Macroeconomic Model Data Base 3.1 - User Guide

This user guide describes how to install and use the Macroeconomic Model Data Base, version 3.1 (hereafter the Modelbase or MMB). Section 1 deals with the installation and the software requirements. Section 2 introduces the menu of the MMB 3.1 and describes how to run the software and conduct comparison exercises employing the models and options contained in the Modelbase. Section 3 explains the structure of the key files that govern the simulations carried out by the MMB 3.1. Section 4 lays out the structure of the model files. These are usual Dynare files, which have been amended with extra lines and commands to suit the MMB. Also, the structure of json files is explained here. Lastly, section 5 deals with adding a model to the MMB, while section 6 deals with adding a monetary policy rule to the MMB. This guide is platform independent and can be applied to Windows, Linux and macOs operating systems.

1 Installation and software requirements

The MMB 3.1 can be downloaded from `macromodelbase.com/download`. The version for windows is `mmb-electron-win.exe`, for macOS it is `mmb-electron-mac.dmg`, and for Linux directly download the source code from the release on github at `https://github.com/IMFS-MMB/mmb-gui-electron/tags`.[1] The Windows version and the Mac version of the file autoinstall the MMB on your computer and opens the redesigned frontend of the MMB.

The files for carrying out the simulations of the models are written in MATLAB, so either some version of MATLAB or a recent version of its freeware clone, OCTAVE, must be installed on your computer. In the case of Matlab, one also needs the *Optimization Toolbox* as well as the *Statistics Toolbox* in order to be able to run all models in the Modelbase.[2]. For model solution the program utilizes DYNARE, which can be downloaded free of charge from the web.[3] Under Windows, double-clicking on the downloaded DYNARE exe-file opens a set of steps that guide you through the installation. Under macOS, locate the downloaded pkg-file in Finder, and Control-click the icon to select Open from the menu, thus creating an exception for the app to be installed. The installer will then guide you through the installation. To install Linux verison of Dynare please follow the instructions on the Dynare Wiki[4].

Compatibility

REVISE THIS SECTION

We have tested the MMB 3.0 with DYNARE 4.5.6 and 4.5.7. Earlier versions may work but have not been tested.

[1] Linux users will have to build from source using `npm`. Find more info on our github page.

[2] For the time being there are some models, which cannot be simulated with Octave 4.4.0. The list of models contains: EA_Q14.

[3] `http://www.dynare.org`

[4] The DYNARE Wiki install guide for Ubuntu and Debian can be found at `http://www.dynare.org/DynareWiki/InstallOnDebianOrUbuntu`

On Windows, DYNARE 4.5.6 is compatible with OCTAVE 4.4.0, whereas DYNARE 4.5.7 is compatible with OCTAVE 4.4.1. Both Dynare versions are compatible with MATLAB R2007b and later. For macOS, the compatibility between DYNARE and MATLAB is the same. However, at the time of this release, the highest OCTAVE-supported version of DYNARE is 4.5.6 (compatible with OCTAVE 4.4.0 on macOS).

Further steps before running comparisons

When using MATLAB, one has to add the DYNARE path to MATLAB. In order to do so, open MATLAB and choose *Set path* from the *File* menu. Use the option *Add folder* and browse to the directory where you have installed DYNARE. The DYNARE subfolder that has to be added is called *MATLAB*. When using OCTAVE, one adds the DYNARE path by opening the command window of OCTAVE (it should be open automatically) and typing "*addpath directoryofdynare*". If you don't change the installation directory of OCTAVE you should type "*addpath C:\dynare\4.5.x\matlab*", where x completes the version of dynare.

