Infrastructure for Model Operations

incomplete

Tutorial

This assumes that you have:

1. installed recent IRIS (for me IRIS 20150119 works well), and added path to the IRIS and have run `irisstartup` in the Matlab command prompt
2. have a sample directory `thai_fpas` somewhere on the computer; as always I do not recommend putting it in a directory contain space in its name, unless you know how to operate it
3. have a directory which `@cmop` and `@pathincr` classes, these should be either on the Matlab path or in the current directory
4. have opened Matlab, current directory can be anything

This tutorial will take you through a sample session. This session can be put to some driver M-file so that parts of it can be run repeatedly if needed, however, the commands are so simple that we just assume that you issue them in the Matlab command prompt.

The first step is to create an instance of `@cmop` class so we are able to operate files in the `thai_fpas` directory.

```
c=cmop('./thai_fpas','th201505','th_');
```

This creates an object which provides a user interface to the `thai_fpas` directory and its files (Matlab functions, data, definitions of reports, etc.) Details on the structure on this directory, what files you need to provide, which functions to implement, etc are given below.

The second argument is a string identifying the current round. It goes to the filenames of the output files as well as to some reports, etc. You want to change it everytime you work on another forecasting round. You will also make a distinct copy of the `thai_fpas` directory as the files in the directory will change from round to round (at least data files with the new data). The third argument is a prefix to input files in the directory. It can be empty, but must be always the same as the input files are assumed to be prefixed with this string.

Now we are ready to use the `cmop` class logic through the object `c`. The next step is to parse and solve the model:

```
c.readmodel();
```

This spits out the following output:

```
run:   ./thai_fpas/codes/th_round_options.m
run:   ./thai_fpas/model/th_set_param.m
read:  ./thai_fpas/model/th_model.model
```

```
read:  ./thai_fpas/model/th_model_trnd.model
write: ./thai_fpas/reports/th201505_model_report.pdf
write: ./thai_fpas/results/th201505_model.mat
```

You can see that this method calls `th_round_options` function to get some common options. This file must be provided by you and you will be changing this file from round to round as it contains round specific parametrization. Then it runs function `th_set_param` to get parameters of the model, and reads the model files. Then it parses the model, imposes the parameters, and produces model report PDF and also MAT file with the parsed and solve IRIS model.

In order to analyze model's dynamic properties, one needs to call `analyzemodel` method:

```
c.analyzemodel();
```

This results in the following output:

```
run:   ./thai_fpas/codes/th_round_options.m
read:  ./thai_fpas/results/th201505_model.mat
write: ./thai_fpas/results/th201505_analyze_model.mat
write: ./thai_fpas/reports/th201505_analyze_model.pdf
```

Again, it is pretty simple. It gets the round options, then it reads the parsed and solved model created by the previous method. Then it runs IRFs and conditional variance decompositions and dumps the results to the MAT file, and produces PDF report.

The data come in two forms. In the hard observed data in the observed databases of three frequencies (monthly, quarterly and yearly). And also in the form of expert judgement, softer observations, future assumptions — generally called tunes. Every model requires a specific conversions of the observed data (seasonal adjustments, various transformations and conversions to the model frequency). This is done by the following method:

```
c.observeddata()
```

This results in the following output:

```
run:   ./thai_fpas/codes/th_round_options.m
read:  ./thai_fpas/data/th_monthly.csv
read:  ./thai_fpas/data/th_quarterly.csv
read:  ./thai_fpas/data/th_yearly.csv
run:   ./thai_fpas/codes/th_observed_data.m
read:  ./thai_fpas/results/th201505_observed_data.csv
```

The method gets the round options, and reads the hard observable database (monthly, quarterly and yearly). Then it runs `th_observed_data` which makes the user defined conversions and the output is stored in the CSV database.

