Package 'teems'

October 21, 2026

Title Trade and Environment Equilibrium Modeling System

Version 1.0.1

Description Equilibrium models are frequently employed to examine the potential impacts of economic, energy, and trade policies as well as form the foundation of most integrated assessment models. The 'teems' package handles all aspects of running equilibrium models while facilitating transparent and reproducible workflows.

License AGPL (>= 3)

URL https://teemsphere.github.io/, https://github.com/teemsphere/teems/

BugReports https://github.com/teemsphere/teems/issues

Depends R (>= 4.3.0)

Imports cli (>= 3.4.0), data.table (>= 1.15.0), graphics, grDevices, jsonlite (>= 1.8.0), purrr (>= 1.0.0), R6 (>= 2.4.0), rlang (>= 1.0.0), stats, tibble (>= 3.0.0), tools, utils

Suggests testthat (>= 3.0.0), withr (>= 3.0.0)

LinkingTo cpp11

Config/testthat/edition 3

Encoding UTF-8

Language en-US

RoxygenNote 7.3.3

SystemRequirements Docker (https://www.docker.com)

NeedsCompilation yes

Author Matthew Cantele [aut, cre, cph] (ORCID: <https://orcid.org/0000-0003-3729-1764>), Tom Kompas [ctb, ths] (ORCID: <https://orcid.org/0000-0002-0665-0966>), Ha Van Pham [ctb] (ORCID: <https://orcid.org/0000-0002-5927-6568>), Martin Ingram [ctb] (ORCID: <https://orcid.org/0000-0001-6269-8342>)

Maintainer Matthew Cantele <matthew.cantele@protonmail.com>

Repository CRAN

Date/Publication 2026-10-21 13:10:02 UTC

Contents

ems_compose . . . 2
ems_custom_shock . . . 4
ems_data . . . 5
ems_deploy . . . 8
ems_example . . . 10
ems_model . . . 12
ems_option_get . . . 14
ems_option_reset . . . 16
ems_option_set . . . 16
ems_probe . . . 19
ems_RK . . . 21
ems_scenario_shock . . . 25
ems_solve . . . 26
ems_swap . . . 32
ems_uniform_shock . . . 33
GTAP_convert . . . 34
plot.teems_probe . . . 36
solve_in_situ . . . 37

Index **41**

`ems_compose` *Compose model results into structured data objects*

Description

`ems_compose()` retrieves and processes results from a solved model run. Data validation and consistency checks are performed during composition.

Usage

```
ems_compose(cmf_path, which = "all", passes = FALSE)
```

Arguments

`cmf_path`	Character length 1, path to the CMF file generated by `ems_deploy()`.
`which`	Character vector of variable length (default `"all"`). When `"all"`, all model variables and coefficients (updated, post-simulation values, plus any PostSim coefficients typed `"postsim"`) are returned. Condensation coefficients (`CSUB<n>`) are returned only when named. Otherwise, a character vector of variable and/or coefficient names to retrieve (case insensitive). Note that some coefficient names differ from their associated headers (e.g., VTWR/VTMFSD). Coefficient values are read from the solver's binary coefficient dump (`sol.cof`/`sol.cbin`).
`passes`	Logical length 1 (default `FALSE`). When `TRUE`, each variable's data.table gains the three separate solutions of a `"Gragg"`, `"Midpoint"` or `"Euler"` run and their accuracy columns, and the output gains an `accuracy` attribute (see Details). An error is raised for a run that wrote no separate solutions.

Details

Every variable whose pre-simulation level is known gains the columns `pre_level` and `post_level`, plus `change` (the ordinary change) for a percent-change variable or `percent_change` for a change variable. The level is known for a variable declared with `ORIG_LEVEL=` (a coefficient over exactly the variable's sets, or a number) and for a levels variable; pre-simulation values come from the solver's `sol.cbin0`.

With `passes = TRUE`, each variable's data.table gains, in the units of `Value` where applicable:

- `pass1` to `pass3`: the separate solutions, fewest to most steps.
- `figures`: the significant figures on which the two extrapolations agree (0 to 6, `NA` when both are below 1e-6); `levels_figures`: the same for the levels value.
- `error`: the difference between the two extrapolations.
- `code`: the convergence code, best `"XMA"` to worst `"OD!"`; see GEMPACK manual 26.2.3.
- `summarized`: whether the element counts in the run's accuracy summary: `TRUE` for endogenous and backsolved results (of the `accuracy_vars` variables when those are given), `FALSE` for exogenous elements, whose values the shock sets.

With subintervals, `figures` and `levels_figures` are the lowest over the subintervals and `code` the worst; `pass1` to `pass3` and `error` come from the last subinterval alone, each pass compounded onto the extrapolated result of the subintervals before it. The per-variable summary, `attr(output, "accuracy")`, has `elements`, `summarized`, `min_figures`, `min_levels_figures` (a zero result counts as 6), `share_figures`, `share_levels_figures` (share at `accuracy_figures` or more, see `ems_option_set()`), `worst_code` and `max_error`.

Value

A tibble with columns "name", "label", "type", and "dat" (a list-column of data.tables) containing model results. For runs solved with an embedded Runge-Kutta method (`"BoSha32"`, `"DoPri54"`; see `ems_solve()`) each variable's data.table carries an additional `error_estimate` column: the solver's component-by-component estimate of the cumulative solution error, `|delta| / max(1, |Value|)`. Sorting on `error_estimate` ranks the results by how settled they are; the estimate is an indicator, not a bound on the error.

See Also

`ems_solve()` for solving the CGE model.

Examples

```
## Not run:
# The following examples require that a model run has taken
# place. See https://teemsphere.github.io/ to get started.

# Return all variables and coefficients
outputs <- ems_compose(cmf_path)

# Return specific variables and/or coefficients by name
outputs <- ems_compose(cmf_path, c("qfd", "EVFP"))
```

```
# Add the three separate solutions of a Gragg run
outputs <- ems_compose(cmf_path, "qgdp", passes = TRUE)

## End(Not run)
```

`ems_custom_shock` *Prepare a custom shock*

Description

Prepares heterogeneous shocks specified on a tuple-by-tuple basis, in the variable's own form (percentage or ordinary change). Only the tuples present in `input` are shocked.

Usage

```
ems_custom_shock(var, input)
```

Arguments

`var`	Character of length 1, the variable to be shocked.
`input`	Path to a CSV file, or a data frame or data frame extension (e.g., tibble, data.table). Must contain one column named `"Value"` with the shock for each tuple (percentage or ordinary change, as the variable is declared), plus one column per set index for the variable. Note that set names must conform to the model-specific hybrid format (e.g., `ACTSa`, `REGr`). Column order is inconsequential.

Value

A `list` object with shock configuration to be passed to the `shock` argument of `ems_deploy()`.

See Also

`ems_deploy()` for loading the output of this function, `ems_swap()` for changing the standard model closure.

Examples

```
# Set elements
sectors <- c("crops", "food", "livestock", "mnfcs", "svces")
regions <- c("asia", "eit", "lam", "maf", "oecd")
time_steps <- 0:7

# Create data frame
aoall_data <- expand.grid(ACTSa = sectors,
                          REGr = regions,
                          ALLTIMEt = time_steps)

aoall_data$Value <- runif(nrow(aoall_data))
head(aoall_data)
```

```
# Assign values to a model variable (aoall)
ems_custom_shock("aoall", aoall_data)

# In the case of temporally dynamic models, chronological
# year(s) specified by the column "Year" may be used instead
# of the the time set.
pop_data <- data.frame(REGr = "asia",
                       Year = 2023:2030)

pop_data$Value <- runif(nrow(pop_data))
pop_data

ems_custom_shock("pop", pop_data)
```

`ems_data` *Load and prepare data inputs*

Description

Loads input files, aggregates basedata and parameters according to set mappings, and returns prepared data for `ems_deploy()`. Accepts GTAP HAR files and objects generated by `GTAP_convert()` as inputs, or a single HAR file holding a complete non-GTAP database (see Details).

Usage

```
ems_data(
  dat_input,
  par_input = NULL,
  set_input = NULL,
  time_steps = NULL,
  par_weights = "value",
  ...
)
```

Arguments

`dat_input`	Character vector of length 1, file name in working directory or path to a data input containing basedata coefficient data, or the corresponding "dat" list object generated by `GTAP_convert()`.
`par_input`	Character vector of length 1, file name in working directory or path to a data input containing parameter coefficient data, or the corresponding "par" list object generated by `GTAP_convert()`. `NULL` (default) together with a `NULL` `set_input` selects the single-file route.
`set_input`	Character vector of length 1, file name in working directory or path to a data input containing set elements and attributes, or the corresponding "set" list object generated by `GTAP_convert()`. `NULL` (default) together with a `NULL` `par_input` selects the single-file route.

`time_steps` Integer vector of variable length (default is `NULL`). For use with temporally dynamic models. Input format is either chronological years of steps including initial reference year corresponding to the database employed or steps from t0 with initial value 0.

`par_weights` Character vector (default is `"value"`), the weights with which parameters are averaged when their sets are aggregated.

- `"value"`: each member is weighted by the value of its nest, the GTAP aggregation rule; the Armington, intermediate-input and value-added elasticities are weighted with world totals and take one value in every region.
- `"share"`: each member is weighted by the value of its CES or CET nest times one minus the sum of its squared input shares, so that the aggregate elasticity reproduces the members' response to a small relative price change. Members whose input shares cannot move (for example a commodity that is never imported) carry no weight, which can leave an aggregate far below the value-weighted elasticity where those members dominate it.

An unnamed element sets the method for every weighted parameter; elements named after a parameter header (e.g. `c("value", ESBM = "share")`) override it for that parameter. The CDE parameters (INCP, SUBP) are weighted by value under either method. Has no effect when only dat_input is given, since that route does not aggregate.

`...` A named pairlist assigning set mappings to model sets that are explicitly read (not constructed via any set operation defined in the model file). Set mappings can be selected from a range of internally available options or provided by the user.

- Internal mapping: character vector of length 1 corresponding to an internally available mapping.
- User-provided mapping: character vector of length 1, file name in working directory or path to a CSV file; a data frame or data frame extension (e.g., tibble, data.table).

