Package 'gEcon'

March 11, 2018

Type Package

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

Version 1.1.0

Date 2018-03-11

Author Grzegorz Klima, Karol Podemski, Kaja Retkiewicz-Wijtiwiak

Maintainer Karol Podemski <gecon.maintenance@gmail.com>

Description Package for developing and solving dynamic (stochastic) and static general equilibrium models.

Depends R(>= 3.1.0), methods, MASS, Matrix, nleqslv, Rcpp(>= 0.12.0)

License_restricts_use yes

License file LICENCE

NeedsCompilation yes

R topics documented:

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

Description

Package for developing and solving dynamic (stochastic) and static general equilibrium models.

Details

gEcon is a framework for developing and solving large scale dynamic (stochastic) & static 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 & CGE models of the Polish economy.

Publicly available toolboxes used in RBC/DSGE modelling require users to derive the first order conditions (FOCs) and linearisation equations by pen & paper (e.g. Uhlig's tool-kit) or at least require manual derivation of the FOCs (e.g. Dynare). Derivation of FOCs is also required by GAMS and GEMPACK - probably the two most popular frameworks used in CGE modelling. Owing to the development of an algorithm for automatic derivation of first order conditions and implementation of a comprehensive symbolic computations library, 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 & CGE models in this natural way. Writing models in terms of optimisation problems instead of the 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. To this end, gEcon provides template mechanism (similar to those found in CGE modelling packages), which allows to declare similar agents (differentiated by parameters only) in a single block. Additionally, 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 derives the FOCs, steady-state equations, and linearisation matrices automatically. Numerical solvers can be then employed to determine the steady state and approximate equilibrium laws of motion around it.

Author(s)

Karol Podemski <gecon.maintenance@gmail.com> (since 10.2012),
Kaja Retkiewicz-Wijtiwiak <kaja.retkiewicz@gmail.com> (since 10.2012),
Grzegorz Klima <gklima@users.sourceforge.net> (lead developer 10.2012-03.2018)

Maintainer: Karol Podemski <gecon.maintenance@gmail.com>

References

Cf. gEcon users' guide, which can be found at http://gecon.r-forge.r-project.org/.

Examples

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

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

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

```
rbc <- solve_pert(rbc)

# compute and print correlations
rbc <- set_shock_cov_mat(rbc, cov_matrix = matrix(0.01, 1, 1),
                         shock_order = "epsilon_Z")
rbc <- compute_model_stats(rbc, ref_var = "Y")
get_model_stats(rbc)

# compute and plot the IRFs
irf_rbc <- compute_irf(rbc)
plot_simulation(irf_rbc)
summary(rbc)
```

---

check_bk *Blanchard-Kahn conditions and eigenvalues*

Description

The check_bk function checks the Blanchard-Kahn conditions and prints information about eigenvalues.

Usage

```
check_bk(model)
```

Arguments

model	an object of gecon_model class.

Details

The function checks if the Blanchard-Kahn conditions have been satisfied and prints info about eigenvalues larger than 1 in modulus and the number of forward looking variables. Eigenvalues are computed when gEcon attempts to solve the 1st order perturbation (solver uses the Lapack zgges function to compute eigenvalues). The solve_pert function must be called before eigenvalues can be retrieved.

References

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

Examples

```
# copy the example to the current working directory, process it
# and solve for the 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 the model in log-linearised form
rbc <- solve_pert(rbc)

# check eigenvalues
check_bk(rbc)
```

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

Description

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

Usage

```
compute_irf(model, variables = NULL, shocks = NULL,
            sim_length = 40, cholesky = TRUE)
```

Arguments

`model`	an object of `gecon_model` class.
`variables`	the names or indices of variables whose responses are to be simulated.
`shocks`	the names or indices of shocks for which IRFs are to be computed. If missing, IRFs are computed for all shocks with non-zero variance.
`sim_length`	the length of simulation path, the default value is 40.
`cholesky`	a logical value. If set to FALSE, IRFs are computed with initial values of all shocks equal to 1, otherwise the Cholesky decomposition of shock covariance matrix is used (the default).

Details

Cf. gEcon users' guide, chapter 'Model analysis'.

Value

The function returns an object of `gecon_simulation` class.

See Also

For details, see `gecon_simulation-class`. Generic functions such as `print` and `summary` provide information about the impulse response functions. The `plot_simulation` function allows to plot the IRFs.

Examples

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

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

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

# compute and print the IRFs
rbc <- set_shock_cov_mat(rbc, cov_matrix = matrix(0.01, 1, 1),
                         shock_order = "epsilon_Z")
rbc_irf <- compute_irf(rbc, variables = c("K_s", "C", "Z", "I", "Y"))
summary(rbc_irf)
plot_simulation(rbc_irf)
```

---

compute_model_stats *Computation of model statistics*

Description

This function computes statistics of the model using spectral (FFT) or simulation methods.

Usage

```
compute_model_stats(model, n_leadlags = 5, ref_var = NULL, lambda = 1600,
                    ngrid = 64 * 16,
                    sim = FALSE, sim_length = 1e5)
```

Arguments

model	an object of gecon_model class.
n_leadlags	the number of leads and lags of the model's variables for which correlations are to be computed.
ref_var	the name or the index of the reference variable with respect to which correlations are to be computed.
lambda	HP filter parameter, if it is set to 0 no filtering is performed, 1600 is the default value (quarterly data).
ngrid	the density of grid used by the Fast Fourier transform (used only if the sim option is set to FALSE). It has to be a multiple of 8 and has to be large enough to guarantee unbiased results.
sim	a logical value. If TRUE simulation method is used for computing correlations, if FALSE, the Fast Fourier transform is used.
sim_length	the length of simulation path (used only if the sim option is set to TRUE).

Details

Cf. gEcon users' guide, chapter 'Model analysis'.

Value

An object of gecon_model class representing the model.

Note

The density of grid used by the FFT has to be large enough (at least 64 * 8) for spectral method to give accurate results.

References

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

See Also

Generic functions such as print and summary allow to show the model's components. The get_model_stats function returns various statistics of the model (both absolute and relative).

Examples

```
# copy the example to the 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 the model
rbc <- steady_state(rbc)
rbc <- solve_pert(rbc)

# compute and print correlations
rbc <- set_shock_cov_mat(rbc, cov_matrix = matrix(0.01, 1, 1),
                         shock_order = "epsilon_Z")
rbc <- compute_model_stats(rbc, ref_var = "Y")
get_model_stats(rbc)
```

---

gecon-solution_status *Model solution status*

Description

Functions allowing to check the solution status of gecon_model objects.

Usage

```
ss_solved(model)

re_solved(model)
```

Arguments

model	an object of the gecon_model class.

Value

ss_solved: TRUE, if the steady state (equilibrium) of the model has been found. FALSE otherwise.

re_solved: TRUE, if the perturbation has been solved. FALSE otherwise.

Examples

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

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

# before the model is solved both return FALSE
ss_solved(rbc)
re_solved(rbc)

# find the steady state
rbc <- steady_state(rbc)
# solve the model in log-linearised form
rbc <- solve_pert(rbc)
get_pert_solution(rbc)

# after the model has been solved both return TRUE
ss_solved(rbc)
re_solved(rbc)
```

---

gecon_model *Create objects of* gecon_model *class*

Description

The gecon_model function is a constructor of objects of gecon_model class.