When the data is ready, we can filter history to obtain the estimates of the initial conditions for the forecast. This is done by method:

```
c.filterhistory('baseline');
```

The method takes one argument being the string designating a scenario. In one round there can be multiple scenarios. This scenario is called `baseline`. It imposes some judgements over the history, as well as judgements and assumptions over the forecasting horizon. The method produces the following output:

```
run:   ./thai_fpas/codes/th_round_options.m
read:  ./thai_fpas/results/th201505_model.mat
read:  ./thai_fpas/results/th201505_observed_data.csv
read:  ./thai_fpas/tunes/th_baseline_tunes.csv
write: ./thai_fpas/results/th201505_baseline_history_mean.csv
write: ./thai_fpas/results/th201505_baseline_history_std.csv
run:   ./thai_fpas/codes/th_report_history.m
write: ./thai_fpas/reports/th201505_baseline_historical_interpretation.pdf
```

The method reads the round options, reads the model created by `readmodel` method, reads the processed data created by `observeddata` method, then it loads the tunes for the `baseline` scenario. It uses a Kalman filter to filter the data with the model, however, it is possible to use a user defined filtering techniques, this is controlled by the round options. The result of the filter is saved in the CSV databases for the mean and stdandard errors. Then the user defined function is invoked to produce the historical report, which is saved in the PDF file.

Because in the round we have also an alternative scenario called `alt1`, we run the same method with a different parameter. The alternative scenario, from the historical point of view, imposes a different judgement on the size of the recent output gap. The call and the result looks as follows:

```
c.filterhistory('alt1');
run:   ./thai_fpas/codes/th_round_options.m
read:  ./thai_fpas/results/th201505_model.mat
read:  ./thai_fpas/results/th201505_observed_data.csv
read:  ./thai_fpas/tunes/th_alt1_tunes.csv
write: ./thai_fpas/results/th201505_alt1_history_mean.csv
write: ./thai_fpas/results/th201505_alt1_history_std.csv
run:   ./thai_fpas/codes/th_report_history.m
write: ./thai_fpas/reports/th201505_alt1_historical_interpretation.pdf
```

The forecast is run by the following command taking one parameters designating the scenario.

```
c.forecast('baseline');
```

The result is the following

```
run:   ./thai_fpas/codes/th_round_options.m
read:  ./thai_fpas/results/th201505_model.mat
read:  ./thai_fpas/tunes/th_baseline_tunes.csv
read:  ./thai_fpas/results/th201505_baseline_history_mean.csv
run:   ./thai_fpas/codes/th_define_scenario_baseline.m
write: ./thai_fpas/results/th201505_baseline_forecast.csv
write: ./thai_fpas/results/th201505_baseline_forecast.mat
```

```
run:   ./thai_fpas/codes/th_define_forecast_report.m
write: ./thai_fpas/reports/th201505_baseline_forecast.pdf
```

So it gets the round options, then it reads the model, the `baseline` scenario tunes, and the mean estimates of the historical interpretation. It reads the definition of the scenario (that is which shocks explain which variables, and which shocks are surprises or are anticipated). It runs the forecast imposing the given tunes, and saves the forecast in the CSV and MAT files. Then it reads a definition of the forecast report, and save the forecast report in the PDF file.

The alternative scenario `alt1` can be solved in the same way. However, if we provide a second argument the `baseline` scenario, the method will also create the report comparing the alternative to the baseline, and also decomposing the difference between these two scenarios into various factors like change in the initial conditions, change in surprise/anticipated information, change in shock values, and also non-linear effects if the model is non-linear. The method is called in this way:

```
c.forecast('alt1','baseline');
```

And this results in:

```
run:   ./thai_fpas/codes/th_round_options.m
read:  ./thai_fpas/results/th201505_model.mat
read:  ./thai_fpas/tunes/th_alt1_tunes.csv
read:  ./thai_fpas/results/th201505_alt1_history_mean.csv
run:   ./thai_fpas/codes/th_define_scenario_alt1.m
write: ./thai_fpas/results/th201505_alt1_forecast.csv
write: ./thai_fpas/results/th201505_alt1_forecast.mat
run:   ./thai_fpas/codes/th_define_forecast_report.m
read:  ./thai_fpas/results/th201505_baseline_forecast.csv
write: ./thai_fpas/reports/th201505_alt1_vrs_th201505_baseline_forecast.pdf

run:   ./thai_fpas/codes/th_round_options.m
read:  ./thai_fpas/results/th201505_model.mat
read:  ./thai_fpas/results/th201505_model.mat
read:  ./thai_fpas/results/th201505_alt1_forecast.mat
read:  ./thai_fpas/results/th201505_baseline_forecast.mat
  impact of model change
  impact of surprise/anticipated change
  impact of initial condition change
  impact of shocks change
    impact of shock eps_ygap change
    impact of shock eps_pi change
    impact of shock eps_s change
    impact of shock eps_istar change
    impact of shock eps_ystargap change
    impact of shock eps_pienergystar change
    impact of shock eps_pistar change
    impact of shock eps_rstarbar change
    impact of shock eps_piexpstar change
  impact of model non-linearity
write:
```

```
./thai_fpas/results/th201505_alt1_vrs_th201505_baseline_decomp_forecast_chan
ge.mat
write:
./thai_fpas/reports/th201505_alt1_vrs_th201505_baseline_decomp_forecast_chan
ge.pdf
```