See teems-mappings for internally available mappings and formatting for user-provided mappings.

Details

Set names passed via `...` must match set names as they appear in the database. Only sets requiring aggregation need to be specified. Unspecified sets whose elements are fully covered by a provided mapping are derived automatically. Any elements not covered by a provided mapping are passed through without aggregation.

Single-file route: when only `dat_input` is given, the HAR file is taken as a complete database in its own right (for example the ORANI-G `basedata.har`). Every character header with distinct entries becomes a set named after its header, every real header becomes a coefficient whose dimensions are the set labels stored in the file, and the sets are passed through without aggregation. No database version is read and no set mappings may be supplied on this route.

On the GTAP-E and GTAP-Power layers ELFVAEN (EFVE) follows FlexAgg's aggpar programs: coal, oil and gas mining are recalibrated to the target supply elasticities (SPLY), assuming as Burniaux and Truong (2002) do a Leontief top nest and perfectly elastic supply of every input other

than the natural resource, and the other fuels take 1. Two results are then overridden, reported when `verbose` is on, and listed in the metadata: a recalibrated value below 0.1, which a natural-resource rent too small to carry its target produces, keeps the aggregated member value; and an aggregate whose members all carry the database placeholder 1e-6 (the technologies without energy inputs) takes 1.

Value

A list of data.tables ready for input into the `.data` argument of `ems_deploy()`

See Also

`ems_deploy()` for loading the output of this function. `GTAP_convert()` for converting GTAP HAR files and data formats.

Examples

```
## Not run:
# The following examples require input data. See
# https://teemsphere.github.io/ to get started.

# Single-file database (non-GTAP), no aggregation
orani_data <- ems_data("oranig/basedata.har")

# Data for a static model using internal mappings
v7_data <- ems_data("v7_data/gsdfdat.har",
                    "v7_data/gsdfpar.har",
                    "v7_data/gsdfset.har",
                    REG = "AR5",
                    ACTS = "food",
                    ENDW = "labor_diff")

# The same aggregation with FlexAgg's value weights for the
# import-sourcing elasticities
v7_value <- ems_data("v7_data/gsdfdat.har",
                     "v7_data/gsdfpar.har",
                     "v7_data/gsdfset.har",
                     REG = "AR5",
                     ACTS = "food",
                     ENDW = "labor_diff",
                     par_weights = c(ESBM = "share"))

# Data for an intertemporal model with explicit time steps
int_data <- ems_data("v7_data/gsdfdat.har",
                     "v7_data/gsdfpar.har",
                     "v7_data/gsdfset.har",
                     REG = "WB23",
                     ACTS = "services",
                     ENDW = "labor_agg",
                     time_steps = c(0, 1, 2, 4, 6, 8, 10, 15))

# Data for an intertemporal model with chronological time
```

```
# steps and a user-provided mapping for the ENDW set
int_data <- ems_data("v7_data/gsdfdat.har",
                     "v7_data/gsdfpar.har",
                     "v7_data/gsdfset.har",
                     REG = "R32",
                     ACTS = "medium",
                     ENDW = "user/endowment/mapping.csv",
                     time_steps = c(2023, 2025, 2027, 2030, 2035))

## End(Not run)
```

`ems_deploy` *Validate and write all model inputs*

Description

Validates all data and model inputs, processes shocks and closure swaps, and writes the input files required by `ems_solve()`.

Usage

```
ems_deploy(
  .data,
  model,
  shock = NULL,
  swap_in = NULL,
  swap_out = NULL,
  shock_file = NULL,
  write_coefficients = FALSE
)
```

Arguments

`.data`	A list of data.tables generated by `ems_data()`.
`model`	A tibble generated by `ems_model()`.
`shock`	A shock object or list of shock objects produced by `ems_uniform_shock()`, `ems_custom_shock()`, or `ems_scenario_shock()`. When `NULL` (default), no shock is applied and input data is returned without change.
`swap_in`	A character vector of variable length, list generated by `ems_swap()`, or combination of the above provided as a list (default `NULL`). Variables to be made exogenous.
`swap_out`	A character vector of variable length, list generated by `ems_swap()`, or combination of the above provided as a list (default `NULL`). Variables to be made endogenous.
`shock_file`	A character vector length 1 (default `NULL`). Path to a file with .shf extension representing a fully prepared shock file. No checks or modifications are carried out on this file.

write_coefficients
: Logical of length 1 (default FALSE). Coefficient values return to R through the solver's binary coefficient dump, read selectively by ems_compose(). Set TRUE to also emit per-coefficient CSV files in out/coefficients/ via auto-generated File/Write statements in the GEMPACK spreadsheet-style text data file format.

Value

Path to the CMF file, to be passed to ems_solve().

See Also

ems_data() for loading and preparing data inputs. ems_model() for parsing and modifying model and closure files. ems_uniform_shock(), ems_custom_shock(), and ems_scenario_shock() for preparing shocks. ems_swap() for preparing closure swaps. ems_solve() for utilizing the output of this function.

Examples

```
## Not run:
# The following examples require input data. See
# https://teemsphere.github.io/ to get started.

# Uniform shock: 1% real GDP growth in every region
shock <- ems_uniform_shock("qgdp", value = 1)

# Full variable swap, qgdp for aoreg (GDP target, endogenous
# productivity)
cmf_path <- ems_deploy(
  .data = dat,
  model = model,
  shock = shock,
  swap_in = "qgdp",
  swap_out = "aoreg"
)

# Partial variable swap (element "row" of set REG with index r)
yp_row <- ems_swap("yp", REGr = "row")
dppriv_row <- ems_swap("dppriv", REGr = "row")

# Swap part of yp for part of dppriv; qgdp for aoreg
cmf_path <- ems_deploy(
 .data = dat,
 model = model,
 shock = shock,
 swap_in = list(yp_row, "qgdp"),
 swap_out = list(dppriv_row, "aoreg")
)

## End(Not run)
```

`ems_example` *Write teems example models and scripts to file*

Description

The teems package includes example models and scripts. This function facilitates access to both, allowing new users to immediately peruse compatible models or generate scripts containing a range of package features.

Usage

```
ems_example(
  model,
  path,
  type = c("model_files", "scripts"),
  dat_input = NULL,
  par_input = NULL,
  set_input = NULL
)
```

Arguments

`model`	Character vector of length 1, name of model. Options include: • `"GTAPv7"`: Standard GTAP model (version 7.1) • `"GTAP-RE"`: Intertemporal GTAP with rational expectations: forward-looking investment, all periods solved at once • `"GTAP-AEZ"`: GTAPv7 with land use by agro-ecological zone, land entering production as a CES composite of the zones (requires a GTAP-AEZ database) • `"GTAP-E"`: GTAPv7 with energy substitution and carbon taxes, a capital-energy composite inside value added (requires a GTAP-E database) • `"GTAP-EP"`: GTAP-E with electricity split into transmission and distribution and generation, and generation into base-load and peak-load technologies (requires a GTAP-Power database) • `"GTAPv6"`: Classic GTAP model (version 6.2)
`path`	Character vector of length 1. Directory where files will be written.
`type`	Character vector of length 1, (default is `"model_files"`). Determines the example output. Options include: • `"model_files"`: Model file and closure file • `"scripts"`: Example scripts for the chosen `model`. `dat_input`, `par_input`, and `set_input` are required.
`dat_input`	Character vector of length 1 (default is `NULL`, only used when `type = "scripts"`), file name in working directory or path to a data input containing basedata coefficient data, or the corresponding "dat" list object generated by `GTAP_convert()`.

par_input — Character vector of length 1 (default is NULL, only used when type = "scripts"), file name in working directory or path to a data input containing parameter coefficient data, or the corresponding "par" list object generated by GTAP_convert().

set_input — Character vector of length 1 (default is NULL, only used when type = "scripts"), file name in working directory or path to a data input containing set elements and attributes, or the corresponding "set" list object generated by GTAP_convert().

Value

For type = "model_files": a named character vector with elements model_file and closure_file. For type = "scripts": a character vector of paths to the written script files.

See Also

GTAP_convert() for converting GTAP HAR files and data formats.

Examples

```
# Retrieve GTAPv7 model files
ems_example("GTAPv7", tempdir())

## Not run:
# The following examples require input data. See
# https://teemsphere.github.io/ to get started.

# Generate GTAPv7 example scripts
ems_example(model = "GTAPv7",
            path = "path/to/example/scripts",
            type = "scripts",
            dat_input = "v7_data/gsdfdat.har",
            par_input = "v7_data/gsdfpar.har",
            set_input = "v7_data/gsdfset.har")

# Generate GTAP-RE example scripts
ems_example(model = "GTAP-RE",
            path = "path/to/example/scripts",
            type = "scripts",
            dat_input = "v7_data/gsdfdat.har",
            par_input = "v7_data/gsdfpar.har",
            set_input = "v7_data/gsdfset.har")

# Generate GTAP-AEZ example scripts (a GTAP-AEZ database is
# required; ems_data() prepares its AEZ layer automatically)
ems_example(model = "GTAP-AEZ",
            path = "path/to/example/scripts",
            type = "scripts",
            dat_input = "aez_data/gsdfdat.har",
            par_input = "aez_data/gsdfpar.har",
            set_input = "aez_data/gsdfset.har")

# Generate GTAP-E example scripts (a GTAP-E database is required;
# ems_data() prepares its energy layer automatically)
```

```
ems_example(model = "GTAP-E",
            path = "path/to/example/scripts",
            type = "scripts",
            dat_input = "e_data/gsdfdat.har",
            par_input = "e_data/gsdfpar.har",
            set_input = "e_data/gsdfset.har")

# Generate GTAP-EP example scripts (a GTAP-Power database is
# required; ems_data() prepares its energy and power layer
# automatically)
ems_example(model = "GTAP-EP",
            path = "path/to/example/scripts",
            type = "scripts",
            dat_input = "power_data/gsdfdat.har",
            par_input = "power_data/gsdfpar.har",
            set_input = "power_data/gsdfset.har")

# Generate GTAPv6 example scripts from a v7 format database
converted <- GTAP_convert(dat_har = "v7_data/gsdfdat.har",
                          par_har = "v7_data/gsdfpar.har",
                          set_har = "v7_data/gsdfset.har",
                          target = "GTAPv6")

ems_example(model = "GTAPv6",
            path = "path/to/example/scripts",
            type = "scripts",
            dat_input = converted$dat,
            par_input = converted$par,
            set_input = converted$set)

## End(Not run)
```

`ems_model` *Parse and modify model and closure files*

Description

Parses the model and closure files, conducts pre-deployment checks, and carries out specified modifications.