Before running simulations with the MMB, you need to specify whether you want to run the simulations in MATLAB or OCTAVE. In order to do so, click on 'Menu' on the upper-right corner of the MMB and then on 'Settings', as shown in the Figure 1. It opens a window, in which you have the option either to let the program scan for versions of MATLAB and OCTAVE installed on your computer, or to search manually. If the scan finds more than one version of MATLAB or OCTAVE, you can choose from the list of programs in this subwindow by clicking on the small arrows on the right side. Note that the scan only looks in common directories as scanning the whole system would take too long. If the scan does not find the desired installation, click on 'Find manually', select whether you want to add MATLAB or OCTAVE to the list of executables and browse to the folder of the installation. Select the executable file which opens the program and click on 'Open'. In case of MATLAB you should browse to "...*matlab\version\bin\matlab.exe*". Accordingly, for OCTAVE this is "...*octave\version\bin\octave-cli.exe*". If you want to remove an entry from the selection menu, select it and click on 'Remove selected'.

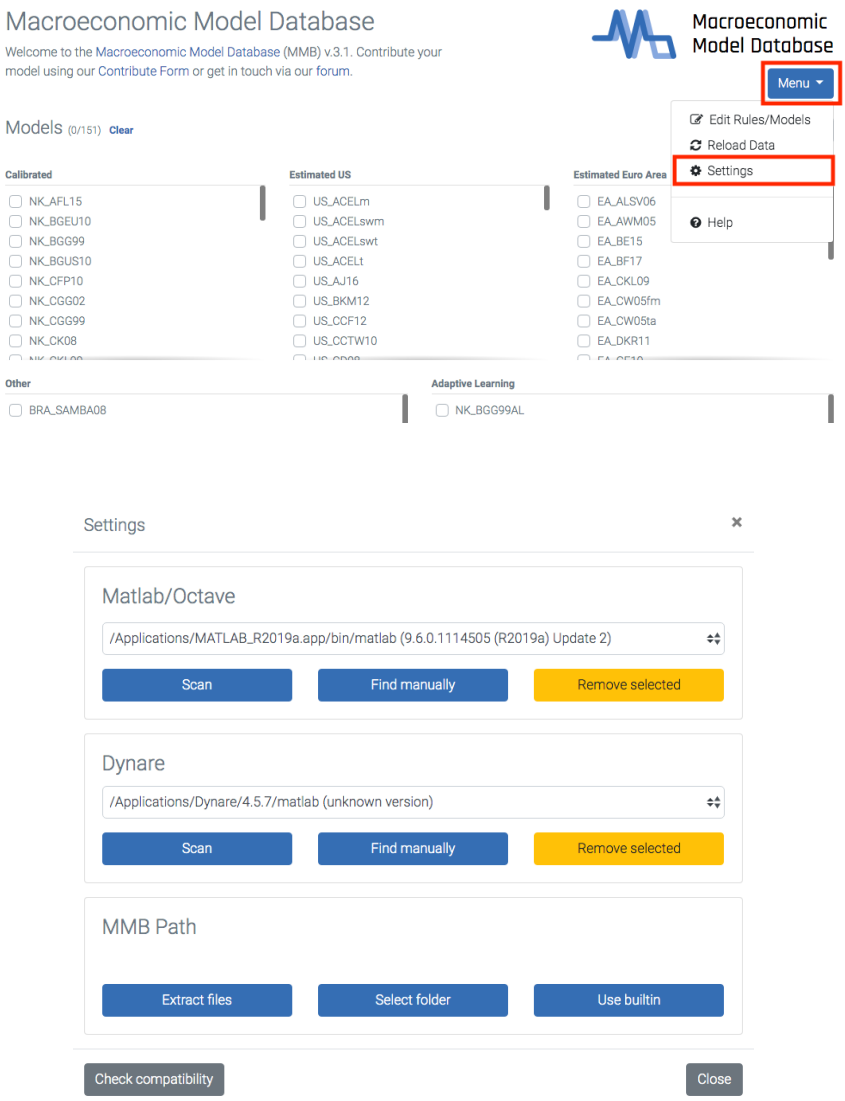
You can follow the same steps to select or delete a version of DYNARE on which you want to run the simulations.