Usage

```
gecon_model(model_info, index_sets,
            variables, variables_tex,
            shocks, shocks_tex,
            parameters, parameters_tex,
            parameters_free, parameters_free_val,
            equations, calibr_equations,
            var_eq_map, shock_eq_map, var_ceq_map, cpar_eq_map,
            cpar_ceq_map, fpar_eq_map, fpar_ceq_map,
            ss_function, calibr_function, ss_calibr_jac_function,
            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.
`index_sets`	a list containing information about index sets. The names of its elements correspond to sets' names. Each element contains a character vector of the names of the relevant set's components.
`variables`	a character vector of variables' names.
`variables_tex`	a character vector of variables' LaTeX names.
`shocks`	a character vector of shocks' names.
`shocks_tex`	a character vector of shocks' LaTeX names.
`parameters`	a character vector of all parameters' names.
`parameters_tex`	a character vector of all parameters' LaTeX names.
`parameters_free`	a character vector of free parameters' names.
`parameters_free_val`	a vector of free parameters' values.
`equations`	a character vector of model equations.
`calibr_equations`	a character vector of calibrating equations.
`var_eq_map`	a sparse matrix (an object of `Matrix` class) representing the mapping of variables to equations.
`shock_eq_map`	a sparse matrix (an object of `Matrix` class) representing the mapping of shocks to equations.
`var_ceq_map`	a sparse matrix (an object of `Matrix` class) representing the mapping of variables to calibrating equations.
`cpar_eq_map`	a sparse matrix (an object of `Matrix` class) representing the mapping of calibrated parameters to equations.
`cpar_ceq_map`	a sparse matrix (an object of `Matrix` class) representing the mapping of calibrated parameters to calibrating equations.
`fpar_eq_map`	a sparse matrix (an object of `Matrix` class) representing the mapping of free parameters to equations.

`fpar_ceq_map` a sparse matrix (an object of `Matrix` class) representing the mapping of free parameters to calibrating equations.

`ss_function` a function returning the steady-state/equilibrium equations' residuals.

`calibr_function`
a function used for the calibration of parameters.

`ss_calibr_jac_function`
a function computing the Jacobian of both steady-state (equilibrium) and calibrating functions or NULL.

`pert` a function returning matrices representing the first order perturbation of the model (in the canonical form).

Value

An object of `gecon_model` class.

Note

The `gecon_model` constructor is used in .R files created by `gEcon`. In general, users should not call this function explicitly.

`gecon_model-class` `gecon_model` *class*

Description

The class for storing models.

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.

`index_sets`**:** a list containing information about index sets. The names of its elements correspond to sets' names. Each element contains a character vector of the names of the relevant set's components.

`parameters`**:** a character vector of all parameters' names.

`parameters_tex`**:** a character vector of all parameters' LaTeX names.

`parameters_free`**:** a character vector of free parameters' names.

`map_free_into_params`**:** an integer vector of free parameters' indices.

`parameters_calibr`**:** a character vector of calibrated parameters' names.

`map_calibr_into_params`**:** an integer vector of calibrated parameters' indices.

`variables`**:** a character vector of variables' names.

`variables_tex`**:** a character vector of variables' LaTeX names.

`shocks`**:** a character vector of shocks' names.

`shocks_tex`: a character vector of shocks' LaTeX names.

`equations`: a character vector of model equations (equilibrium relationships).

`calibr_equations`: a character vector of calibrating equations.

`var_eq_map`: a sparse matrix (an object of `Matrix` class) representing the mapping of variables to equations.

`shock_eq_map`: a sparse matrix (an object of `Matrix` class) representing the mapping of shocks to equations.

`var_ceq_map`: a sparse matrix (an object of `Matrix` class) representing the mapping of variables to calibrating equations.

`cpar_eq_map`: a sparse matrix (an object of `Matrix` class) representing the mapping of calibrated parameters to equations.

`cpar_ceq_map`: a sparse matrix (an object of `Matrix` class) representing the mapping of calibrated parameters to calibrating equations.

`fpar_eq_map`: a sparse matrix (an object of `Matrix` class) representing the mapping of free parameters to equations.

`fpar_ceq_map`: a sparse matrix (an object of `Matrix` class) representing the mapping of free parameters to calibrating equations.

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

`is_dynamic`: logical. If TRUE, the model has at least one lead or lagged variable.

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

`ss_function`: a function returning the steady-state/equilibrium equations' residuals.

`ss_calibr_jac_function`: a function computing the Jacobian of both steady-state (equilibrium) and calibrating functions or NULL.

`calibr_function`: a function returning calibrating equations' residuals.

`parameters_free_init_val`: a vector of free parameters' values declared in the .gcn file.

`parameters_free_val`: a vector of current free parameters' values.

`parameters_free_mod_flag`: a logical vector indicating which free parameters' values have been modified relative to the .gcn file setting.

`parameters_calibr_val`: a vector of current calibrated parameters' values.

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

`residual_vector`: a numeric vector of residuals of the steady-state (equilibrium) function computed for variables' values and calibrated parameters after the non-linear solver has exited.

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

`parameters_val`: a vector of model parameters' values.

`variables_ss_val`: a vector of variables' steady-state/equilibrium values.

`ss_solved`: logical. If TRUE, the steady state (equilibrium in case of static models) has been found.

`pert`: a function of the first order perturbation (returning a list of matrices).

`loglin_var`: logical. Flags are set to TRUE for log-linearised variables.

`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 string, solver exit information.

`solution_resid`: a list of residuals of perturbation equations, verifying if the model has been solved.

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

`active_shocks`: a logical vector of the length equal to the number of shocks. If an entry is set to FALSE, the variance of the corresponding shock is zero (the shock is not taken into account during stochastic simulations of the model).

`shock_cov_mat`: a covariance matrix of model shocks.

`shock_cov_mat_flag`: logical. Set to TRUE when the user specifies non-default entries in a covariance matrix of model shocks.

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

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

`ref_var_corr_mat`: a matrix of correlations of model variables with the reference variable's lead and lagged values.

`ref_var_idx`: an integer value, the index of the reference variable used in statistics computation.

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

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

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

Methods

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

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

summary `signature(object = "gecon_model")`: prints detailed information about the model's computation results.

See Also

`gecon_model` is a constructor of objects of `gecon_model` class.

Examples

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

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

gecon_par_info-class gecon_par_info *class*

Description

The class storing information about parameters selected by the user.

Slots

`r_object_name`: a character string. The name of an R object of `gecon_model` class storing the model from which the information about parameters comes from.

`parameters`: a character vector of parameters' names.

`gcn_values`: a numeric vector of free parameters' values specified in the .gcn file.

`current_values`: a numeric vector of parameters' current values.

`calibr_flag`: a logical vector of the length equal to the number of the parameters. If TRUE, a corresponding parameter is a calibrated parameter.

`incid_mat`: a object of `Matrix` class representing the mapping of parameters to equations and calibrating equations.

Methods

print `signature(x = "gecon_par_info")`: Prints all the available information (values, types, incidence matrix) about the parameters which were selected when creating a `gecon_par_info-class` object.

show `signature(object = "gecon_par_info")`: Prints all the available information (values, types, incidence matrix) about the parameters which were selected when creating a `gecon_par_info-class` object.

summary `signature(object = "gecon_par_info")`: Prints all the available information (values, types, incidence matrix) about the parameters which were selected when creating a `gecon_par_info-class` object.

See Also

`par_info` creates an object of `gecon_par_info` class. `gecon_shock_info-class` and `gecon_var_info-class` are similar classes storing the information about shocks and variables, respectively.

Examples

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

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

# compute the steady state and calibrate alpha
```

```
rbc <- steady_state(rbc)
rbc_par_info <- par_info(rbc, all = TRUE)
print(rbc_par_info)
summary(rbc_par_info)
show(rbc_par_info)
```