Besides the steps performed by the method with one argument, the two argument version calculates decomposition of the differences between two scenarios and saves the decomposition in the MAT file and produces PDF report.

Now the historical simulations. The frequency of the simulations can be different than the frequency of the model:

```
c.histforecast(mm(2003,1):1:mm(2015,3));
```

Resulting in:

```
run:   ./thai_prod/codes/th_round_options.m
read:  ./thai_prod/results/th201505_model.mat
run:   ./thai_prod/codes/th_historical_databases.m
read:  ./thai_prod/data/th_quarterly.csv
write: ./thai_prod/results/th201505_historical_forecasts.mat
write: ./thai_prod/reports/th201505_historical_forecasts.pdf
```

CMOP class and methods

Constructor

Construct the @cmop object

Synopsis:

```
c = cmop(dirname, name, prefix);
```

where `c` is a constructed object of the `cmop` class, `dirname` is a path (either relative or absolute) to the directory root structure of the files described in the section below, and `prefix` is a prefix used for the input files in the directory structure.

Method readmodel

Read and parse model files, get the model parameters, impose them on the model, solve the reduced form, generate reports and save the resulting model in the MAT file.

Synopsis:

```
c.readmodel();
```

where c is an object of the @cmop class.

The method parses model files in the model directory, which have suffix .model. The method creates either one version of the model, or multiple versions of the model, one for history and others for the scenarios. Whether there is one model version or multiple model versions is decided by the option model_linear.

If only one version of the model is needed, then model_linear is a single value, either true or false. Then option model_switches is a sinlge struct assigning values of the model switches. In this case, the model is used for all methods (historical interpretation, model analysis, forecasts for all scenarios, …).

```
  opt.round_scenarios = {'baseline', 'alt1'};
  opt.model_linear = true;
  opt.model_switches = struct('taylorrule',true,...
                              'lossfunction',false,...
                              'nonlinzgap',false,...
                              'gdpcomp',true);
```

If one needs to use multiple versions of the model, the model_linear must be a struct with the given fields. One of the fields must be history, assigning the linear flag for the model version to be used for historical interpretation, model analysis, and historical forecasts. Other fields must correspond to the scenarios declared by the option round_scenarios. In this case, model_switches option must be struct of structs. The fields of the structs also correspond to the history and the scenarios. Each struct then sets the model switches for the the particular model version. The following chunk of the round_options.m function sets up multiple models:

```
  opt.round_scenarios = {'baseline', 'alt1'};
  opt.model_linear   = struct('history',true,'baseline',false,'alt1',false);
  opt.model_switches = struct('history',
struct('linear_cpixfe',true,'endocred',false),...

'baseline',struct('linear_cpixfe',true,'endocred',true),...

'alt1',struct('linear_cpixfe',true,'endocred',true));
```

Note also, that the model switches are passed to set_param.m function. So that means if one needs to use a different model parameters for the different model versions, then it must be done within the set_param function.

Method analyzemodel

Generate impulse response functions, conditional variance decompositions, and report them including the reduced form for the chosen variables.

Synopsis:

```
c.analyzemodel();
```

where c is an object of the @cmop class.

Method observeddata

Prepare observed data for the model in the model frequency. It basically reads the three files in the `data` directory and calls a user defined function which performs all the desired transformations (seasonal adjustments, frequency conversions, ..).