Finally, the 'Settings' menu also allows you to set a path of the MMB. You can either select a folder by clicking directly into the path field and extract the files shipped with the MMB into this folder by clicking on 'Extract files'. You can also click on 'Select folder' to choose a folder in which you already have installed the MMB. This option can be used if you have installed more than on version of the MMB, e.g. an original one and one in which you changed or added models or policy rules. Thirdly, you can click on 'Use builtin' to reset the folder and use the default MMB files shipped with user interface you are using.

To check whether your installations of MATLAB/OCTAVE and DYNARE are compatible with the MMB, click on 'Check compatibility' in the lower left corner. This check can take up to 30 seconds.

You can close the 'Settings' window by clicking on 'Close' in the lower right corner or on the cross in the upper right corner.

Figure 1: SETTINGS



2 The Modelbase: Models, Rules and Options

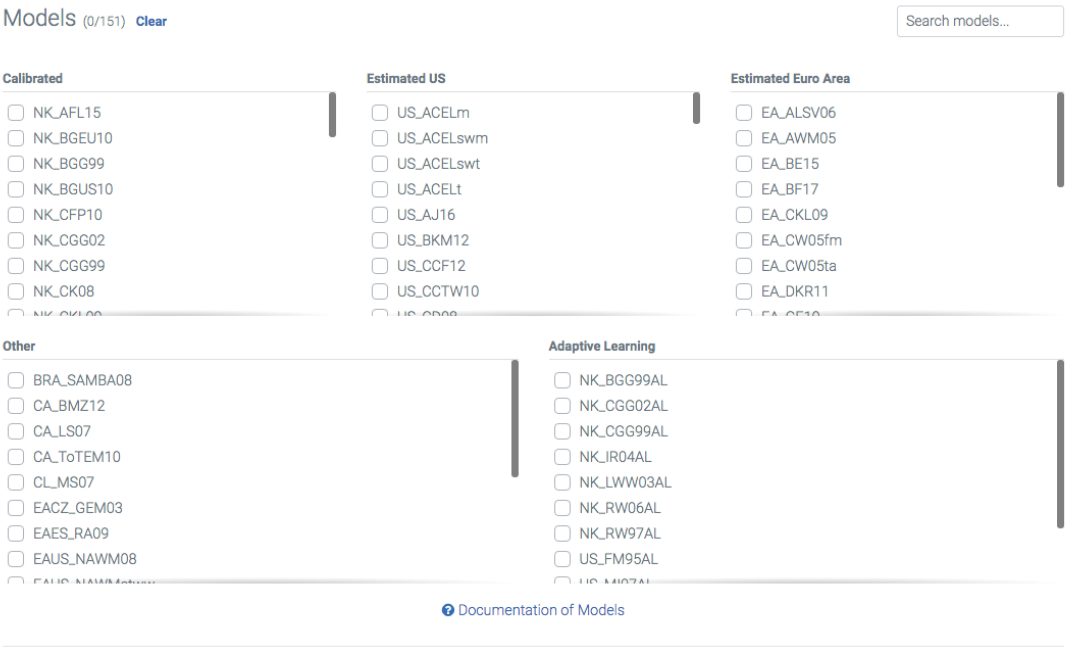
This section introduces the menu of models, policy rules and options, which can be selected in the MMB 3.1.

Models

For the time being there are some models, which cannot be simulated with Octave 4.4.0. The list of models contains: EA_Q14. We will address these issues in the next release.

The user can select from the number of models displayed in the frontend, and from the number of policy rules. In contrast to earlier versions, the MMB 3.1 simultaneously allows for the selection of more than one policy rule and more than one model (the earlier versions either only allowed for one model, many rules or one rule, many models). So, the user can select policy rules, shocks and variables, which are common in all selected models or remove common variables, which are automatically selected.