---

gecon_shock_info-class

gecon_shock_info *class*

Description

The class storing information about shocks selected by the user.

Slots

`r_object_name`**:** a character string. The name of an R object of `gecon_model` class storing the model for which simulations have been performed.

`shocks`**:** a character vector of shocks' names.

`cov_matrix`**:** a numeric matrix containing columns of shock covariance matrix corresponding to selected shocks.

`cov_matrix_flag`**:** a logical value. Set to TRUE when the user enters non-default data into a covariance matrix of shocks.

`incid_mat`**:** an object of `Matrix` class representing the mapping of shocks to equations.

Methods

print `signature(x = "gecon_shock_info")`: Prints all the available information (the incidence matrix, the covariance matrix) about the shocks which were selected when creating a `gecon_shock_info-class` object.

show `signature(object = "gecon_shock_info")`: Prints all the available information (the incidence matrix, the covariance matrix) about the shocks which were selected when creating a `gecon_shock_info-class` object.

summary `signature(object = "gecon_shock_info")`: Prints all the available information (the incidence matrix, the covariance matrix) about the shocks which were selected when creating a `gecon_shock_info-class` object.

See Also

`shock_info` creates an object of `gecon_shock_info` class. `gecon_var_info-class` and `gecon_par_info-class` are similar classes storing the information about variables and parameters, respectively.

Examples

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

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

# solve the model
rbc <- steady_state(rbc)
rbc <- solve_pert(rbc)
rbc <- set_shock_cov_mat(rbc, cov_matrix = matrix(0.01, 1, 1),
                         shock_order = "epsilon_Z")
rbc <- compute_model_stats(rbc,  ref_var = "Y")

# create an object of gecon_shock_info class and display info about the shocks
rbc_shock_info <- shock_info(rbc, all = TRUE)
print(rbc_shock_info)
summary(rbc_shock_info)
show(rbc_shock_info)
```

`gecon_simulation` *Create objects of* `gecon_simulation` *class*

Description

The function creates an object of `gecon_simulation` class.

Usage

```
gecon_simulation(sim,
                 shocks, shocks_tex, variables, variables_tex,
                 sim_name, model_info, r_object_name)
```

Arguments

`sim`	an array with simulation results (3-dimensional if the IRFs have been performed, 2-dimensional otherwise).
`shocks`	a character vector with the names of the shocks for which the simulations have been performed.
`shocks_tex`	a character vector with LaTeX names of the shocks for which the simulations have been performed.
`variables`	a character vector with the names of the variables for which the simulations have been performed.
`variables_tex`	a character vector with LaTeX names of the variables for which the simulations have been performed.
`sim_name`	a character string, the simulation name.

`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.

`r_object_name` a character string with the name of an R object storing the model for which the simulations have been performed.

Value

An object of `gecon_simulation` class.

Note

The `gecon_simulation` constructor is invoked by the `random_path`, `simulate_model`, and `compute_irf` functions. In general, users should not call this function explicitly.

See Also

Generic functions such as `print` and `summary` provide information about the simulations. The `plot_simulation` function allows to visualize the impact of simulations on the model's variables.

---

gecon_simulation-class

`gecon_simulation` *class*

Description

The class storing simulation results.

Slots

`sim`**:** a three-dimensional array with impulse response functions (the dimensions: variables, time, shocks) or two-dimensional array with simulation results.

`shocks`**:** a character vector with the names of the shocks for which the simulations have been performed.

`shocks_tex`**:** a character vector with LaTeX names of the shocks for which the simulations have been performed.

`variables`**:** a character vector with the names of the variables for which the simulations have been performed.

`variables_tex`**:** a character vector with LaTeX names of the variables for which the simulations have been performed.

`sim_name`**:** a character string, the simulation name.

`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.

`r_object_name`**:** a character string, the name of an R object storing the model for which the simulations have been performed.

Methods

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

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

summary `signature(object = "gecon_simulation")`: prints the simulation results.

See Also

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

Examples

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

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

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

# compute and print the IRFs
rbc <- set_shock_cov_mat(rbc, cov_matrix = matrix(0.01, 1, 1),
                         shock_order = "epsilon_Z")
irf_rbc <- compute_irf(rbc)
summary(irf_rbc)
class(irf_rbc)
```

---

gecon_var_info-class gecon_var_info *class*

Description

The class storing information about variables selected by the user.

Slots

`r_object_name`**:** a character string. The name of an R object of `gecon_model` class storing the model from which the information about variables comes from.

`variables`**:** a character vector of selected variables' names.

`is_stochastic`**:** logical. Set to TRUE, if the model, based on which the info was generated, has stochastic shocks.

`is_dynamic`**:** logical. Set to TRUE, if the model, based on which the info was generated, has any lead or lagged variables.

`ss_solved`: logical. Set to TRUE, if the steady state/equilibrium of the model has been found.

`re_solved`: logical. Set to TRUE, if the model, based on which the info was generated, has been solved.

`corr_computed`: logical. Set to TRUE, if correlations and other statistics of variables have been computed.

`ss_val`: a vector of the steady-state (dynamic models) or equilibrium (static models) values of selected variables. If the steady state has not been computed, the vector contains initial values of variables.

`state`: a logical vector of length equal to the number of selected variables. A TRUE entry denotes that a corresponding variable is a state variable.

`state_var_impact`: the rows of the matrices P and R of state space representation corresponding to selected variables.

`shock_impact`: the rows of the matrices Q and S of state space representation corresponding to selected variables.

`std_dev_val`: a numeric vector of standard deviations of selected variables.

`loglin_flag`: a logical vector of length equal to the number of selected variables. A TRUE entry denotes that a corresponding variable was log-linearised before solving the model.