The output database should be still in the user units (for example percent growth rates, percent inflations, non-compounded interest rates, etc) and these variables will be automatically transformed to the internal units of the model according to the mnemonics system (growth rates and inflations in log differences, interest rates continuously compounded).

Synopsis:

```
c.observeddata();
```

where c is an object of the @cmop class.

Method filterhistory

Method which filters the historical data estimating latent variables either by Kalman filter or by a user supplied function.

It reads the model (output of the method `readmodel`), observed data (output of the method `observeddata`), and user tunes. Then it runs the Kalman filter or the user supplied function which estimates latent variables on its own. Then it outputs the databases with the estimates.

Again, the database on the tunes can have variables in external units (growth rates and inflations in percents), then the method automatically transforms the data according to the mnemonics system. After the estimation of the historical variables is finished, the method performs inverse conversions to store the results in the external units.

The range of the filter is given by the round option `history_range`, which can actually overlap the range for the forecast. In this case, the filtered shocks will enter the forecast step and will be used there unless they are overwritten by some endogenous tunes. This is useful when plugging some forecast information into the historical interpretation step can help identify some historical latent variables.

Synopsis:

```
c.filterhistory(scenario);
```

where `scenario` is a string standing for the scenario. The allowed values must be declared in the round options through option (cell array of strings) `round_scenarios`.

Method forecast

Run forecast for the given scenario.

Synopsis:

```
c.forecast(sc);
c.forecast(sc, sc1);
c.forecast(sc, c2, sc2);
c.forecast(sc,'--',option, value, ..);
c.forecast(sc, sc1, '--', option, value, ..);
c.forecast(sc, c2, sc2, '--', option, value, ..);
```

where `c` and `c2` are objects of the `@cmop` class, `sc` and `sc1` strings naming scenarios from the `c` object, and `sc2` scenario from the `c2` object. `option`, and `value` pairs can be given after `'–'` and they override setting in the round options.

The forecast uses the following information:

Information	Source
Model	If the round has a single model, then this model is used, otherwise it is the model for the chosen scenario `sc`.
Initial condition	By default, the initial condition is taken from the result of c.filterhistory(sc), unless it is overwritten by the `use_hist` option. Then the information from the scenario tunes is used, this overrides the information from the historical filter. The remaining possibly missing information in the initial condition is taken from the steady state of the scenario's model.
Surprise/anticipated shocks	The list of anticipated shocks is taken from the result returned by codes/<prefix>define_scenario_<sc>.m in the field `anticipated_shocks`.
Tuned variables and shocks	Taken from tunes/<prefix><sc>_tunes.csv.
Which shocks are used to explain tuned variables	Taken from the result returned by codes/<prefix>define_scenario_<sc>.m in the field `endo_tunes_by_exo`.

The forecast method does the following

1. Runs the forecast using the information given by the table above
2. Calculate confidence bands
   1. If option `stochastic_simulations` is set to zero or the model is linear, then the linear confidence bands are calculated using Kalman filter using the observed data only, not the tunes
   2. If option `stochastic_simulations` is non-zero, then the non-linear simulations are run by latin cube method and the confidence bands are calculated from these draws
3. Creates the forecasting report defined by codes/<prefix>define_forecast_report.m as
   1. If the method has only one parameter, then the forecast report does not make any comparison
   2. If the method has two parameters, then the forecast report compares the forecast with the forecast from the other scenario `sc1`
   3. If the methed has three patameters, then the forecast is compared with the forecast of the

scenario `sc2` of the round `c2`

4. Runs decomposition of the forecast changes (see the method `decompose_forecast_change` below:
    1. If the method has only one parameter, no decomposition is run
    2. If the method has two parameters, the decomposition of the change is taken from the scenario `sc1` of the same round
    3. If the method has three parameters, the decomposition of the change is taken from the scenario `sc2` of the round `c2`

Method histforecast

Method which reports and evaluates the the historical forecasts: plots and RMSEs.