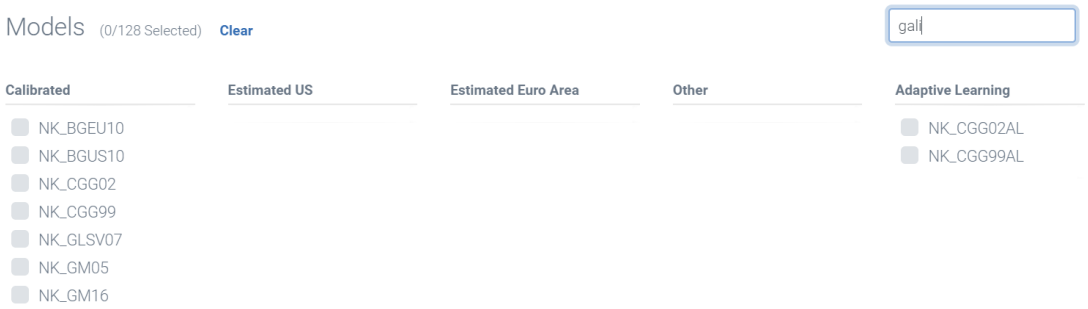
Figure 2: MODELS MENU



The models are sorted in columns, the first of which contains models that have been calibrated to match a closed economy (NK_xxx). The second column lists models that have been estimated on US data (US_xxx). The third column lists models that have been estimated on Euro area data (EA_xxx). The column 'Other' contains models that have been either calibrated or estimated on multi-country data, or have been estimated on other countries, such as 'CA_BMZ12', which has been estimated on Canadian data, or 'EAUS_NAWM08', which has been estimated on Euro area and US data in a two-economy-setting. The last column contains models in which agents form their expectations via adaptive learning (xxxAL).

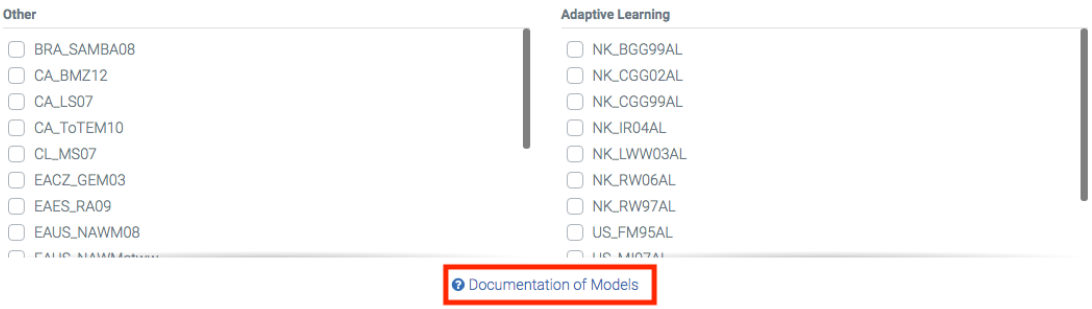
Hovering with the mouse over the models in the list and over the common policy rules displays some basic information such as the title, author and academic reference of the article, in which the model was used. To search for models in the MMB, one can also use the search line on top of the menu. Here one can search for the name of the author or words that appear in the title of the articles, in which the models were used as well as journals or year of publication. For instance, searching for Galí, yields the results shown in Figure 3.

Figure 3: MODELS SEARCH: GALÍ



In addition, the user can download a complete list of all models currently used in the MMB on `macromodelbase.com/downloads`. Lastly, short descriptions of the models and their features are available on the same page. The menu of the MMB contains a direkt link to the model descriptions below the model selection menu as shown in Figure 4.