`cr`: a matrix containing the correlations of selected variables with all model variables.

`incid_mat`: an object of `Matrix` class representing the mapping of variables to model equations and calibrating equations.

Methods

print `signature(x = "gecon_var_info")`: Prints all the available information (values, statistics, incidence matrix, etc.) about the variables, which were selected when creating a `gecon_var_info-class` object.

show `signature(object = "gecon_var_info")`: Prints all the available information (values, statistics, incidence matrix, etc.) about the variables, which were selected when creating a `gecon_var_info-class` object.

summary `signature(object = "gecon_var_info")`: Prints all the available information (values, statistics, incidence matrix, etc.) about the variables, which were selected when creating a `gecon_var_info-class` object.

See Also

`var_info` creates an object of `gecon_var_info` class. `gecon_shock_info-class` and `gecon_par_info-class` are similar classes storing the information about shocks and parameters, respectively.

Examples

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

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

```
# compute the steady state
rbc <- steady_state(rbc)
rbc <- solve_pert(rbc)
rbc_var_info <- var_info(rbc, all = TRUE)
print(rbc_var_info)
summary(rbc_var_info)
show(rbc_var_info)
```

`get_index_sets` *List of index sets*

Description

The `get_index_sets` function retrieves a list of all the index sets specified in the `.gcn` file.

Usage

```
get_index_sets(model)
```

Arguments

`model`	an object of `gecon_model` class.

Details

Cf. gEcon users' guide, chapter 'Templates'.

Value

The function returns a list of index sets. Each element of the list corresponds to one set and contains the set components' names as a character vector.

Examples

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

# make and load the model
pure_exchange_t <- make_model("pure_exchange.gcn")

# retrieve the index sets
pure_exchange_ind_sets <- get_index_sets(pure_exchange_t)
```

get_model_info *Accessing information about the name and the creation date of the model*

Description

The `get_model_info` function returns a character vector with information about the model.

Usage

```
get_model_info(model)
```

Arguments

`model`	an object of `gecon_model` class.

Value

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

See Also

For details, see `gecon_model-class`.

Examples

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

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

# retrieve and show model information
model_info <- get_model_info(rbc)
model_info
```

get_model_stats *Statistics of the model*

Description

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

Usage

```
get_model_stats(model, variables = NULL,
                basic_stats = TRUE,
                corr = TRUE,
                autocorr = TRUE,
                var_dec = TRUE,
                to_tex = FALSE, silent = FALSE)
```

Arguments

model	an object of gecon_model class.
variables	the names or the indices of the variables of interest. All variables are selected by default.
basic_stats	a logical value. If TRUE, the following information is returned for selected variables: steady-state value, standard deviation, variance, and information whether a variable has been log-linearised. The default value is TRUE.
corr	a logical value. If TRUE, a correlation matrix is returned. If a reference variable was not NULL while invoking the 'compute_model_stats' function, then correlations of selected variables with leads and lags of the reference variable are also returned. The default value is TRUE.
autocorr	a logical value. If TRUE, autocorrelations of selected variables are returned. The default value is TRUE.
var_dec	a logical value. If TRUE, variance decomposition (contributions of shocks to the variables' variances) is returned. The default value is TRUE.
to_tex	a logical value. If TRUE, the output is written to a .tex file. The default value is FALSE.
silent	a logical value. If TRUE, console output is suppressed. The default value is FALSE.

Value

The function returns a list of model statistics, which may contain the following fields:

- basic_stats - a data frame with steady-state values, standard deviations, variances, and information whether variables have been log-linearised,
- corr - a correlation matrix,
- corr_refvar - a matrix of correlations of selected variables with the reference variable (its lags and leads),
- autocorr - a matrix of autocorrelations,
- var_dec - a matrix of variance decomposition, i.e. contributions of shocks to the variables' variances.

See Also

The compute_model_stats function computes statistics of the model using spectral (FFT) or simulation methods.

Examples

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

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

# solve the model
rbc <- steady_state(rbc)
rbc <- solve_pert(rbc)
rbc <- set_shock_cov_mat(rbc, cov_matrix = matrix(0.01, 1, 1),
                         shock_order = "epsilon_Z")

# compute and retrieve model statistics
rbc <- compute_model_stats(rbc,  ref_var = "Y")
get_model_stats(model = rbc)
```

---

get_par_names *Accessing parameter names used by objects of* gecon_model *class*

Description

The get_par_names function allows to retrieve the names of parameters from objects of gecon_model class.

Usage

```
get_par_names(model, free_par = TRUE, calibr_par = TRUE)
```

Arguments

model	an object of gecon_model class.
free_par	logical. If TRUE (default), free parameters' names are added to the vector of parameters' names.
calibr_par	logical. If TRUE (default), calibrated parameters' names are added to the vector of parameters' names.

Value

The function returns a character vector of parameters' names stored in a given object of gecon_model class.

See Also

For details, see gecon_model-class.

Examples

```
# copy the example to the 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")

# get names of all parameters
get_par_names(rbc)

# get names of free parameters
get_par_names(rbc, calibr_par = FALSE)

# get names of calibrated parameters
get_par_names(rbc, free_par = FALSE)
```

`get_par_names_by_index` *Parameters corresponding to given indices*

Description

The `get_par_names_by_index` function allows to retrieve the names of parameters with given indices.

Usage

```
get_par_names_by_index(model, index_names)
```

Arguments

`model`	an object of `gecon_model` class.
`index_names`	a character vector of indices.

Details

Cf. gEcon users' guide, chapter 'Templates'.

Value

The function returns a character vector of relevant parameters' names.

Examples

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

# make and load the model
pure_exchange_t <- make_model("pure_exchange.gcn")

# model calibration
pure_exchange_t <- set_free_par(pure_exchange_t,
                                free_par= c("alpha__A__1" = 0.3, "alpha__A__2" = 0.4,
                                            "alpha__A__3" = 0.3, "alpha__B__1" = 0.3,
                                            "alpha__B__2" = 0.4, "alpha__B__3" = 0.3,
                                            "e_calibr__A__1" = 3, "e_calibr__B__1" = 1,
                                            "e_calibr__A__2" = 2, "e_calibr__B__2" = 1,
                                            "e_calibr__A__3" = 1, "e_calibr__B__3" = 3))

# get all parameters associated with agent A
par_names_A <- get_par_names_by_index(pure_exchange_t, index_names = "A")
par_info(pure_exchange_t, par_names_A)

# get all parameters associated with agent B
par_names_B <- get_par_names_by_index(pure_exchange_t, index_names = "B")
par_info(pure_exchange_t, par_names_B)
```

---

get_par_values *Parameters' values*

Description

The get_par_values function prints and returns parameters' values.