Method decompose_forecast_change

This section documents a static method `decompose_forecast_change` which can be called as shown below, but usually it is called by higher level methods. This section's aim is not to describe its invokation in order to allow users to call it, but to document how the decomposition is done.

```
dc = cmop.decompose_forecast_change(c1, sc1, c2, sc2);
```

The method decomposes difference between scenario given by its name `sc1` in the `@cmop` object `c1` and scenario of the name `sc2` in the object `c2`.

As the two scenarios and the two rounds can have a very different assumptions (models can be different, assumptions on the shocks anticipated/surprise, even shocks can be different), we need to carefully specify how the method works. We need to introduce some notation:

Element	Scenario 1	Scenario 2	Details
Model	M1	M2	Model object (equations, parameter values, standard errors of the shocks, list of variables, list of shocks).
Range	R1	R2	Range of the forecast in the scenario.
Surprise	S1	S2	Surprise/anticipated information. It is a list of anticipated shocks.
Init	I1	I2	Initial condition (i.e. all lagged variables with the data before first date of the range.)
Shocks	D1	D2	Data of the shocks in the model over the range.

Now, we want to decompose the differences between Scenario 1 and Scenario 2, which are given by [M1,R1,S1,I1,D1] and by [M2,E2,S2,I2,D2] resp. However, in order to compare anything, first we have to have the same model. So the first step is to calculate the change of the scenarios caused by the model change, that is from M2 to M1. However, S2, I2, and D2 are incompatible with the M1 model, as it can have different lagged variables, and shocks. First, we have to adjust this information for the new model M1. For that reason we define

Element	Adjusted for M1	Description
Range	R2A=R1	We just do everything for the range used for the Scenario 1.

Element	Adjusted for M1	Description
Surprise	S2A	List of anticipated shocks for the model M1 is such that only those shocks which were anticipated in the model M2 and they exist in the model M2 are anticipated in M1. All new shocks are considered surprise.
Init	I2A	Adjusted initial condition for the model M2 is constructed so that all variables which were in the M2 and are not in M1 are ignored, and all variables in M1 which were not in M2 are initialized to the steady state.
Shocks	D2A	Adjustment of the shocks data takes all shocks existing in model M1 and in model M2 and periods existing in R1 and R2 from D2. All other shocks values (new shocks in the model and new time periods) are taken as zeros.

Change	Difference between	Note
Model	[M1,R1,S2A,I2A,D2A]-[M2,R2,S2,I2,D2]	That means that the model change includes also changes wrt adjustments descibed above
Surprise	[M1,R1,S1,I2A,D2A]-[M1,R1,S2A,I2A,D2A]	
Init	[M1,R1,S1,I1,D2A]-[M1,R1,S1,I2A,D2A]	
All shocks	Sum of [M1,R1,S1,I1,D1A2(i+1)]-[M1,R1,S1,I1,D1A2(i)]	D1A2(i) is a database where shocks 1..i are from D1 and the rest from D2A.
Non-linear		It is everything what is left between [M1,R1,S1,I1,D1] and [M2,R2,S2,I2,D2] unexplained by the sum of all components above.

Method allround

The synopsis of the method is simple:

```
c.allround();
```

where `c` is object of the `@cmop` class.

The method runs all the methods in the correct order besides the `histforecast` method. It calls `observeddata`, `readmodel`, then `filterhistory` only for the scenarios whose initial condition is used in some other forecasts. Then it runs `forecast` method for the first scenario without any comparison, and then `forecast` method for the remaining scenarios comparing and decomposing the forecast change from the first scenario.

Directory structure

Directory structure of input files

The user has to provide a directory with the following structure. The directory is then used as an input argument to the constructr of `cmop` class, which through various methods calls or refers to the files provided by the user.