Usage

```
ems_model(
  model_file,
  closure_file,
  backsolve = NULL,
  ignore_condense = FALSE,
  ...
)
```

Arguments

model_file	Character vector length 1, path to a TABLO file with .tab extension. See teems-models for vetted models and compatible Tablo file formatting.
closure_file	Character vector length 1, path to a closure file with .cls extension. See teems-models for closure file formatting.
backsolve	Character vector of variable length (default is NULL), variable names to substitute out of the system by backsolving. Unnamed entries resolve their defining equation by the E_<variable> convention; use named entries for exceptions (e.g., c(qgd = "E_qgdtot")). Backsolved variables must be endogenous; their values remain available in solve outputs. Each nomination is checked to make sure its equation can be solved for the variable (see GEMPACK manual 14.1.10); a nomination that fails aborts with the reason.
ignore_condense	Logical length 1 (default is FALSE). If TRUE, in-TAB Substitute, Backsolve and Omit statements are ignored. Otherwise Substitute statements are executed as backsolves and Omit statements make the omitted variables exogenous where the closure does not already. The solver automatically drops unshocked exogenous variables from the solve, and their zero results remain in the outputs.
...	A named pairlist assigning values to model coefficients. Name must match a coefficient declared in the model file. Value may be a length-1 numeric, a data frame or data frame extension (e.g., tibble, data.table), or a path to a CSV file.

Details

If a length-1 numeric is supplied via ..., this value is applied uniformly across all tuples for that coefficient. For heterogeneous values, data frame and CSV inputs must contain a Value column and one column per set index using the model-specific naming convention (set name concatenated with its index letter, e.g., REGr, COMMc). Inputs are subject to structure checks against the set aggregations specified in ems_data() and all tuples must be present.

Value

A tibble comprising parsed model statements to be passed to the model argument of ems_deploy().

See Also

ems_example() for writing teems example models and scripts to file. ems_deploy() for loading the output of this function as well as conducting closure swaps.

Examples

```
# Retrieve the static and the intertemporal example models
GTAPv7 <- ems_example("GTAPv7", tempdir())
GTAP_RE <- ems_example("GTAP-RE", tempdir())

# Simple static model without condensation
ems_model(GTAPv7[["model_file"]],
          GTAPv7[["closure_file"]],
          ignore_condense = TRUE)
```

```
# Construct data frame
sectors <- c("crops", "food", "livestock", "mnfcs", "svces")
regions <- c("usa", "chn", "row")
time_steps <- 0:5

SUBPAR <- expand.grid(COMMc = sectors,
                      REGr = regions,
                      ALLTIMEt = time_steps)
SUBPAR$Value <- runif(nrow(SUBPAR))

# Model load with:
# 1) uniform numeric value applied to KAPPA coefficient
# 2) heterogeneous values allocated to SUBPAR via data frame

ems_model(model_file = GTAP_RE[["model_file"]],
          closure_file = GTAP_RE[["closure_file"]],
          KAPPA = 0.03,
          SUBPAR = SUBPAR)
```

`ems_option_get` *Get default package options*

Description

`ems_option_get()` returns default package options. If `name` is `NULL` (the default), all option values are returned as a list.

Usage

```
ems_option_get(name = NULL)
```

Arguments

`name`	Name of the option for which to retrieve a value. One of:

- `NULL` Returns all option values.
- `"tempdir"` Character. Directory used for temporary file storage during a model run.
- `"verbose"` Logical. If `FALSE`, function-specific diagnostics are silenced.
- `"ndigits"` Integer. Number of digits to the right of the decimal point written to file for numeric type double (at least `ndigits` significant digits for small values).
- `"accuracy_threshold"` Numeric. Share of results that must be accurate to at least `accuracy_figures` figures.
- `"accuracy_figures"` Integer. Significant figures a result must reach to count as accurate (the default is `4`).

- "accuracy_criterion" Character. "solution" (the default), "data" or "both": the accuracy summary judged.
- "required_figures" Integer or NULL (the default). Figures for the convergence code XRA.
- "full_exclude" Character vector. Headers to fully exclude from all aspects of the model run.
- "check_shock_status" Logical. If FALSE, no check on shock element endogenous/exogenous status is conducted.
- "timestep_header" Character. Header written with the time of each period (time_steps of ems_data()).
- "n_timestep_header" Character. Header written with the number of periods.
- "docker_tag" Character. Tag of the teems Docker image to use. "latest" when unset.
- "assertions" Character. "fatal" (the default), "warn" or "off": severity of TAB Assertion failures in ems_solve().
- "range_test_initial" Character. "fatal" (the default), "warn" or "off": severity of declared-range violations on initial values.
- "range_test_updated" Character. "warn" (the default), "fatal" or "off": as "range_test_initial", for updated values.
- "random_seed" Integer. Seed of RANDOM(a,b) (the default is 1).
- "refine" Logical. TRUE (the default): one refinement step per "DBBD" solve (see ems_option_set()).
- "converge" Logical. FALSE (the default); TRUE applies GEMPACK's treatment of poorly converging results (see ems_option_set()).

Value

If name is NULL, a named list of all current option values. Otherwise, the value of the requested option.

See Also

ems_option_set() for setting package options. ems_option_reset() for resetting package options.

Examples

```
# Retrieve all options values
ems_option_get()

# Retrieve option value for `ndigits`
ems_option_get("ndigits")
```

ems_option_reset *Reset to default package options*

Description

ems_option_reset() resets all package options to default values.

Usage

```
ems_option_reset()
```

Value

invisible(NULL), called for its side effects.

See Also

ems_option_set() for setting package options. ems_option_get() for retrieving package options.

Examples

```
# Set multiple options
ems_option_set(verbose = FALSE, ndigits = 8)

# Retrieve modified option value for `verbose`
ems_option_get("verbose")

# Reset options to default values
ems_option_reset()

# Retrieve default option value for `verbose`
ems_option_get("verbose")
```

ems_option_set *Set advanced package options*

Description

ems_option_set() allows the user to customize advanced options.

Usage

```
ems_option_set(
  tempdir = NULL,
  verbose = NULL,
  ndigits = NULL,
  accuracy_threshold = NULL,
  accuracy_figures = NULL,
  accuracy_criterion = NULL,
  required_figures = NULL,
  full_exclude = NULL,
  check_shock_status = NULL,
  timestep_header = NULL,
  n_timestep_header = NULL,
  docker_tag = NULL,
  assertions = NULL,
  range_test_initial = NULL,
  range_test_updated = NULL,
  random_seed = NULL,
  refine = NULL,
  converge = NULL
)
```

Arguments

`tempdir`	A character vector length 1. Default is what is returned by `tempdir()`. Directory where input files, model diagnostics, and output files are written as well as where the model run takes place (via Docker bind mount).
`verbose`	Logical of length 1 (default is `TRUE`). If `FALSE`, function-specific diagnostics are silenced.
`ndigits`	Integer (default is `6`). Number of digits to the right of the decimal point kept when numeric type double data are written to file. Values smaller than 0.1 in magnitude keep `ndigits` significant digits instead, so small nonzero data are never rounded to zero.
`accuracy_threshold`	Numeric length 1 (default `0.8`). The share of results that must be accurate to at least `accuracy_figures` significant figures under `accuracy_criterion`; a warning is generated if the share is lower.
`accuracy_figures`	Integer length 1 from 1 to 6 (default `4`). The significant figures a result must reach to count as accurate.
`accuracy_criterion`	Character length 1 (default `"solution"`). Which accuracy summary is judged: `"solution"` the levels values of the endogenous results, `"data"` the values read from data files and updated, `"both"` the two together. The criteria of GEMPACK's automatic accuracy (GEMPACK manual 26.4), whose default is `"data"`. A run that updates no data is judged on the solution under `"data"` and `"both"`.

required_figures
: Integer length 1 from 1 to 6 (default NULL, none). Results of a three-solution "Gragg", "Midpoint" or "Euler" run whose two extrapolations agree to at least this many figures get the convergence code XRA in the accuracy summary (GEMPACK manual 26.2.6, RQF). Reporting only.

full_exclude
: A character vector of variable length (default is c("DREL", "DVER", "XXCR", "XXCD", "XXCP", "SLUG", "EFLG")). Headers to fully exclude from all aspects of the model run. Failure to designate these headers properly will result in errors. Modify with caution.

check_shock_status
: Logical of length 1 (default is TRUE). If FALSE, no check on the endogenous/exogenous status of a variable designated for a shock is conducted.

timestep_header
: A character vector length 1 (default is "YEAR"). Header written with the time of each period, as given in the time_steps argument of ems_data() (years, or steps from the initial period); the model takes the differences between consecutive periods as interval lengths. For novel intertemporal models — modify with caution.

n_timestep_header
: A character vector length 1 (default is "NTSP"). Header written with the number of periods (the length of time_steps). For novel intertemporal models — modify with caution.

docker_tag
: Character length 1 (default "latest"). Tag of the teems Docker image to use.

assertions
: Character length 1, "fatal" (default), "warn" or "off". Severity of TAB Assertion statement failures: "warn" reports and continues, "off" skips the checks.

range_test_initial
: Character length 1, "fatal" (default), "warn" or "off". Severity of declared-range violations (e.g. (ge 0)) on initial values: every coefficient, read or computed, is tested once the initial Reads and Formulas are done.

range_test_updated
: Character length 1, "warn" (default), "fatal" or "off". As range_test_initial, for updated values (range test updated values).

random_seed
: Integer (default is 1). Seed of the TAB function RANDOM(a,b). The same seed reproduces every draw; each Formula element keeps its draw across steps and extrapolation passes. Change the seed for a new sequence.

refine
: Logical length 1 (default TRUE). One step of iterative refinement after every linear solve of matrix_method = "DBBD" in ems_solve(), which brings its accuracy to that of "LU". Refinement holds the system matrix and the block factorizations through each solve, so it raises peak memory; the memory estimate counts it. "LU" and "SBBD" are in practice as accurate as a single LU solve and are not refined; "NDBBD" is not refined, as it cannot yet keep its factorization for a second solve.

converge
: Logical length 1 (default FALSE). If TRUE, ems_solve() applies GEMPACK's treatment of poorly converging results (see GEMPACK manual 26.2.5) to a run extrapolated from three multi-step solutions: a component whose three solutions

lie very close together or near zero reports their average, one whose solutions do not converge cleanly reports the solution with the most steps, and the rest are extrapolated. Not available with subtotals.