Usage

```
get_par_values(model, parameters = NULL, to_tex = FALSE, silent = FALSE)
```

Arguments

model	an object of the gecon_model class.
parameters	the names or the indices of the parameters whose values are to be returned. All parameters are listed by default.
to_tex	logical. If TRUE, the output is written to a .tex file. The default value is FALSE.
silent	logical. If TRUE, console output is suppressed. The default value is FALSE.

Value

This function returns both free and calibrated parameters' values.

See Also

For details, see gecon_model-class.

Examples

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

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

# find the steady state
rbc <- steady_state(rbc)

# get parameters' values
get_par_values(rbc)
```

`get_pert_solution` *Recursive laws of motion of the model's variables*

Description

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

Usage

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

Arguments

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

Value

A list of `P`, `Q`, `R`, `S` matrices (`P` and `R` only in case of a deterministic model), a vector of variables' steady-state values (`ss_val`), a vector of flags indicating which variables were log-linearised (`loglin`), and a vector of indices of state variables (`state_ind`). `P` and `Q` matrices describe the impact of lagged state variables and current values of shocks on current values of state variables. `R` and `S` matrices describe the impact of lagged state variables and current values of shocks on current values of non-state (jump) variables.

See Also

solve_pert for the details of finding-a-solution procedure and the description of output which is returned.

Examples

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

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

# find the steady state
rbc <- steady_state(rbc)

# solve the model in log-linearised form and get the results
rbc <- solve_pert(rbc)
get_pert_solution(rbc)
```

---

get_residuals *Retrieving equations' residuals*

Description

The get_residuals function allows to check the residuals of the steady-state (equilibrium) equations of a dynamic (static) model and identify equations with the largest errors. This may help to assign initial values to the model's variables more accurately when the solver cannot find the steady state (equilibrium).

Usage

```
get_residuals(model, largest = 5, calibration = TRUE)
```

Arguments

model	an object of gecon_model class.
largest	the number of equations with the largest errors which are to be printed.
calibration	if FALSE, calibrating equations will not be taken into account when computing equations' residuals. Initial values of calibrated parameters will be then treated as their values.

Value

The function returns a list of two elements: initial and final. Initial residuals are steady-state (equilibrium) equations' residuals computed using the initial values of variables. Final residuals are residuals computed after the solver has exited. The function prints the indices of equations with the largest initial and final errors. The equations can be further investigated using the list_eq function.

Examples

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

# make and load the model
home_prod_templ <- make_model("home_production_templ.gcn")

# for the purpose of the example, initial values are set very far from the solution
home_prod_templ <- initval_var(home_prod_templ, c(N = 0.02,
                                                  N__H = 0.01,
                                                  N__M = 0.01))
home_prod_templ <- steady_state(home_prod_templ)
get_residuals(home_prod_templ)

# after setting more reasonable values the steady state is found
home_prod_templ <- initval_var(home_prod_templ, c(N = 0.5,
                                                  N__H = 0.25,
                                                  N__M = 0.25))
home_prod_templ <- steady_state(home_prod_templ)
get_residuals(home_prod_templ)
```

`get_shock_cov_mat` *Accessing a covariance matrix of model shocks.*

Description

The `get_shock_cov_mat` function returns a covariance matrix of model shocks.

Usage

```
get_shock_cov_mat(model)
```

Arguments

`model`	an object of `gecon_model` class.

Value

The function returns a covariance matrix of model shocks.

See Also

For details, see `gecon_model-class`. The `set_shock_cov_mat` function allows to set/modify shock covariance matrix. The `set_shock_distr_par` function sets distribution parameters (standard deviations, correlations, etc.) of shocks.

Examples

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

# make and load the model
home_production <- make_model("home_production.gcn")
shock_info(home_production, all = TRUE)

# set the shock distribution parameters
home_production <- set_shock_distr_par(home_production,
                                       distr_par = list("sd(epsilon_h)" = 0.7,
                                                        "var(epsilon_m)" = 0.49,
                                                        "cor(epsilon_m,
                                                             epsilon_h)" = 2/3))

# retrieve and show the covariance matrix of model shocks
cov_mat <- get_shock_cov_mat(home_production)
cov_mat
```

---

get_shock_names *Accessing shock names used by objects of* gecon_model *class*

Description

The get_shock_names function allows to retrieve the names of shocks from objects of gecon_model class.

Usage

```
get_shock_names(model)
```

Arguments

model	an object of gecon_model class.

Value

The function returns a character vector of shock names stored in a given object of gecon_model class.

See Also

For details, see gecon_model-class.

Examples

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

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

# get shock names
get_shock_names(rbc)
```

`get_shock_names_by_index` *Shocks corresponding to given indices*

Description

The `get_shock_names_by_index` function allows to retrieve the names of shocks with given indices.

Usage

```
get_shock_names_by_index(model, index_names)
```

Arguments

`model`	an object of `gecon_model` class.
`index_names`	a character vector of indices.

Details

Cf. gEcon users' guide, chapter 'Templates'.

Value

The function returns a character vector of relevant shocks' names.

Examples

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

# make and load the model
home_prod_templ <- make_model("home_production_templ.gcn")

# get shocks affecting home production technology
hp_shocks <- get_shock_names_by_index(home_prod_templ, "H")
```

```
# print information about selected shocks
shock_info(home_prod_templ, hp_shocks)
```

`get_simulation_results` *Retrieve time series of simulated variables*

Description

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

Usage

```
get_simulation_results(sim_obj)
```

Arguments

`sim_obj`	An object of `gecon_simulation-class` class.

Value

The results are returned as a matrix if the simulation has been generated by a call to the `random_path` or `simulate_model` function or a list of matrices corresponding to selected shocks in a call to the `compute_irf` function.

See Also

For details, see `gecon_simulation-class`.

Examples

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

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

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

# compute and print the IRFs
rbc <- set_shock_cov_mat(rbc, cov_matrix = matrix(0.01, 1, 1),
                         shock_order = "epsilon_Z")
irf_rbc <- compute_irf(rbc, cholesky = TRUE,
                       variables = c("K_s", "C", "Z", "I", "Y"))
get_simulation_results(irf_rbc)
```

get_ss_values *Variables' steady-state (equilibrium) values*

Description

The get_ss_values function returns (and prints) the steady state/equilibrium of the model.

Usage

```
get_ss_values(model, variables = NULL, to_tex = FALSE, silent = FALSE)
```

Arguments

model	an object of gecon_model class.
variables	the names or the indices of the variables whose steady-state (equilibrium) values are to be returned. All variables are listed by default.
to_tex	logical. If TRUE, the output is written to a .tex file. The default value is FALSE.
silent	logical. If TRUE, console output is suppressed. The default value is FALSE.

Value

A numeric vector of the steady-state (equilibrium) values of variables.

Examples

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

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

# find and print the steady-state values
rbc <- steady_state(rbc)
get_ss_values(rbc)
get_ss_values(rbc, variables = c("K_s", "L_s"))
```

get_var_names *Accessing variables' names used by objects of* gecon_model *class*

Description

The get_var_names function allows to retrieve the names of variables from objects of gecon_model class.