Directory	Filename	Details
model	<prefix>model.model	Root IRIS model file, other *.model files imported to it
model	<prefix>*.model	Other model files imported into the root file
model	<prefix>set_param.m	Matlab function returning a struct having parameter values, initial values for the steady state of endo variables, standard errors of the shocks. It takes one argument, which is a struct of the model switches from the user options.
codes	<prefix>round_options.m	Matlab function returning a struct of various options for this round. For the list of options see below.
codes	<prefix>observed_data.m	Matlab user defined function which takes the model switches, and monthly, quarterly and yearly databases as inputs and returns one database of the frequency of the model expressing observed information in the model model variables. The function should perform all the transformations including seasonal adjustments, unit transformations, frequency conversions etc.
codes	<prefix>filter_history.m	Matlab user defined function which takes model, model switches, range, observations database, historical tunes, and returns the database of all the model lags up to the historical range. This function is used if round option use_kalman is false.
codes	<prefix>report_history.m	Matlab user defined function which takes: report object, model, model switches, range, observations database, historical tunes, smoothed database, and returns the report object with added graphs and tables. This function is used always.
codes	<prefix>define_scenario_<scenario>.m	Function which defines which shocks explain which variables in the forecast and which shocks come as surprises, or are completely anticipated. See below for the explanation.
codes	<prefix>define_forecast_report.m	Function which returns definition of the forecasting report (graphs and plots). See below for the explanation.
codes	<prefix>historical_databases.m	Function which returns for the given range (non necessarily same frequency of the model) returns a cell array of historical databases as they were observed in the given dates. The databases can contain also historical NTFs, historical forecasts of some variables, etc.
codes	<prefix>convert_other_freq.m	Function which returns the forecast database in other frequency than the base frequency of the model. It takes two arguments. The first one is a struct with model switches, the second one is frequency character of the required output frequency. The function is required to exist only if `report_table_range2` is given the round option and is not empty.

Directory	Filename	Details
data	<prefix>monthly.csv	Observed various data in monthly frequency.
data	<prefix>quarterly.csv	Observed various data in quarterly frequency.
data	<prefix>yearly.csv	Observed various data in yearly frequency.
tunes	<prefix><scenario>_tunes.csv	Database of the tunes to be used for the scenario.

Directory structure of output files

When the object of the `cmop` class is constructed and methods are sucessfully run, then the methods produce the following output files. The files in the results directory also can serve as inputs to the subsequent methods. The files in the reports directory are to be read by humans, and are not refered by the subsequent methods.

Directory	Filename	Produced by method	Details
results	<name>_model.mat	readmodel	MAT-file containing one variable `m` being the IRIS model object. In the sinlge model set-up, this is the model used for all methods. In multple model setup, this is the model used for `filterhistory()`, `analyzemodel()` and `histforecast()` methods.
results	<name>_<scenario>_model.mat	readmodel	In mupltiple model setup, this model is used for `forecast()` and `decompose_forecast_change()` methods for the given scenario.
results	<name>_<scenario>_model.xml	readmodel	XML version of model file used for GUI. In single model setup, _<scenario> is omitted. In multiple model setup, an XML is generated for each model.
results	<name>_analyze_model.mat	analyzemodel	MAT-files containing IRFs and variance decompositions
results	<name>_observed_data.csv	observeddata	CSV file containing all (hard) observations in the frequency and variables of the model.
results	<name>_<scenario>_history_mean.csv	filterhistory	CSV file containing mean estimates of the historical smoother (if Kalman filter is used), or results of the user defined filter.
results	<name>_<scenario>_history_std.csv	filterhistory	CSV file containing std estimates of the historical smoother. Produced only when Kalman filter is used.
results	<name>_<scenario>_history_shock_decomp.mat	filterhistory	MAT file containing decomposition of the historical smoothed estimates to the effect of initial conditions and individual shocks. The file is produced only if the option `hist_decomp_groups` is among the options and is not empty.
results	<name>_<scenario>_forecast.csv	forecast	CSV file with the forecast of the scenario in the model base frequency.