Value

`invisible(NULL)`, called for its side effects.

See Also

`ems_option_get()` for retrieving package options. `ems_option_reset()` for resetting package options.

Examples

```
# Set multiple options
ems_option_set(verbose = FALSE, ndigits = 8)

# Retrieve value of `tempdir`
ems_option_get("tempdir")

# Reset options to default values
ems_option_reset()
```

---

ems_probe *Structural probe of a deployed model*

Description

Runs the solver up to the point where the equation system is assembled, without solving, and checks that system. It reports whether the closure is valid, and names the elements at fault when it is not, as well as attempts to judge whether backsolving is advantageous. It also recommends the matrix method, task count and thread count.

Usage

```
ems_probe(cmf_path, components = TRUE, cores = NULL, memory = NULL, ...)
```

Arguments

`cmf_path`	Character length 1, path to the CMF file written by `ems_deploy()`.
`components`	Logical length 1 (default `TRUE`). Also find the simultaneous components of the system, the groups of equations that must be solved together, and the make-up of the largest; needed by `plot(x, type = "components")`.
`cores`	Integer length 1 or `NULL` (default). The number of cores the recommendation is made for. `NULL` reads it from the container the solver runs in (on Docker Desktop, the Resources setting); give a number for another machine.

memory	Numeric length 1 or NULL (default). The memory in GB the recommendation is made for, read from the container when NULL.
...	Named solver arguments, the same ones ... of ems_solve() accepts: the MA48 workspace guesses (laA, laD, laDi) and the expert solver flags. The Runge-Kutta controls are accepted and ignored.

Details

A probe runs on one task. It takes seconds on a condensed GTAP model and minutes at a few million uncondensed equations. A singular system is reported, not raised as an error: the printed object names the undetermined variables and the over-constrained equations, plot(x, type = "dm") locates them and x$defects lists every element.

Value

A teems_probe object: valid; the structural and realized (base-data) rank patterns; the defects, statements and incidence tibbles; components; structure (the time chain and block partition measured); condense; recommendation, with its status; and paths. print() gives the verdict and the recommendation, summary() the full report and plot.teems_probe() the pictures.

Condensation

Backsolving shrinks the system but densifies the blocks the parallel methods split along; on GTAP-RE the standard condensation makes "SBBD" runs much slower. The probe's verdict on the deployment is in x$condense$verdict:

- "slower": condensed, with a block partition and either a time chain or more than about 120 thousand condensed equations; try redeploying without backsolve.
- "lu_fine": condensed, with a partition, below that size; threaded "LU" on the condensed system was the fastest measured.
- "faster": condensed, without a partition; the system is "LU"-bound and condensation tends to speed it.
- "candidate": uncondensed, "LU"-bound and a million equations or more; backsolve in ems_model() may help.
- "none": no verdict.

Recommendation

From the measured structure, the deploy metadata and a memory model per method, the probe suggests a matrix method with its tasks and threads for the machine (the container's, or the cores and memory given). It prints as a ready-to-paste ems_solve() call and is kept in x$recommendation. In three cases the probe makes no recommendation and recommendation$status says why: "singular", the closure must be fixed first; "no_host", the container's cores could not be read, so pass cores and memory; "wont_fit", no method fits in the memory. A system below 300 thousand equations, the smallest size measured, is sent to "LU". The rules and the measurements behind them are in the solving and benchmarks chapters of the manual. ems_probe() is experimental: its rules come from a limited set of models and machines, so a different method, task count or thread count may be faster for yours. Try alternatives with ems_solve(), and report a suggestion that is off at https://github.com/teemsphere/teems/issues.

See Also

ems_deploy() for writing the CMF file; ems_solve() for the solve; plot.teems_probe().

Examples

```
## Not run:
# The following examples require the teems solver to be built.
# See https://teemsphere.github.io/ to get started.

probe <- ems_probe(cmf_path)
probe
plot(probe, type = "incidence")
plot(probe, type = "components")

## End(Not run)
```

---

ems_RK *Solve a model with a Runge-Kutta method*

Description

An ems_solve() wrapper restricted to the Runge-Kutta methods, with the step controls that exist only for them and defaults chosen for them: adaptive "DoPri54" from four initial steps.

Usage

```
ems_RK(
  cmf_path,
  solution_method = c("DoPri54", "BoSha32", "RK4", "Heun", "RK2"),
  steps = 4L,
  eps_tolerance = 0.01,
  adaptive = NULL,
  matrix_method = c("LU", "DBBD", "SBBD", "NDBBD"),
  n_tasks = 1L,
  n_threads = 1L,
  precision = c("single", "double"),
  n_subintervals = 1L,
  verbosity = 1L,
  suppress_outputs = FALSE,
  terminal_run = FALSE,
  accuracy_vars = NULL,
  max_retries = 3L,
  retry_adjust = 0.5,
  chart = c("log", "percent"),
  error_norm = c("max", "rms"),
  controller = c("std", "pi"),
  scope = c("pct", "all"),
```

```
  h_init = NULL,
  guard = NULL,
  ...
)
```

Arguments

cmf_path	Character length 1, path to the CMF file generated by ems_deploy().
solution_method	The method: "DoPri54" (default), "BoSha32", "RK4", "Heun" or "RK2". "DoPri54" and "BoSha32" are embedded pairs, so each step also yields an error estimate for adaptive control to act on; "RK4", "Heun" and "RK2" take fixed steps. "DoPri54" measured the most accurate per step on large shocks and "RK4" with a few steps the best cheap fixed-step method. When most retries come from step checks rather than from eps_tolerance, the cheaper steps of "BoSha32" can make it faster (Schiffmann 2022).
steps	Integer length 1 (default 4L), the number of steps; under adaptive control, the number to start from. Runge-Kutta methods do not extrapolate, so there is no step-count triple.
eps_tolerance	Numeric length 1 (default 0.01), the error estimate a step may not exceed under adaptive control. On large shocks 0.01 gives four correct digits on most elements; 0.001 adds the last digits at about 1.6 times the cost. GEMPACK suggests 0.1, which measured rougher on large shocks. Ignored when adaptive = "no".
adaptive	Character length 1. NULL (default) means "yes" for "DoPri54" and "BoSha32" and "no" for the fixed-step methods, which have no error estimate. • "no": fixed steps. • "yes": a step whose error estimate exceeds eps_tolerance is redone shorter, and a step that passes may lengthen the next one, by at most a factor of two either way. A step that fails a step check (a percentage-change variable crossing -100%, a coefficient range test, a TAB assertion or the guard) is also retried shorter. • "accuracy-only": as "yes", but step checks do not trigger a retry.
matrix_method	Character of length 1, matrix solution method (default is "LU"). Choices: • "LU": Standard LU decomposition, the most robust and potentially slowest for a large model. For use with both static and dynamic models. • "DBBD": Doubly bordered block diagonal, parallel solution for static models partitioned by their regional set. • "SBBD": Singly bordered block diagonal, parallel solution for intertemporal models partitioned along the time chain. • "NDBBD": Nested doubly bordered block diagonal, for intertemporal models whose "SBBD" host copy does not fit the memory available. ems_probe() (experimental) suggests the method and the tasks and threads to run it with for a deployment and a machine, and prints the call to paste (see Details).
n_tasks	Integer length 1 (default is 1L), number of tasks to run in parallel. Must be 1L if "matrix_method" == "LU".

`n_threads` Integer length 1 (default `1L`), OpenMP threads per MPI task. `"LU"`, `"SBBD"` and `"NDBBD"` give byte-identical results at any thread count; `"DBBD"` with more than one thread agrees only to rounding (see the solving chapter of the user manual).

`precision` Character length 1, `"single"` (default) or `"double"`: floating-point precision of the solver's coefficient storage. `"single"` halves memory traffic and suffices for ordinary step counts. `"double"` selects the double-precision solver binary shipped in the same image, removing the single-precision update-rounding floor that deep extrapolation ladders and many-step large-shock runs otherwise converge onto (per-pass rounding accumulates with step count, so beyond the floor extra steps stop helping). Roughly doubles coefficient-array memory. The effective precision is recorded in `sol.stats.json` and the model diagnostics solve record.

`n_subintervals` Integer length 1 (default is `1L`), number of subintervals for the applied shock. More subintervals may alleviate accuracy issues stemming from large shock magnitudes. In a model with active complementarities each subinterval runs its own approximate and accurate run; the Runge-Kutta methods take subintervals only there.

`verbosity` Integer length 1, `0`, `1` (default) or `2`. Console detail while the solver runs: `0` shows errors, warnings and the accuracy figures; `1` adds phase progress and timings; `2` shows every solver line and adds per-rank and per-block debug detail. The full solver log, including the per-subinterval and updated-data accuracy summaries, the convergence codes and every Zerodivide warning, is always written to `model_diagnostics.txt` in the run directory.

`suppress_outputs` Logical length 1 (default is `FALSE`). When `TRUE` solver outputs are not automatically converted into structured data with `ems_compose()`.

`terminal_run` Logical length 1 (default is `FALSE`). When `TRUE`, the function is exited without running the solver. This allows the user to close any R IDE or other programs prior to running from the terminal. When `TRUE` solver outputs are not automatically converted into structured data with `ems_compose()`.