Usage

```
get_var_names(model)
```

Arguments

model	an object of gecon_model class.

Value

The function returns a character vector of variables' names stored in a given object of gecon_model class.

See Also

For details, see gecon_model-class.

Examples

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

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

# get variables' names
get_var_names(rbc)
```

---

get_var_names_by_index

Variables corresponding to given indices

Description

The get_var_names_by_index function allows to retrieve the names of variables with given indices.

Usage

```
get_var_names_by_index(model, index_names)
```

Arguments

model	an object of gecon_model class.
index_names	a character vector of indices.

Details

Cf. gEcon users' guide, chapter 'Templates'.

Value

The function returns a character vector of relevant variables' names.

Examples

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

# make and load the model
pure_exchange_t <- make_model("pure_exchange.gcn")

# model calibration
pure_exchange_t <- set_free_par(pure_exchange_t,
                                free_par = c("alpha__A__1" = 0.3, "alpha__A__2" = 0.4,
                                             "alpha__A__3" = 0.3, "alpha__B__1" = 0.3,
                                             "alpha__B__2" = 0.4, "alpha__B__3" = 0.3,
                                             "e_calibr__A__1" = 3, "e_calibr__B__1" = 1,
                                             "e_calibr__A__2" = 2, "e_calibr__B__2" = 1,
                                            "e_calibr__A__3" = 1, "e_calibr__B__3" = 3))

# find the equilibrium
pure_exchange_t <- steady_state(pure_exchange_t)

# get all variables' names associated with agent A
var_names_A <- get_var_names_by_index(pure_exchange_t, index_names = "A")


# get all variables' names associated with agent B
var_names_B <- get_var_names_by_index(pure_exchange_t, index_names = "B")

# compare equilibrium allocations
get_ss_values(pure_exchange_t, var_names_A)
get_ss_values(pure_exchange_t, var_names_B)
```

---

initval_calibr_par *Setting initial values of calibrated parameters*

Description

The initval_calibr_par function enables setting the initial values of calibrated parameters for the non-linear steady-state (equilibrium) solver or their expected values if calibration is omitted. If not set by this function, parameters' values are set to 0.5 by default.

Usage

```
initval_calibr_par(model, calibr_par, warnings = TRUE)
```

Arguments

model	an object of gecon_model class.
calibr_par	a named list or vector of calibrated parameters' initial values.
warnings	logical, should warnings be displayed?

Details

The values of calibrated parameters passed to the object of gecon_model class are treated as initial values for the steady-state solver when the user specifies calibrating equations in a .gcn file and requests that the steady_state function shall use it. If the calibration is omitted, the initial values of calibrated parameters are treated as their final values. Calibrated parameters have to be set to correct values upon the decision to make calibrating equations inactive.

Value

An updated object of gecon_model class representing the model. Generic functions such as print and summary allow to show the model's elements. The get_par_values function returns parameters' values.

Examples

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

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

# set initial values for calibrated parameters
rbc <- initval_calibr_par(rbc, calibr_par=list(alpha = 0.4))

# find the steady state and values of calibrated parameters
rbc <- steady_state(rbc)
get_par_values(rbc, c("alpha"))
```

---

initval_var *Setting initial values of variables.*

Description

The initval_var function 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, warnings = TRUE)
```

Arguments

model	an object of the gecon_model class.
init_var	a named list or vector of the initial values of variables.
warnings	logical, should warnings be displayed?

Value

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

Examples

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

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

# set initial values of variables
rbc <- initval_var(rbc, list(K_s = 10, C = 2, I = 0.5, Y = 2.5))
rbc <- initval_var(rbc, c(pi=0))
```

---

is.gecon_model *Is it an object of* gecon_model *class?*

Description

This function checks whether its argument is an object of gecon_model class.

Usage

```
is.gecon_model(x)
```

Arguments

x	an object to be checked

Value

Logical value.

list_calibr_eq *List calibrating equations*

Description

The list_calibr_eq function returns calibrating equations according to the specified indices.

Usage

```
list_calibr_eq(model, eq_idx = NULL)
```

Arguments

model	an object of gecon model class.
eq_idx	an integer/numeric value/vector specifying the indices of requested equations.

Value

A character (one-column) matrix with requested equations.

Examples

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

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

# list calibrating equations
list_calibr_eq(rbc, c(1))
```

list_eq *List model equations*

Description

The list_eq function returns equations according to the specified indices.

Usage

```
list_eq(model, eq_idx = NULL)
```

Arguments

model	an object of gecon_model class.
eq_idx	an integer/numeric value/vector specifying the indices of requested equations.

Value

A character (one-column) matrix with requested equations.

Examples

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

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

# get the 3rd and the 5th model equation
list_eq(rbc, c(3, 5))
```

---

load_model *Load model from* .model.R *file*

Description

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

Usage

```
load_model(filename)
```

Arguments

filename	the path to the .model.R file containing the model's functions and variables. The .model.R extension is optional.

Details

The .model.R file with the model specification has to be created first. It can be done with the make_model function taking a .gcn file with model specification as an argument.

Value

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

See Also

The make_model function in order to create an .R file with the model specification.

Examples

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

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

`make_model` *Make model from* `.gcn` *file*

Description

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

Usage

```
make_model(filename)
```

Arguments

`filename`	the path to the .gcn file containing model formulation. The `.gcn` extension is optional.

Details

Cf. gEcon users' guide, chapters 'Model description language' and 'Derivation of First Order Conditions'.

Value

An object of the `gecon_model` class 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 .gcn file directory. Additional files such as Latex documentation files or a logfile may be created when relevant options are set in the .gcn file.

See Also

`load_model` function to load a `.model.R` file created earlier.

Examples

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

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

`par_info` *Information about parameters*

Description

The `par_info` function allows to create an object of `gecon_par_info` class, containing information about the model's parameters. It allows to check types and values of a set of parameters as well as their incidence matrix.

Usage

```
par_info(model, parameters = NULL, all = FALSE)
```

Arguments

`model`	an object of the `gecon_model` class.
`parameters`	the names or indices of the parameters of interest.
`all`	logical value. If TRUE, information about all model parameters is generated (FALSE by default).

Details

If the function's result is not assigned to any variable, the information about the requested parameters is printed to the console.

Value

An object of `gecon_par_info` class.

See Also

`shock_info` for information about the shocks and `var_info` for information about the variables.

Examples

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

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

# find the steady state and the values of calibrated parameters
rbc <- steady_state(rbc)

# get information about all parameters
par_info(rbc, all = TRUE)
```

---

plot_simulation *Plot simulation results stored in an object of* gecon_simulation *class*

Description

The plot_simulation 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 gecon_simulation class.
to_eps	logical. if TRUE, the plot(s) is (are) saved as .eps file(s) in the model's subdirectory /plots and is (are) added to a .results.tex file.

Value

If more than five variables have been selected for simulations, at least two plots are created (max. 5 variables can be depicted on one plot). Separate plots are created for all the impulses, if the compute_irf function has been used for generating simulations.