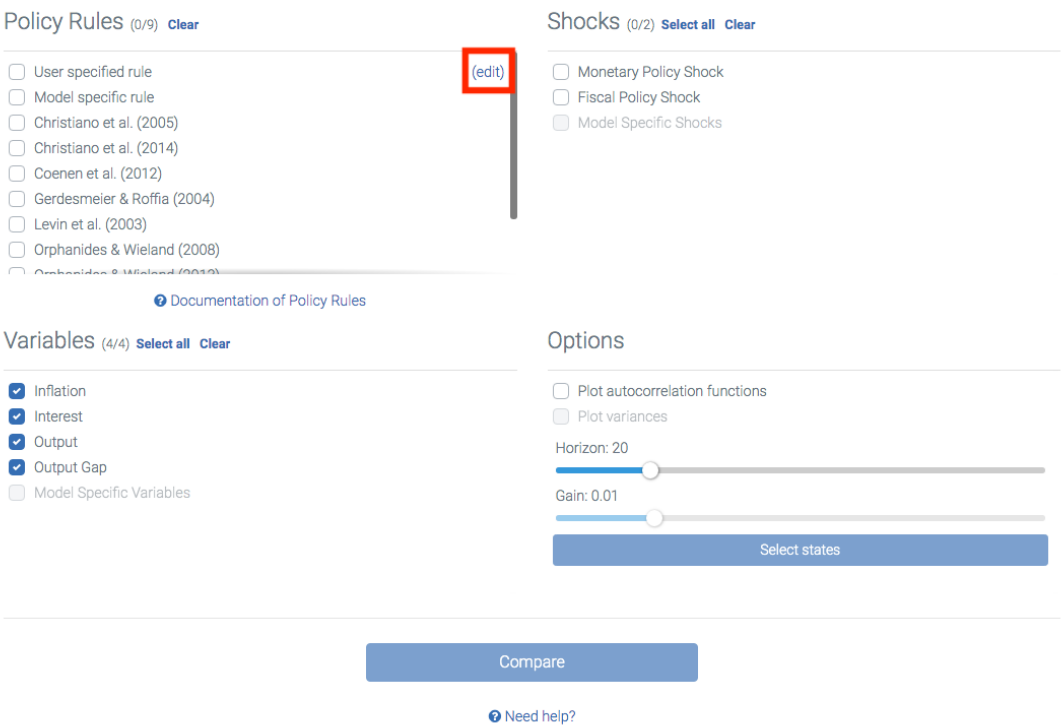
Figure 4: MODEL DOCUMENTATION



Monetary policy rules

Currently, we consider nine monetary policy rules that are taken from Taylor (1993), Levin et al. (2003), Smets and Wouters (2007), among others. Next to these common rules that can (in principle) be used for all models, the selection menu also features the options model specific rule and user-specific rule. The model specific rules are taken from the original articles, in which the models were used. Not all models can be simulated with their model-specific rule, as all interest rate rules that can be used in the modelbase, have to be expressed in terms of the common variables. These common variables are the quarterly output gap, quarterly output, the year-on-year rate of inflation and the policy interest rate in annual terms. As some of the interest rate rules used in the literature feature responses to financial indicators, exchange rates, etc. these rules cannot be included in the modelbase. Therefore only 93 of the 128 models included in the MMB 3.0 feature a model-specific rule.

Figure 5: POLICY RULES MENU



The user specific rule allows the user to directly determine the feedback coefficients in the interest rate rule. In order to do so, the user has to click on '(edit)' next to the entry 'User specific rule' as shown in figure 5. Then, the subwindow shown in figure 6 opens. In this example, the coefficients for current and lagged inflation rates, as well as for the current output gap are chosen such as to mimic the pre-programmed Taylor rule. The time indices refer to quarters. Click on 'OK' to set the policy rule or on 'Cancel' to discard changes.

The user can edit the coefficients of the responses of the policy rates to the leads and lags of the variables in this submenu. However, it is not guaranteed that the Blanchard-Kahn conditions will hold and a determined equilibrium is obtained in the solution of the models with this rule.