`accuracy_vars` Character vector of variable names (default `NULL`). When given, the accuracy summary of a three-solution `"Gragg"`, `"Midpoint"` or `"Euler"` run covers just these variables, exogenous ones included, instead of every endogenous result. Under the `"solution"` and `"both"` accuracy criteria (see `ems_option_set()`) the warning then judges only them, so a run can be held to the results it is about.

`max_retries` Integer length 1 (default `3L`). How many times a step that fails a step check is retried shorter before the run aborts.

`retry_adjust` Numeric length 1 in (0, 1) (default `0.5`). The factor the step is shortened by on each retry.

`chart` Character length 1, the coordinates the stages are combined in. `"log"` (default) carries each percentage-change variable as the logarithm of its level ratio, so levels stay positive whatever the step and no percentage-change variable can reach `-100%`; the methods keep their order (Munthe-Kaas 1999). `"percent"` combines the stages in percentage changes from the initial level.

`error_norm` Character length 1. Whether the worst element (`"max"`, default) or the root mean square over elements (`"rms"`) is compared with `eps_tolerance`.

controller	Character length 1, the step-size rule: "std" (default), or "pi", which also weighs the previous step's estimate and rejects fewer steps on rough solution paths.
scope	Character length 1, which elements the accept test looks at: "pct" (default), the percentage-change variables, whose error estimate is dimensionless; or "all", which adds the ordinary-change variables. Their estimate is in their own units, so an accumulator in dollars can reject every step early in the path. error_estimate is reported for every element either way.
h_init	Numeric length 1 in (0, 1/steps] (default NULL, which is 1/steps), the first step length. Under adaptive control the first step is also capped from the initial gradient.
guard	Numeric length 1 greater than 1 (default NULL, no guard), chart = "log" only: the level ratio beyond which a stage is rejected and the step retried.
...	Named extras, as ... of ems_solve(): the MA48 workspace guesses (laA, laD, laDi) and the expert solver flags. The Runge-Kutta controls are formal arguments here, so their rk_* names are refused.

Value

As ems_solve().

References

Schiffmann, F. (2022), "Runge Kutta integrators for fast and accurate solutions in GEMPACK", GEMPACK Software, Centre of Policy Studies, Victoria University, Melbourne (14 April).

Munthe-Kaas, H. (1999), "High order Runge-Kutta methods on manifolds", Applied Numerical Mathematics, vol. 29, pp. 115-127.

Dormand, J.R. and Prince, P.J. (1980), "A family of embedded Runge-Kutta formulae", Journal of Computational and Applied Mathematics, vol. 6, pp. 19-26.

Bogacki, P. and Shampine, L.F. (1989), "A 3(2) pair of Runge-Kutta formulas", Applied Mathematics Letters, vol. 2, pp. 321-325.

See Also

ems_solve().

Examples

```
## Not run:
# The following examples require the teems solver to be built.
# See https://teemsphere.github.io/ to get started.

# Adaptive DoPri54 (the defaults):
ems_RK(cmf_path)

# Fixed-step RK4:
ems_RK(cmf_path, solution_method = "RK4", steps = 8L)

## End(Not run)
```

`ems_scenario_shock` *Prepare a scenario shock*

Description

Prepares a trajectory shock for an intertemporal model from absolute values: a value per pre-aggregation tuple and year, in the units of the variable's data (million USD, persons). At deployment the values are summed over the aggregation of `ems_data()`, and each year's aggregated value becomes the percentage change from the base-year value (the ordinary change for a variable declared `(change)`). The same input therefore serves any aggregation. Because the values are summed, scenario shocks suit additive variables, values and quantities.

By default `input` must hold every pre-aggregation tuple of the variable in every year. With `partial = TRUE` it may hold only the tuples behind some aggregated tuples, each complete: an aggregated tuple covered in part stops the deployment by name, and the aggregated tuples not covered keep a zero change. See the scenario shocks chapter of the manual for a worked population trajectory.

Usage

```
ems_scenario_shock(var, input, partial = FALSE)
```

Arguments

`var`	Character of length 1, the variable to be shocked.
`input`	Path to a CSV file, or a data frame or data frame extension (e.g., tibble, data.table), with a `Value` column of absolute values, a `Year` column of chronological years including the base year, and one column per remaining set index of the variable in the model-specific format (e.g., `REGr`). Column order is inconsequential.
`partial`	Logical of length 1 (default `FALSE`). If `TRUE`, `input` may cover a subset of the variable's aggregated tuples; only those are shocked, and they must be exogenous.

Value

A `list` object with shock configuration to be passed to the `shock` argument of `ems_deploy()`.

See Also

`ems_deploy()` for loading the output of this function. `ems_custom_shock()` for shocks given directly as percentage or ordinary changes at the model's aggregation.

Examples

```
# A trajectory for part of the world: three pre-aggregation
# regions that the aggregation of ems_data() groups into one
# region, in millions, base year 2017 first. At deployment the
# three are summed and each later year becomes the percentage
# change from 2017. Without partial = TRUE every region of the
```

```
# database would have to be present; the manual's population
# chapter builds such a trajectory from the POP header.
pop <- expand.grid(REGr = c("aus", "nzl", "xoc"),
                   Year = 2017:2020,
                   stringsAsFactors = FALSE)
pop$Value <- c(24.6, 4.8, 1.9) * 1.015^(pop$Year - 2017)
pop

ems_scenario_shock("pop", pop, partial = TRUE)

# The same from a CSV file
csv <- tempfile(fileext = ".csv")
write.csv(pop, csv, row.names = FALSE)
ems_scenario_shock("pop", csv, partial = TRUE)
```

`ems_solve` *Configure and solve model*

Description

Calls the teems-solver Docker image to solve the model's nonlinear equation system. Runs singularity, accuracy, and error checks on the results and structures outputs with `ems_compose()`.

Usage

```
ems_solve(
  cmf_path,
  solution_method = c("Gragg", "Johansen", "Euler", "Midpoint", "RK2", "Heun", "RK4",
    "BoSha32", "DoPri54"),
  matrix_method = c("LU", "DBBD", "SBBD", "NDBBD"),
  n_subintervals = 1L,
  steps = NULL,
  n_tasks = 1L,
  n_threads = 1L,
  precision = c("single", "double"),
  verbosity = 1L,
  suppress_outputs = FALSE,
  terminal_run = FALSE,
  accuracy_vars = NULL,
  ...
)
```

Arguments

`cmf_path`	Character length 1, path to the CMF file generated by `ems_deploy()`.
`solution_method`	The solution method to be used, default is `"Gragg"` with `steps = c(2L, 4L, 6L)`. Choices:

- "Gragg": Gragg's method (the smoothed modified midpoint method; Pearson 1991). The shock is applied over a sequence of linearized steps (a Euler first step, midpoint "leapfrog" steps, and a terminal smoothing pass) and the sequence is repeated for each of the three step counts in steps. The three solutions are then combined by Richardson extrapolation, giving rapid error decay as steps increase plus an accuracy estimate from the agreement between levels.
- "Johansen": The one-step Johansen method is fast however it should only be used as a rough approximation due to handling of nonlinear equations.
- "Euler": The forward Euler method, computed for each of the three step counts in steps and combined by Richardson extrapolation. Converges more slowly than "Gragg" (its truncation error shrinks linearly in the step size, so the extrapolated accuracy gains one order per solution instead of two) but it is the most robust of the multistep methods: each step uses only the current state, so it tolerates near-asymptote levels and severe shocks — including exact -100% shocks — where the midpoint family can become unstable. Any strictly increasing step counts are permitted.
- "Midpoint": The midpoint method, Gragg's method without the terminal smoothing pass (N passes for N steps), combined by Richardson extrapolation over steps like "Gragg". It never takes the exogenous variables past their end point, so it suits shocks of exactly -100%, which Gragg's final pass cannot take.
- "RK2", "Heun", "RK4": Explicit Runge-Kutta methods (the midpoint rule, the explicit trapezoid rule and the classic fourth-order method; Schiffmann 2022). A single solution is computed with the one step count in steps — no Richardson extrapolation — with each step built from 2 ("RK2", "Heun") or 4 ("RK4") staged linear solves. "RK4" reaches a given accuracy in far fewer steps than the extrapolating methods.
- "BoSha32", "DoPri54": Embedded Runge-Kutta pairs (Bogacki-Shampine 3(2), 4 stages; Dormand-Prince 5(4), 7 stages). Alongside the solution they compute a component-by-component estimate of the cumulative error — returned as the error_estimate column by ems_compose() and summarized as a face value in the solver log — and they support adaptive step-size control via adaptive and eps_tolerance. "DoPri54" with adaptive = "yes" is the recommended high-accuracy choice.

matrix_method Character of length 1, matrix solution method (default is "LU"). Choices:

- "LU": Standard LU decomposition, the most robust and potentially slowest for a large model. For use with both static and dynamic models.
- "DBBD": Doubly bordered block diagonal, parallel solution for static models partitioned by their regional set.
- "SBBD": Singly bordered block diagonal, parallel solution for intertemporal models partitioned along the time chain.
- "NDBBD": Nested doubly bordered block diagonal, for intertemporal models whose "SBBD" host copy does not fit the memory available. ems_probe() (experimental) suggests the method and the tasks and threads to run it with for a deployment and a machine, and prints the call to paste (see Details).

`n_subintervals` Integer length 1 (default is `1L`), number of subintervals for the applied shock. More subintervals may alleviate accuracy issues stemming from large shock magnitudes. In a model with active complementarities each subinterval runs its own approximate and accurate run; the Runge-Kutta methods take subintervals only there.

`steps` Integer (default is `NULL`, resolved per method: `c(2L, 4L, 6L)` for the extrapolating methods, `4L` for the Runge-Kutta methods). `"Gragg"`, `"Midpoint"`, `"Johansen"` and `"Euler"` take a vector of length 3: the three counts must increase for `"Gragg"`, `"Midpoint"` and `"Euler"`, and must be all even or all odd for `"Gragg"` and `"Midpoint"` (even counts are recommended). `"Gragg"`, `"Midpoint"` and `"Euler"` also take a single count, a single multi-step run without Richardson extrapolation, or two increasing counts, extrapolated from two solutions; no extrapolation accuracy estimate is then available. `"RK2"`, `"Heun"`, `"RK4"`, `"BoSha32"` and `"DoPri54"` take a single count of 1 or more. A larger number of steps may improve accuracy for some model runs.