Examples

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

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

# solve the model
```

```
rbc <- steady_state(rbc)
rbc <- solve_pert(rbc)

# compute and plot the IRFs
rbc <- set_shock_cov_mat(rbc, cov_matrix = matrix(0.01, 1, 1),
                         shock_order = "epsilon_Z")
irf_rbc <- compute_irf(rbc)
plot_simulation(irf_rbc)
```

---

print-methods *Print methods for classes in the gEcon package*

Description

Prints information about objects of the gecon_simulation, gecon_model, gecon_var_info, gecon_shock_info, and gecon_par_info classes.

Methods

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

signature(x = "gecon_model") Prints a short description of the model, its creation date, model's solution status, and more detailed information concerning model variables and parameters then the show generic function.

signature(object = "gecon_var_info") Prints the incidence matrix and the results that have been already obtained in terms of model variables.

signature(object = "gecon_shock_info") Prints the incidence matrix and the covariance matrix of shocks.

signature(object = "gecon_par_info") Prints the incidence matrix and information about parameters' types and values.

---

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

Description

The function generates random shock paths based on the shock covariance matrix specified by the user and simulates the behaviour of the system.

Usage

```
random_path(model, variables = NULL, sim_length = 40)
```

Arguments

model	an object of gecon_model class.
variables	the names or indices of variables whose paths are to be simulated. By default all variables are selected.
sim_length	the length of simulation path, the default value is 40.

Details

Cf. gEcon users' guide, chapter 'Model analysis'.

Value

An object of gecon_simulation class storing simulated paths of the model's variables.

See Also

The simulate_model function allows the user to specify her own shock paths and simulate their impact on the model's variables. This function also returns an object of gecon_simulation class. Generic functions such as print and summary provide information about simulations' results. The plot_simulation function allows to visualize the impact of simulated shock paths on the model's variables.

Examples

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

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

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

# perform simulation and plot the results
rbc <- set_shock_cov_mat(rbc, cov_matrix = matrix(0.01, 1, 1),
                         shock_order = "epsilon_Z")
rbc_rndpath <- random_path(rbc, variables = c("K_s", "C", "Z", "I", "Y"),
                           sim_length = 100)
plot_simulation(rbc_rndpath)
```

set_free_par *Setting free parameters' values*

Description

The set_free_par function allows to set values of free parameters occurring in a gecon_model class object.

Usage

```
set_free_par(model, free_par = NULL, reset = FALSE, warnings = TRUE)
```

Arguments

model	an object of gecon_model class.
free_par	a named list or a vector of parameters.
reset	a logical value, if TRUE, the function allows to reset free parameters to values from the .gcn file.
warnings	logical, if TRUE, a warning is displayed whenever the default parameter value (specified in the .gcn file) is overwritten.

Value

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

Examples

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

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

# set free parameters' values different from those in the .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_shock_cov_mat *Setting a covariance matrix of stochastic shocks.*

Description

The set_shock_cov_mat function allows to set a covariance matrix of model shocks.

Usage

```
set_shock_cov_mat(model, cov_matrix, shock_order = NULL)
```

Arguments

model	an object of gecon_model class.
cov_matrix	a numeric matrix. Shock covariance matrix.
shock_order	a character vector specifying the order of shocks in the cov_matrix. If not specified, shocks will be sorted according to their internal ordering (as reported by the get_shock_names function).

Details

The order of rows/columns of shock covariance matrix must agree with the internal order of shocks in a corresponding gecon_model-class object, unless the shock_order argument is supplied. Shocks' internal order can be checked with the shock_info function as well as the generic function print.

Value

An (updated) object of gecon_model class representing the model.

See Also

Generic functions such as print and summary allow to show the model's components. The shock_info function returns the names of shocks, information about equations which they appear in, and their current covariance matrix.

Examples

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

# make and load the model
rbc <- make_model("rbc.gcn")
shock_info(rbc, all = TRUE)

# set the shock covariance matrix
rbc <- set_shock_cov_mat(rbc, cov_matrix = matrix(0.01, 1, 1),
```

```
                          shock_order = "epsilon_Z")
shock_info(rbc, all = TRUE)
```

`set_shock_distr_par` *Setting distribution parameters of model shocks*

Description

The `set_shock_distr_par` function sets distribution parameters (standard deviations, correlations, etc.) of shocks in an object of `gecon_model` class.

Usage

```
set_shock_distr_par(model, distr_par)
```

Arguments

`model`	an object of `gecon_model` class.
`distr_par`	a list or a vector of distribution parameters with named elements.

Details

By default, `gEcon` uses an identity matrix as the covariance matrix of shocks. Valid parameters' names should match any of the following patterns:

`"sd( SHOCK_NAME )"`

`"var( SHOCK_NAME )"`

`"cov( SHOCK_NAME_1, SHOCK_NAME_2 )"`

`"cor( SHOCK_NAME_1, SHOCK_NAME_2 )"`

There are two issues which the user should be careful about while using the `set_shock_distr_par` function. First, in contrast to other parameters, shock distribution parameters require quotation marks to be assigned properly. If quotation marks are omitted, R parser treats elements of the `distr_par` list or vector as functions and attempts to evaluate them, producing errors. Second, parameters passed to the `distr_par` argument should not be specified twice.

Value

An (updated) object of the `gecon_model` class representing the model.

Examples

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

# make and load the model
home_production <- make_model("home_production.gcn")
shock_info(home_production, all = TRUE)
```

```
# set the shock distribution parameters
home_production <- set_shock_distr_par(home_production,
                                       distr_par = list("sd(epsilon_h)" = 0.7,
                                                        "var(epsilon_m)" = 0.49,
                                                        "cor(epsilon_m,
                                                             epsilon_h)" = 2/3))
# get information about shocks in the model
shock_info(home_production, all = TRUE)
```

---

shock_info *Information about shocks*

Description

The shock_info function prints information about the model's shocks (occurrence in equations, covariance matrix). It also allows to create an object of the gecon_shock_info class, which stores this information.

Usage

```
shock_info(model, shocks = NULL, all = FALSE)
```

Arguments

model	an object of gecon_model class.
shocks	the names or indices of shocks of interest.
all	logical value. If TRUE, information about all model shocks is generated (FALSE by default).

Value

An object of the gecon_shock_info-class.

Examples

```
# copy the example to the 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 the shock covariance matrix
rbc <- set_shock_cov_mat(rbc, cov_matrix = matrix(0.01, 1, 1),
                         shock_order = "epsilon_Z")

# get information about shocks
shock_info(rbc, all = TRUE)
```

show-methods *Show methods for classes in the gEcon package*

Description

This method prints general information about objects of the `gecon_simulation`, `gecon_model`, `gecon_var_info`, `gecon_shock_info`, and `gecon_par_info` classes.

Methods

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

`signature(object = "gecon_model")` Prints a short description of the model, its creation date, model's solution status, and the information about the numbers of model variables and parameters.