Directory	Filename	Produced by method	Details
results	<name>_<scenario>_forecast_<freq>.csv	forecast	CSV file with the forecast of the scenario in the frequency of `table_report_range2` if given.
results	<name>_<scenario>_forecast.mat	forecast	MAT file with the forecast of the scenario includingin all intermediate steps as the surprise information was evolving (in each step there are input, and output databases, and simulation plans).
results	<n1>_<sc1>_vrs_<n2>_<sc2>_decomp_forecast_change.mat	forecast	MAT file with the decomp structure having all data on the decomposition into various factors of the difference between two scenarios (sc1 and sc2) possibly from two different rounds (n1 and n2).
results	<name>_<scenario>_scenario_spec.xml	forecast	XML file containing the forecast scenario definition to be used by the GUI.
results	<name>_<scenario>_report_spec.xml	forecast	XML file of the forecast report definition to be used by the GUI.
results	<scenario>_historical_forecasts.mat	histforecast	MAT file with the results of the historical forecasts (cell array of input databases, output databases, ranges, forecasting dates…).
reports	<name>_model_report.pdf	readmodel	Report on the model equations and calibration
reports	<name>_<scenario>_model.pdf	readmodel	Report on the model equations and calibration which is used for the scenario.
reports	<name>_analyze_model.pdf	analyzemodel	Report on the model dynamic properties: IRFs and conditional variance decompositions
reports	<name>_<scenario>_historical_interpretation.pdf	filterhistory	Report on the historical interpretation.
reports	<name>_<scenario>_history_shock_decomp.pdf	filterhistory	Report on the decomposition of the historical smoothed estimates to the effect of initial conditions and individual shocks. The file is produced only if the option `hist_decomp_groups` is among the options and is not empty.
reports	<name>_<scenario>_forecast.pdf	forecast	Report on the forecast.
reports	<n1>_<sc1>_vrs_<n2>_<sc2>_forecast.pdf	forecast	Report on the deviation between scenario sc1 of the round n1 and scenario sc2 of the round n2.
reports	<n1>_<sc1>_vrs_<n2>_<sc2>_decomp_forecast_change.pdf	forecast	Report on the decomposition between two forecasts into various factors.
reports	<name>_historical_forecasts.pdf	histforecast	Report on the historical forecasts: plots and RMSEs.

Round options

The options valid for various steps in the round are given by the user defined function `codes/<prefix>round_options.m`, which is a function which returns a Matlab struct of the following fields:

Field	**Content**	**Details**
use_sirius	TODO	TODO
model_switches	Model switches for parsing step of the model reading	This struct is passed to the IRIS model constructor when it is created so that !if, !for IRIS statements can be evaluated.
model_linear	Flag for the model linearity	True if the model is linear (or to be linear), false otherwise.
model_options	TODO	TODO
hist_use	Assignment of the initial conditions to the scenarios.	By default, a forecast of the scenario uses initial condition form the `filterhistory` method for the same scenario. This option overrides the default behavior. It is a struct, whose fields are forecast scenarios, and values are scenarios from which the initial condition should be taken for the scenario in the fieldname.
irfs_periods	Number of periods for IRFs	Integer number.
irfs_shocks	Shocks for which IRFs will be run	Cell array of strings.
vardecomp_periods	Number of periods for conditional variance decomposition.	Integer number.
vardecomp_contribs	Maximum number of contributions in the conditional variance decompositions.	Integer number.
vardecomp_variables	Variables whose conditional variance decomposition will be reported.	Cell array of strings.
round_scenarios	Scenarios used defined by this round	Cell array of strings. First scenario is considered as a baseline. That means that other scenarios are reported in deviations to the first scenario.
use_kalman	Flag whether to use Kalman filter for history or not.	True or false.
history_range	Range for the interpretation of the history (Kalman filter or user provided function).	IRIS range. It can overlap with the forecasting range, if one wants to take forecasting tunes intoaccount when filtering the history.
histdecomp_variables	List of variables displayed in historical decomposition to shocks.	Cell array of variable names.
histdecomp_groups	Definition of groups, that is which factors belong to each group. Factors can be either invididual shocks, or `Initial conditions`. If the `Initial conditions` are not named in any group, they will be part of the `Rest` group, which is always generated automatically. If this cell is empty, no decomposition is run.	Cell array of cell array of factors. The factor corresponding to initial conditions is represented by a string `Initial conditions`.