`n_tasks` Integer length 1 (default is `1L`), number of tasks to run in parallel. Must be `1L` if `"matrix_method"` == "LU".

`n_threads` Integer length 1 (default `1L`), OpenMP threads per MPI task. `"LU"`, `"SBBD"` and `"NDBBD"` give byte-identical results at any thread count; `"DBBD"` with more than one thread agrees only to rounding (see the solving chapter of the user manual).

`precision` Character length 1, `"single"` (default) or `"double"`: floating-point precision of the solver's coefficient storage. `"single"` halves memory traffic and suffices for ordinary step counts. `"double"` selects the double-precision solver binary shipped in the same image, removing the single-precision update-rounding floor that deep extrapolation ladders and many-step large-shock runs otherwise converge onto (per-pass rounding accumulates with step count, so beyond the floor extra steps stop helping). Roughly doubles coefficient-array memory. The effective precision is recorded in `sol.stats.json` and the model diagnostics solve record.

`verbosity` Integer length 1, `0`, `1` (default) or `2`. Console detail while the solver runs: `0` shows errors, warnings and the accuracy figures; `1` adds phase progress and timings; `2` shows every solver line and adds per-rank and per-block debug detail. The full solver log, including the per-subinterval and updated-data accuracy summaries, the convergence codes and every Zerodivide warning, is always written to `model_diagnostics.txt` in the run directory.

`suppress_outputs` Logical length 1 (default is `FALSE`). When `TRUE` solver outputs are not automatically converted into structured data with `ems_compose()`.

`terminal_run` Logical length 1 (default is `FALSE`). When `TRUE`, the function is exited without running the solver. This allows the user to close any R IDE or other programs prior to running from the terminal. When `TRUE` solver outputs are not automatically converted into structured data with `ems_compose()`.

`accuracy_vars` Character vector of variable names (default `NULL`). When given, the accuracy summary of a three-solution `"Gragg"`, `"Midpoint"` or `"Euler"` run covers just these variables, exogenous ones included, instead of every endogenous result. Under the `"solution"` and `"both"` accuracy criteria (see `ems_option_set()`) the warning then judges only them, so a run can be held to the results it is about.

... Additional named solver arguments; anything else is an error, never a silently ignored flag. Effective values of recorded flags land in `sol.stats.json` regardless of how they were passed. Three groups are accepted:

Expert solver flags, with the solver default applied when a flag is absent:

- `postsim` (logical, default `TRUE`): `FALSE` skips the TAB's `PostSim` sections.
- `inmemory` (logical, default `TRUE` except for `"NDBBD"`): keep value arrays and block factors resident in memory rather than in scratch files. The solver falls back to scratch with a warning when the estimated need exceeds the memory available.
- `fastrefac` (logical, default `FALSE`): persistent-pivot refactorization; under `"DBBD"` it helps only small systems, whose block orderings repeat from step to step.
- `ma48_cntl2` (numeric in (0, 1], default each library's own: MA48 0.1, HSL_MP48 0.01): MA48/HSL_MP48 pivot threshold `CNTL(2)`, at every factorization including the `"DBBD"`/`"NDBBD"` rank probes, so it can also move their block sizes; a calibration knob, not a tuning recommendation.
- `ma48_cntl4` (numeric >= 0, default `1e-4`): MA48 `CNTL(4)` in the MA48/MA51 rank probes that size the `"DBBD"`/`"NDBBD"` blocks; pivots below it count as zero (`0` = MA48's own default, exact zeros only). A calibration knob, not a tuning recommendation.
- `nsbbdblocks` (integer, default `2`): HSL MC66 `nblocks`, the number of row blocks MC66 partitions `"SBBD"` into; used only with `withmc66 = TRUE`.
- `withmc66` (logical, default `FALSE`): MC66 row ordering for SBBD.
- `smllthreads` (integer, default the run's thread count): OpenMP threads for the `"NDBBD"` interface factorization and back-substitution.
- `nowrites` (logical, default `FALSE`): skip the solver-side output-file dumps; coefficient composition then has nothing to read. Distinct from `suppress_outputs`, which only skips the R-side composition.
- `condest` (logical, default `FALSE`): per-solve quality diagnostics on the `"LU"` matrix method: componentwise backward error and scaled condition numbers via HSL MA60/MC71, logged per linear solve with run maxima recorded under `condest` in `sol.stats.json`. Diagnostic only (solutions are unchanged) and informative only for nonzero shocks.
- `jacdump` (logical, default `FALSE`): write the base-point Jacobian of the condensed system over every variable element, before the first step, to `sol.jac` with its row map in `sol.jac.json`; the run then solves as asked.

Runge-Kutta step controls: `adaptive`, `eps_tolerance`, `max_retries` and `retry_adjust`; see `ems_RK()` for their documentation and RK-tuned defaults.

MA48 workspace initial guesses, integer percents of the system nonzeros. When omitted, a previous run of the same deployment warm-starts them from its recorded `la_used` in `sol.stats.json`, else package defaults apply; the solver grows the workspace itself if any guess proves too small.

- `laA`: `"LU"`/`"SBBD"` and the diagonal blocks.
- `laD`: the `"DBBD"`/`"NDBBD"` interface systems.
- `laDi`: `"NDBBD"` intermediate interfaces.

Value

A tibble of model output variables and coefficients. The CMF path, invisibly, if `suppress_outputs = TRUE`; pass it to `ems_compose()`. Instructions for terminal execution if `terminal_run = TRUE`.

Solving

The model is written in linearized (percentage-change) form, and the shock is applied as a path from the base data, which is the initial solution, to the post-shock equilibrium (Pearson 1991). Each step along the path solves one or more linear systems in the model's Jacobian (a Gragg run of N steps makes N + 1 passes, a `"DoPri54"` step 7 stages) and then updates the data. `"Johansen"` takes that path in a single step, so its error grows with the size of the shock. The multistep methods divide the path into smaller steps, so the linearization error shrinks as the number of steps rises, and the extrapolating methods then remove most of the error that remains. Computing time grows with the total number of steps, and with `n_subintervals`, since each subinterval repeats the full solve on its share of the shock. A run can be reproduced from its record in `sol.stats.json`.

Accuracy

The extrapolating methods solve at three step counts, and the agreement between the three solutions gives an accuracy summary over the endogenous results, and another over the values read from data files and updated. By default the measure is the share of nonzero results whose levels values are accurate to at least four significant figures (results judged zero, both extrapolations below 1e-6, are counted apart). A percent-change result usually has more accurate figures in its levels value than in the percentage itself. The `accuracy_criterion` and `accuracy_figures` options switch to the updated data or both summaries and change the figures required. A warning is given when the share falls below the `accuracy_threshold` option (default 80%, see `ems_option_set()`), and both summaries are written to `model_diagnostics.txt` in the run directory. When this happens, raise `n_subintervals` first: a few steps over several subintervals beat many steps over one, and a midpoint instability is cured by restarting the integrator, not by more steps (GEMPACK manual 26.1; Schiffmann 2022). More steps (for example `steps = c(4L, 6L, 8L)`) help while the totals stay modest. Adaptive `"DoPri54"` estimates the error of every result (see `ems_RK()`). With one or two step counts there is no estimate. If adding steps stops improving accuracy, the run has reached the rounding limit of single precision, and `precision = "double"` is the remedy.

Solution method

The default `"Gragg"` with `c(2L, 4L, 6L)` suits most static policy shocks. `"Johansen"` is a quick check that a new model and closure run, but its results should not be reported. Shocks that drive a CES or CET share close to zero, or a variable close to `-100%`, can make the midpoint methods unstable (Schiffmann 2022). In those cases use `"Euler"`, which may still converge slowly, or adaptive `"DoPri54"` or `"BoSha32"`, whose robustness exceeds Euler's (Schiffmann 2022); a shock of exactly `-100%` needs `"Midpoint"` or `"Euler"`. A single step count, without extrapolation, suits models whose updates must follow the steps themselves.

Size, matrix method and resources

The linear system has one row for each endogenous variable element left after condensation. Its size grows with the number of regions times the number of commodities, and for intertemporal models it also grows with the number of periods. `"LU"` factorizes the whole system at once. It works for

every model and is usually best for static models, with `n_threads` set to the available cores. The bordered methods divide the system into blocks that are solved in parallel and joined through a smaller border system.

With a bordered method, `n_tasks` is the number of processes that share the blocks, and `n_threads` is the number of threads each process runs. `n_tasks * n_threads` should not exceed the available cores. `ems_probe()` measures a deployed model and the machine, then recommends `matrix_method`, `n_tasks` and `n_threads`. Treat its recommendation as a starting point.

References

Pearson, K.R. (1991), "Solving Nonlinear Economic Models Accurately via a Linear Representation", Impact Project Preliminary Working Paper No. IP-55, Monash University (revised June 2002).

Horridge, J.M., Jerie, M., Mustakinov, D. and Schiffmann, F. (2024), GEMPACK manual (documents Release 12.2), GEMPACK Software, Centre of Policy Studies, Victoria University, Melbourne, ISBN 978-1-921654-34-3. https://ideas.repec.org/p/cop/wpaper/gpman.html

Schiffmann, F. (2022), "Runge Kutta integrators for fast and accurate solutions in GEMPACK", GEMPACK Software, Centre of Policy Studies, Victoria University, Melbourne (14 April).