`signature(object = "gecon_var_info")` Prints the incidence matrix and the results that have been already obtained in terms of model variables.

`signature(object = "gecon_shock_info")` Prints the incidence matrix and the covariance matrix of shocks.

`signature(object = "gecon_par_info")` Prints the incidence matrix and information about parameters' types and values.

simulate_model *Simulation of the model*

Description

The `simulate_model` function simulates the impact of shock paths specified by the user on the model's variables. In particular, it allows to compute the impact of negative shocks.

Usage

```
simulate_model(model, variables = NULL, shocks = NULL,
               shock_path, sim_length = 40)
```

Arguments

`model`	an object of `gecon_model` class.
`variables`	the names or indices of variables whose paths are to be simulated. By default all variables are selected.
`shocks`	the names or indices of shocks corresponding to consecutive rows of the shock_path matrix. If missing, the `rownames` of the `shock_path` matrix are used.
`shock_path`	a matrix simulated paths of `shocks` in rows.
`sim_length`	the length of simulation path, the default value is 40.

Value

An object of gecon_simulation class storing simulated paths of the model's variables.

See Also

The random_path function generates random paths of shocks for the system behaviour simulation. This function also returns an object of gecon_simulation class. Generic functions such as print and summary provide information about simulations' results. The plot_simulation function allows to visualize the impact of simulated shock paths on the model's variables.

Examples

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

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

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

# set shock path
shock_path <- matrix(0, 1, 4)
shock_path[1, 1] <- -0.05
shock_path[1, 4] <- -0.05
# simulate model and plot results
rbc_sim <- simulate_model(rbc, variables = c("K_s", "C", "Z", "I", "Y"),
                          shocks = "epsilon_Z", shock_path = shock_path)
plot_simulation(rbc_sim)
```

---

solve_pert *Solve the model in a (log-)linearised form (the first order perturbation)*

Description

The function solves the first order perturbation of a model in a (log-)linearised form using Christopher Sims' gensys solver.

Usage

```
solve_pert(model, loglin = TRUE,
           loglin_var = NULL, not_loglin_var = NULL,
           tol = 1e-6, solver = "gensys")
```

Arguments

model	an object of gecon_model class.
loglin	a logical value. If TRUE, all variables are selected for log-linearisation.
loglin_var	a vector of variables that are to be log-linearised (effective only if the loglin argument is set to FALSE).
not_loglin_var	a vector of variables that are not to be log-linearised (overrides previous settings).
tol	a numeric value. Tolerance level of a solution (1e-6 by default).
solver	the name of the first order perturbation solver. The default solver is Christopher Sims' gensys solver.

Details

Cf. gEcon users' guide, chapter 'Solving the model in linearised form'.

Value

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

References

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

Examples

```
# copy the example to the current working directory
file.copy(from = file.path(system.file("examples", package = "gEcon"),
          "rbc.gcn"), to = getwd())
# make and load the model
rbc <- make_model("rbc.gcn")
# find the steady state
rbc <- steady_state(rbc)

# solve in log-linearised form
rbc <- solve_pert(rbc)
get_pert_solution(rbc)

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

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

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

Description

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

Usage

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

Arguments

model	an object of gecon_model class.
solver	the name of non-linear equations solver. Current gEcon version provides only an interface to the nleqslv function.
use_jac	logical. If TRUE, the Jacobian matrix generated by the symbolic library is used, else numerical derivatives are computed.
calibration	logical. If FALSE, calibrating equations will not be taken into account in the computation of the steady state (equilibrium) of a dynamic (static) model. Calibrated parameters' values will be fixed then at their initial levels.
options_list	a list of chosen nleqslv solver specific settings; the following options are available:

- method a character string with the name of the method to be used for solving non-linear system of equations. Available methods are: "Newton" and "Broyden", the default option is "Newton".
- global a character string with the name of global search strategy to be applied. Strategies provided are: "dbldog", "pwldog", "qline", "gline", "none", the default option is "qline".
- xscalm a character string with the name of the method for scaling initial values. It can be set to "fixed" or "auto". The default option is "fixed".
- max_iter a numeric value, the maximal number of iterations. The default value is 150.
- tol a numeric value setting a numeric tolerance for a solution (function value tolerance). The default value is 1e-6.
- xtol a numeric value setting a numeric tolerance for a solution (iteration relative step length tolerance). The default value is 1e-6.

solver_status	logical. Should the solver exit code be returned?

Details

Cf. gEcon users' guide, chapter 'Deterministic steady state & calibration'.

Value

An object of gecon_model class representing the model. Generic functions such as print and summary allow to show the model's components. The get_ss_values and get_par_values functions return steady state (equilibrium) and parameters' values, respectively.

See Also

nleqslv for the detailed description of the nleqslv solver capabilities. If the steady state has not been found, the get_residuals function can be used to check initial and final equations' residuals.

Examples

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

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

# find the 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)

# find the steady state without calibrating alpha
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 methods for classes in the gEcon package*

Description

This method summarizes information about objects of gecon_simulation, gecon_model, gecon_var_info, gecon_shock_info, and gecon_par_info classes.

Methods

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

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

`signature(object = "gecon_var_info")` Prints the incidence matrix and the results that have been already obtained in terms of model variables.

`signature(object = "gecon_shock_info")` Prints the incidence matrix and the covariance matrix of shocks.

`signature(object = "gecon_par_info")` Prints the incidence matrix and information about parameters' types and values.

`var_info` *Information about variables*

Description

The `var_info` function allows to create an object of `gecon_var_info` class, containing information about selected model variables. It allows to check variables' equation incidence matrix as well as the already computed statistics for them.

Usage

```
var_info(model, variables = NULL, all = FALSE)
```

Arguments

`model`	an object of `gecon_model` class.
`variables`	the names or indices of the variables of interest.
`all`	logical value. If TRUE, information about all model variables is generated (FALSE by default).

Details

The `var_info` function may be useful while debugging a model. It allows also to retrieve information quickly when a model is large. If the R command is not assigned to any R object, the information about the requested variables is printed to the console. Depending on which operations have been performed on an object of `gecon_model` class, the information printed consists of a subset of the following elements:

- incidence information,
- steady-state (equilibrium) values,
- variables info (which variables are log-linearised and which are state variables),
- state variables impact on the selected variables,
- shocks' impact on the selected variables,
- basic statistics,
- correlations.

Value

An object of `gecon_var_info-class` class.

Note

The function only displays and returns the model's characteristics that have been already set or computed. Eg. if the model has been solved but the statistics have not been computed, the correlations will not be passed to the object of `gecon_var_info` class.

See Also

`shock_info` for information about the shocks.
`get_ss_values`, `get_pert_solution`, `get_model_stats` to extract statistics of the model variables.

Examples

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

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

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

# get information about variables
rbc_var_info <- var_info(rbc, variables = c("K_s"))
print(rbc_var_info)
show(rbc_var_info)
summary(rbc_var_info)
```

Index