been computed, field `correlations` will not appear in the `var_info` output.

Author(s)

Karol Podemski <karol.podemski@gmail.com>,
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com>

See Also

`shock_info` for information about the shocks.
`get_ss_values`, `get_pert_solution`, `get_moments` to extract directly the steady state (equilibrium) values, the solution of model, various moments and statistics of the model.

Examples

```
# copy model to current working directory
file.copy(from = file.path(system.file("examples", package=gEcon),
          rbc.gcn), to = getwd())

# make and load model
rbc <- make_model(rbc.gcn)

# solve model
rbc <- steady_state(rbc)
rbc <- solve_pert(rbc)

# get information about variables
var_info(rbc, var_names = c(K_d))
```

Index