Van Ha, P. and Kompas, T. (2016), "Solving intertemporal CGE models in parallel using a singly bordered block diagonal ordering technique", Economic Modelling, vol. 52, pp. 3-12. doi:10.1016/j.econmod.2015.07.011

Van Ha, P., Kompas, T., Nguyen, H.T.M. and Long, C.H. (2017), "Building a better trade model to determine local effects: A regional and intertemporal GTAP model", Economic Modelling, vol. 67, pp. 102-113. doi:10.1016/j.econmod.2016.10.015

Kompas, T. and Van Ha, P. (2019), "The 'curse of dimensionality' resolved: The effects of climate change and trade barriers in large dimensional modelling", Economic Modelling, vol. 80, pp. 103-110. doi:10.1016/j.econmod.2018.08.011

HSL (2026), A collection of Fortran codes for large scale scientific computation. https://www.hsl.rl.ac.uk/

See Also

`ems_deploy()` for generating `"cmf_path"`. `solve_in_situ()` for calling the solver on existing input files. `ems_compose()` for structuring data when `"suppress_outputs"` or `"terminal_run"` is `TRUE`. `ems_option_set()` for the severities of TAB `Assertion` failures and declared-range violations (`assertions`, `range_test_initial`, `range_test_updated`), and for the refinement step of `"DBBD"` solves (`refine`).

Examples

```
## Not run:
# The following examples require the teems solver to be built.
# See https://teemsphere.github.io/ to get started.

# Solving a static model with Gragg 2-4-6 (the default):
ems_solve(cmf_path)

# A quick one-step approximation:
```

```
ems_solve(cmf_path, solution_method = "Johansen")

# Solving a dynamic model with the SBBD method:
ems_solve(cmf_path,
          solution_method = "Gragg",
          matrix_method = "SBBD",
          n_tasks = 6)

## End(Not run)
```

`ems_swap` *Prepare closure swaps*

Description

Prepares a closure swap for input into the `swap_in` or `swap_out` arguments of `ems_deploy()`.

Usage

```
ems_swap(var, ...)
```

Arguments

`var`	Character of length 1, model variable to swap.
`...`	Optional named arguments restricting the swap to a subset of variable elements. Each name must use the model-specific set-index format (set name concatenated with its index letter, e.g., `REGr`, `COMMc`). Unspecified sets default to all elements. `SET` corresponds to the generic set name while `i` is specific to the variable index. Three forms are accepted: • Single element: `SETi = "element"` • Multiple elements: `SETi = c("element1", "element2")` • Subset: `SETi = "SUBSET"`

Details

`ems_swap()` return values have no purpose used in isolation. The closure provided to the `closure_file` argument of `ems_model()` will be used if no swaps are specified. Note that full variable swaps can be directly inputted as a character string in the `"swap_in"` and `"swap_out"` arguments of `ems_deploy()`.

Swaps "in" (joining the list of exogenous variables) and swaps "out" (leaving it) are reconciled together rather than in a fixed order: each tuple ends exogenous or endogenous by the net of the swaps that name it, so a variable may be swapped in whole and a slice of it swapped back out, or a swap reversed by its mirror. Swaps are rejected only when no order of them is valid on the closure, such as a tuple swapped in twice or a endogenous variable designated for a swap out.

A closure must leave as many endogenous variable elements as there are equations, so in a working closure, every element swapped in needs one swapped out; `ems_deploy()` aborts when a closure is invalid.

Value

A list of class "full" or "partial" and "single" or "multi" containing the swap specification. Intended for use as swap_in or swap_out in ems_deploy().

See Also

ems_model() for parsing and modifying model and closure files. ems_deploy() for loading the output of this function.

Examples

```
# Full variable swaps
ems_swap("tfd") # out
ems_swap("qfd") # in

# Partial variable swaps with uniform "PROD_COMM" set
# application (note distinction between "REGr" and "REGs") in
# the classic GTAP model (version 6.2)
ems_swap("tfd", TRAD_COMMi = "food", REGr = "chn") # out
ems_swap("qfd", TRAD_COMMi = "food", REGs = "chn") # in

# Partial variable, multiple element swaps with uniform "ACTS"
# set application in the standard GTAP model (version 7.0).
ems_swap("tfd", COMMc = c("food", "crops"), REGr = "usa") # out
ems_swap("qfd", COMMc = c("food", "crops"), REGr = "usa") # in

# Valid subsets may also be selected.
ems_swap("txs", COMMc = "NMRG", ALLTIMEt = "FWDTIME") # out
ems_swap("qxs", COMMc = "NMRG", ALLTIMEt = "FWDTIME") # in
```

---

ems_uniform_shock *Prepare a uniform shock*

Description

Prepares a uniform shock to be implemented as a single value uniformly across all or a subset of variable tuples, in the variable's own form: a percentage change for a percent-change variable, an ordinary change for a change variable.

Usage

```
ems_uniform_shock(var, value, ...)
```

Arguments

var	Character of length 1, the variable to be shocked.
value	Numeric length 1, value of the shock in the variable's own form (percentage or ordinary change).

`...` Optional named arguments restricting the shock to a subset of the variable's elements. Each name must use the model-specific set-index format (set name concatenated with its index letter, e.g., `REGr`, `COMMc`). Value may consist of one or more elements or a subset. In an intertemporal model, `Year = <year>` may stand in for the time set. When omitted, the shock is applied to all variable tuples.

Value

A `list` object with shock configuration to be passed to the `shock` argument of `ems_deploy()`.

See Also

`ems_deploy()` for loading the output of this function. `ems_swap()` for changing the standard model closure.

Examples

```
# Full uniform: 2% shock imposed on all afeall tuples
ems_uniform_shock(var = "afeall", value = 2)

# Partial uniform by element: applied only to the "chn"
# element in set REGr (REG). Note that set designations must
# consist of the concatenation of the standard set (e.g., REG)
# and variable-specific index (e.g., r).
ems_uniform_shock(var = "afeall",
                  REGr = "chn",
                  value = 2)

# Partially uniform by subset and element: applied to "chn"
# and "usa" across the MARG and FWDTIME subsets.
ems_uniform_shock(var = "qxs",
                  COMMc = "MARG",
                  REGs = c("chn", "usa"),
                  ALLTIMEt = "FWDTIME",
                  value = -1)
```

`GTAP_convert` *Load GTAP HAR files and convert data formats*

Description

`GTAP_convert()` reads a GTAP database's HAR files into lists of arrays at full resolution. With `target` it can also convert a standard GTAP database between the classic v6.2 and standard v7 data formats, or prepare a GTAP-AEZ, GTAP-E or GTAP-Power database for the respective model. The output can be inspected or modified in R, passed directly to `ems_data()` via its `dat_input`, `par_input`, and `set_input` arguments, or written to disk for later use.

Usage

```
GTAP_convert(dat_har, par_har, set_har, target = NULL)
```

Arguments

dat_har	Character vector of length 1, file name in working directory or path to a GTAP HAR file with basedata coefficient data.
par_har	Character vector of length 1, file name in working directory or path to a GTAP HAR file with parameter coefficient data.
set_har	Character vector of length 1, file name in working directory or path to a GTAP HAR file with set elements and attributes.
target	Character vector of length 1 (default is NULL). One of: • NULL: the files are read as they are, with no conversion or preparation. • "GTAPv6" or "GTAPv7": converts a standard GTAP database to the v6.2 or v7 data format. Version conversion applies to the standard GTAP database only. • "GTAP-AEZ", "GTAP-E" or "GTAP-EP": prepares a v7-format GTAP-AEZ, GTAP-E (12a) or GTAP-Power (12a) database for the corresponding model, adding the sets and parameters the model reads that the database does not ship. These targets do not convert between data formats: a v6-format database (the GTAP 10a layers) is refused. ems_data() applies the same preparation on its own when it detects one of these databases.

Value

A named list with elements dat, par, and set, each containing a list of arrays corresponding to database headers. Suitable for direct use with ems_data().

See Also

ems_data() for loading and preparing converted data for a model run.

Examples

```
## Not run:
# The following examples require input data. See
# https://teemsphere.github.io/ to get started.

# Convert HAR files to lists of arrays
ls_arrays <- GTAP_convert(
  dat_har = "path/to/gsdfdat.har",
  par_har = "path/to/gsdfpar.har",
  set_har = "path/to/gsdfset.har"
)

# Convert v7.0 files to v6.2 format
converted2v6 <- GTAP_convert(
  dat_har = "v7_data/gsdfdat.har",
  par_har = "v7_data/gsdfpar.har",
```

```
  set_har = "v7_data/gsdfset.har",
  target = "GTAPv6"
)

# Convert v6.2 files to v7.0 format
converted2v7 <- GTAP_convert(
  dat_har = "v6_data/gsddat.har",
  par_har = "v6_data/gsdpar.har",
  set_har = "v6_data/gsdset.har",
  target = "GTAPv7"
)

## End(Not run)
```

`plot.teems_probe` *Plot a structural probe*

Description

Visualizations of a `ems_probe()` result:

- `"incidence"`: the equation-system structure — a spy-style matrix of equation statements (rows, declaration order) by referenced variables (columns, first-appearance order), shaded by element-level incidence weight.
- `"dm"`: the coarse Dulmage-Mendelsohn localization of a structurally singular system — the under-, well- and over-determined blocks to scale (only available when the probe found defects).
- `"components"`: the simultaneous components of the system, the groups of equations that must be solved together: their sizes, and the equation statements of the largest (requires `components = TRUE`).

Usage

```
## S3 method for class 'teems_probe'
plot(x, type = c("incidence", "dm", "components"), max_labels = 60L, ...)
```

Arguments

`x`	A `teems_probe` object from `ems_probe()`.
`type`	Character length 1: `"incidence"` (default), `"dm"`, or `"components"`.
`max_labels`	Integer length 1 (default `60L`). Axis labels are dropped when a dimension exceeds this count.
`...`	Unused.

Value

The input, invisibly. Base-graphics side effects; the underlying tibbles (`x$incidence`, `x$components`, pattern blocks) are exposed for custom plotting.

`solve_in_situ` *Configure and solve model in-situ*

Description

Calls the teems-solver Docker image directly with user-supplied input files, bypassing the `ems_data()` / `ems_model()` / `ems_deploy()` pipeline. All input files must be provided in their final form.