Field	**Content**	**Details**
histdecomp_grnames	Group names. The length of the cell array must be the same as of `histdecomp_groups`.	Cell array of strings.
histdecomp_range	Range for the historical decomposition.	Iris range.
forecast_range	Range for the forecasts of all the scenarios.	IRIS range. It can overlap with the history_range.
report_table_range	Range for the tables in the forecast report.	IRIS range in the base model frequency.
report_table_range2	Range for the second tables in the forecast report.	IRIS range of a different frequency. If not empty, then the user has to supply <prefix>convert_other_freq.m function.
report_plot_range	Range for the plots in the forecast report.	IRIS range.
report_table_col1width	TeX dimension of the width for the first column in the forecast report tables.	String. Fore example '34mm'. You will need to accommodate this to the widht of the comments of you variables in the model.
decomp_contribs	Number of components shown in the plots in the forecast change decomposition.	Number. Includes the REST.
decomp_variables	List of variables for which the decomposition of forecast changes is reported.	Cell array of strings.
histfcast_periods	Length of historical forecasts.	Number.
histfcast_vars	List of variables for which the historical forecasts including RMSEs are reported.	Cell array of strings.
histfcast_show_variable	For each variable, one can optionally display another variable in the same graph.	Struct mapping some of the variable names in `histfcast_vars` to other variable names.
histfcast_plot_hist	Number of historical periods to be drawn back to history for each historical forecast.	Number.
histfcast_range_left	Number of periods to be displayed in the historical forecast graphs before the beginning of the first historical forecast.	Number.
histfcast_nahead	Horizons for which the RMSEs are reported.	Vector of integers. Zero means the last historical period.
rs_maturities	Maturities of interest rates in the model taking into account day count convention for the underlying instruments..	Struct having fields the names of variables representing the interest rates, and values representing the fractions (or multiples of years) of the maturity of the instrument. This information is used for automatic transformations bewteen market quoting of the interest rates (non-coumpounded having the operator NC_ in the mnemonics system) and continuous compounding and back.
stochastic_simulations	TODO	TODO

Definition of the forecasting report

A report of the forecast is defined by a function in the `codes` directory having the form `<prefix>define_forecast_report.m`. The definition is used in the context of reporting a single scenario as well as reporting a given scenario compared to a baseline.

Example:

```
function rd = th_define_forecast_report(model_switches)

  rd = cell(1,0);

  p1 = struct();
  p1.title = 'Summary table';
  p1.table = {struct('label','Main variables',...
                     'series',{{'picpi4','pi4','dy4','ygap','i','s'}}),...
              struct('label','External sector',...

'series',{{'pistar4','ystargap','istar','pienergystar'}})};
  rd{end+1} = p1;

end
```

Definition of the forecast tunes

Example:

```
function f = th_define_scenario_baseline()

  f = struct();
  f.endo_tunes_by_exo = struct(...
    'ytrnd',            'eps_dytrnd',...
    'ygap',             'eps_ygap',...
    'mgap',             'eps_mgap',...
    'xgap',             'eps_xgap',...
    'ca',               'eps_ca',...
    'core',             'eps_pi',...
    'pi_transient',     'eps_pi_transient',...
    'pi_1round',        'eps_pi_1round',...
    'pienergy',         'eps_pienergy',...
    'pienergy_1round',  'eps_pienergy_1round',...
    'cpi',              'eps_pifood',...
    'pifood_1round',    'eps_pifood_1round',...
    's',                'eps_s',...
    'i',                'eps_i',...
    'r',                'eps_piexp',...
    'pigap',            'eps_piexp2',...
    'istar',            'eps_istar',...
    'ystargap',         'eps_ystargap',...
    'pienergystar',     'eps_pienergystar',...
    'cpistar',          'eps_pistar',...
    'rstarbar',         'eps_rstarbar',...
    'dzbar',            'eps_dzbar',...
    'zbar',             'eps_zbar',...
    'premium',          'eps_premium',...
    'rstar',            'eps_piexpstar'...
      );

  f.anticipated_shocks = {...
      'eps_istar',...
      'eps_ystargap',...
      'eps_pienergystar',...
      'eps_pistar',...
      'eps_rstarbar',...
      'eps_piexpstar'...
      };

end
```

TODO

1. think of the way how to decide on the scale of y-axis in the impulse response function so that very small or negligble response are visibly zero
2. historical forecast report: add steady state line for variables which are stationary

From:
http://nasa.ogresearch.com/dokuwiki/ -

Permanent link:
http://nasa.ogresearch.com/dokuwiki/doku.php?id=infrastructure_for_model_operations

Last update: **2017/07/06 14:04**