Usage

```
solve_in_situ(
  ...,
  model_dir,
  model_file,
  closure_file,
  shock_file,
  ignore_condense = FALSE,
 solution_method = c("Gragg", "Johansen", "Euler", "Midpoint", "RK2", "Heun", "RK4",
    "BoSha32", "DoPri54"),
  matrix_method = c("LU", "DBBD", "SBBD", "NDBBD"),
  n_subintervals = 1L,
  steps = NULL,
  n_tasks = 1L,
  n_threads = 1L,
  precision = c("single", "double"),
  verbosity = 1L,
  suppress_outputs = FALSE,
  terminal_run = FALSE,
  accuracy_vars = NULL,
  solver_args = NULL
)
```

Arguments

`...`	Named arguments corresponding to input files necessary for an in-situ model run. Names must correspond to "File" statements within the model Tablo file. Values correspond to file paths where these files are found. No checks or modifications are conducted on input files used in this manner. All model declared input files as well as `"model_file"`, `"closure_file"`, and `"shock_file"` are required for in-situ model runs.
`model_dir`	Character of length 1, base directory where input files will be copied (if not all already present) and model outputs will be written.
`model_file`	Character vector length 1, path to a TABLO file with .tab extension. See teems-models for vetted models and compatible Tablo file formatting.
`closure_file`	Character vector length 1, path to a closure file with .cls extension. See teems-models for closure file formatting.

shock_file
: A character vector length 1, path to a file with .shf extension representing a fully prepared shock file. No checks or modifications are carried out on this file.

ignore_condense
: Logical length 1 (default is FALSE). If TRUE, Substitute and Backsolve statements found in the model file are ignored, as for ems_model() (in-TAB Omit statements are always ignored). Set this to match the ems_model() call that produced the closure and shock files.

solution_method
: The solution method to be used, default is "Gragg" with steps = c(2L, 4L, 6L). Choices:

  - "Gragg": Gragg's method (the smoothed modified midpoint method; Pearson 1991). The shock is applied over a sequence of linearized steps (a Euler first step, midpoint "leapfrog" steps, and a terminal smoothing pass) and the sequence is repeated for each of the three step counts in steps. The three solutions are then combined by Richardson extrapolation, giving rapid error decay as steps increase plus an accuracy estimate from the agreement between levels.
  - "Johansen": The one-step Johansen method is fast however it should only be used as a rough approximation due to handling of nonlinear equations.
  - "Euler": The forward Euler method, computed for each of the three step counts in steps and combined by Richardson extrapolation. Converges more slowly than "Gragg" (its truncation error shrinks linearly in the step size, so the extrapolated accuracy gains one order per solution instead of two) but it is the most robust of the multistep methods: each step uses only the current state, so it tolerates near-asymptote levels and severe shocks — including exact -100% shocks — where the midpoint family can become unstable. Any strictly increasing step counts are permitted.
  - "Midpoint": The midpoint method, Gragg's method without the terminal smoothing pass (N passes for N steps), combined by Richardson extrapolation over steps like "Gragg". It never takes the exogenous variables past their end point, so it suits shocks of exactly -100%, which Gragg's final pass cannot take.
  - "RK2", "Heun", "RK4": Explicit Runge-Kutta methods (the midpoint rule, the explicit trapezoid rule and the classic fourth-order method; Schiffmann 2022). A single solution is computed with the one step count in steps — no Richardson extrapolation — with each step built from 2 ("RK2", "Heun") or 4 ("RK4") staged linear solves. "RK4" reaches a given accuracy in far fewer steps than the extrapolating methods.
  - "BoSha32", "DoPri54": Embedded Runge-Kutta pairs (Bogacki-Shampine 3(2), 4 stages; Dormand-Prince 5(4), 7 stages). Alongside the solution they compute a component-by-component estimate of the cumulative error — returned as the error_estimate column by ems_compose() and summarized as a face value in the solver log — and they support adaptive step-size control via adaptive and eps_tolerance. "DoPri54" with adaptive = "yes" is the recommended high-accuracy choice.

matrix_method
: Character of length 1, matrix solution method (default is "LU"). Choices:

- "LU": Standard LU decomposition, the most robust and potentially slowest for a large model. For use with both static and dynamic models.
- "DBBD": Doubly bordered block diagonal, parallel solution for static models partitioned by their regional set.
- "SBBD": Singly bordered block diagonal, parallel solution for intertemporal models partitioned along the time chain.
- "NDBBD": Nested doubly bordered block diagonal, for intertemporal models whose "SBBD" host copy does not fit the memory available. ems_probe() (experimental) suggests the method and the tasks and threads to run it with for a deployment and a machine, and prints the call to paste (see Details).

n_subintervals Integer length 1 (default is 1L), number of subintervals for the applied shock. More subintervals may alleviate accuracy issues stemming from large shock magnitudes. In a model with active complementarities each subinterval runs its own approximate and accurate run; the Runge-Kutta methods take subintervals only there.

steps Integer (default is NULL, resolved per method: c(2L, 4L, 6L) for the extrapolating methods, 4L for the Runge-Kutta methods). "Gragg", "Midpoint", "Johansen" and "Euler" take a vector of length 3: the three counts must increase for "Gragg", "Midpoint" and "Euler", and must be all even or all odd for "Gragg" and "Midpoint" (even counts are recommended). "Gragg", "Midpoint" and "Euler" also take a single count, a single multi-step run without Richardson extrapolation, or two increasing counts, extrapolated from two solutions; no extrapolation accuracy estimate is then available. "RK2", "Heun", "RK4", "BoSha32" and "DoPri54" take a single count of 1 or more. A larger number of steps may improve accuracy for some model runs.

n_tasks Integer length 1 (default is 1L), number of tasks to run in parallel. Must be 1L if "matrix_method" == "LU".

n_threads Integer length 1 (default 1L), OpenMP threads per MPI task. "LU", "SBBD" and "NDBBD" give byte-identical results at any thread count; "DBBD" with more than one thread agrees only to rounding (see the solving chapter of the user manual).

precision Character length 1, "single" (default) or "double": floating-point precision of the solver's coefficient storage. "single" halves memory traffic and suffices for ordinary step counts. "double" selects the double-precision solver binary shipped in the same image, removing the single-precision update-rounding floor that deep extrapolation ladders and many-step large-shock runs otherwise converge onto (per-pass rounding accumulates with step count, so beyond the floor extra steps stop helping). Roughly doubles coefficient-array memory. The effective precision is recorded in sol.stats.json and the model diagnostics solve record.

verbosity Integer length 1, 0, 1 (default) or 2. Console detail while the solver runs: 0 shows errors, warnings and the accuracy figures; 1 adds phase progress and timings; 2 shows every solver line and adds per-rank and per-block debug detail. The full solver log, including the per-subinterval and updated-data accuracy summaries, the convergence codes and every Zerodivide warning, is always written to model_diagnostics.txt in the run directory.

suppress_outputs
: Logical length 1 (default is FALSE). When TRUE solver outputs are not automatically converted into structured data with ems_compose().

terminal_run
: Logical length 1 (default is FALSE). When TRUE, the function is exited without running the solver. This allows the user to close any R IDE or other programs prior to running from the terminal. When TRUE solver outputs are not automatically converted into structured data with ems_compose().

accuracy_vars
: Character vector of variable names (default NULL). When given, the accuracy summary of a three-solution "Gragg", "Midpoint" or "Euler" run covers just these variables, exogenous ones included, instead of every endogenous result. Under the "solution" and "both" accuracy criteria (see ems_option_set()) the warning then judges only them, so a run can be held to the results it is about.

solver_args
: Named list (default NULL). The named solver arguments ems_solve() accepts through ..., passed as a list because the in-situ ... carries the input files: the MA48 workspace initial guesses (laA, laD, laDi), the expert solver flags (postsim, inmemory, fastrefac, nsbbdblocks, withmc66, smllthreads, nowrites, condest, jacdump, ma48_cntl2, ma48_cntl4) and the Runge-Kutta run controls (rk_chart, rk_norm, rk_controller, rk_scope, rk_h0, rk_guard) and the Runge-Kutta step controls (adaptive, eps_tolerance, max_retries, retry_adjust; see ems_RK()). See the ems_solve() ... documentation for each argument's meaning.

Value

A tibble containing model output variables and coefficients. Alternatively, if "suppress_outputs" is TRUE, file path to a CMF file that may be used with ems_compose().

See Also

ems_solve() for the standard package-supported solver.

Examples

```
## Not run:
# The following examples require the teems solver to be built.
# See https://teemsphere.github.io/ to get started.

solve_in_situ(
 GTAPDATA = "path/to/dat_file.txt",
 GTAPPARM = "path/to/par_file.txt",
 GTAPSETS = "path/to/sets_file.txt",
 model_dir = "existing/dir/for/model_run",
 model_file = "path/to/model_file.tab",
 closure_file = "path/to/closure_file.cls",
 shock_file = "path/to/shock_file.shf"
 )

## End(Not run)
```

Index

ems_compose, 2
ems_compose(), *9*, *23*, *26–28*, *30, 31*, *38*, *40*
ems_custom_shock, 4
ems_custom_shock(), *8, 9*, *25*
ems_data, 5
ems_data(), *8, 9*, *13*, *15*, *18*, *25*, *34, 35*, *37*
ems_deploy, 8
ems_deploy(), *2*, *4, 5*, *7*, *13*, *19*, *21, 22*, *25, 26*, *31–34*, *37*
ems_example, 10
ems_example(), *13*
ems_model, 12
ems_model(), *8, 9*, *20*, *32, 33*, *37, 38*
ems_option_get, 14
ems_option_get(), *16*, *19*
ems_option_reset, 16
ems_option_reset(), *15*, *19*
ems_option_set, 16
ems_option_set(), *3*, *15, 16*, *23*, *28*, *30, 31*, *40*
ems_probe, 19
ems_probe(), *22*, *27*, *31*, *36*, *39*
ems_RK, 21
ems_RK(), *29, 30*, *40*
ems_scenario_shock, 25
ems_scenario_shock(), *8, 9*
ems_solve, 26
ems_solve(), *3*, *8, 9*, *15*, *18*, *20, 21*, *24*, *40*
ems_swap, 32
ems_swap(), *4*, *8, 9*, *34*
ems_uniform_shock, 33
ems_uniform_shock(), *8, 9*

GTAP_convert, 34
GTAP_convert(), *5*, *7*, *10, 11*

plot.teems_probe, 36
plot.teems_probe(), *20, 21*

solve_in_situ, 37
solve_in_situ(), *31*