Figure 6: MODEL DOCUMENTATION

In general, when a model (or a pre-specified rule) is chosen that is not compatible with a policy rule (or a model) in the way that a simulation of this model-rule-combination does not render the rational expectation equilibrium locally unique, the respective policy rule (or model) is faded out in the menu and cannot be selected for the comparison exercise. Unselecting the model (or rule), makes the rule (or model) again accessible for selection.

Output options

Having chosen the models and a policy rule, the user can make some non-exclusive choices regarding the exercise outcomes to be displayed. The user can decide whether to see the unconditional variances and plot autocorrelation functions of the common variables, both of which are computed using theoretical moments of the solution for each variable. Also the user can opt for plotting impulse response functions of the common variables and specify the horizon for the analysis that is set to twenty periods as a default.

As default the common MMB variables (quarterly output gap, quarterly output, the year-on-year rate of inflation and the policy interest rate in annual terms) are selected. If only one model is chosen, all other variables as defined in the mod-file are also available. As a new feature in the MMB 3.1, some more variables which are featured in many of the models, but not in all of them (among others consumption, investment and capital) can be selected for comparison if available in all selected models.

One can choose impulse responses to a unit monetary policy shock (one percent point increase in the monetary policy shock), and/or to a unit fiscal policy shock (one percent increase in GDP share of government expenditures). Note that all models of the Modelbase have a monetary policy shock, but a

significant number of them do not have a fiscal policy shock. If this is the case, the impulse responses to a fiscal policy shock will not be available. Again, if only one model is selected, all model-specific shocks as defined in the mod-file are available for selection. Alongside the new comparable variables, also a number of shocks appearing in many models are now available for comparison (e.g. Technology shock, Preference shock).

When you select an adaptive learning model, you can also set the gain parameter and select the states by clicking on 'Select states'. In the window that pops up you can select the states individually for all adaptive learning models you have selected for the comparison.

Lastly, for certain models, the unconditional variances are not defined and the autocorrelation functions do not exist. This is the case for some models which feature unit roots. The presence of unit roots prevents a calculation of unconditional moments of some or all variables. Nonetheless, for these models IRFs can still be generated.

Menu

In the upper right corner of the MMB GUI you can find the button 'Menu'. When clicking the button, a drop-down menu opens, where you can click four buttons.
Firstly, 'Edit Rules/Models' opens a sub-window with all models included in the MMB. Here you can manually find the model you want to edit. After clicking on a model name, you can see its two relevant files, namely the .json and .mod file. The .json file contains basic information about the model, like paper title, author and journal, but it also contains information about the model capabilities, i.e. if the model is capable of producing the common variables (output, inflation, interest rate, output gap) and common shock (monetary and fiscal policy shock). As an example, *fiscal_shock": false* means that the model does not contain a fiscal policy shock. You can also see if unconditional variances are possible for this model and if it is an adaptive learning model. Furthermore, under *rules* you can see which policy rule is eligible for the model you want to edit, while under *msr* you can see and change the coefficients for the model specific policy rule. In the block below, you can see the name and the text of each shock and below that of each variable contained in the model. In case you want to change the name of a shock or variable appearing in the main-window, please change its text to the desired text. Hint: The text is case sensitive, so if you want to change the name of a variable for many models, be consistent with the text. The other file appearing in for the selected model is the .mod file. This file consists of a preamble defining all variables, shocks and parameters. It is followed by the parameter initialization and the model equations, which define the sticky price and wage economy and the flex economy, as well as shocks. In the case of non-zero steady state values of variables, there usually also exists a separate *modelname_steadystate.m* file, which defines the steady state of the model. In the upper right corner of the sub-window you can save the changes of the current file. It is important to keep in mind that you need to save the file before switching to another file because then unsaved changes are gone. Also, saving the changes does not apply them. You need to reload the data. At any time you can close the sub-window by clicking the 'Close' button in the lower right corner of the sub-window.

The second button in the drop-down menu is 'Reload Data' and applies the saved changes made to the files, i.e. renaming a shock or variable.

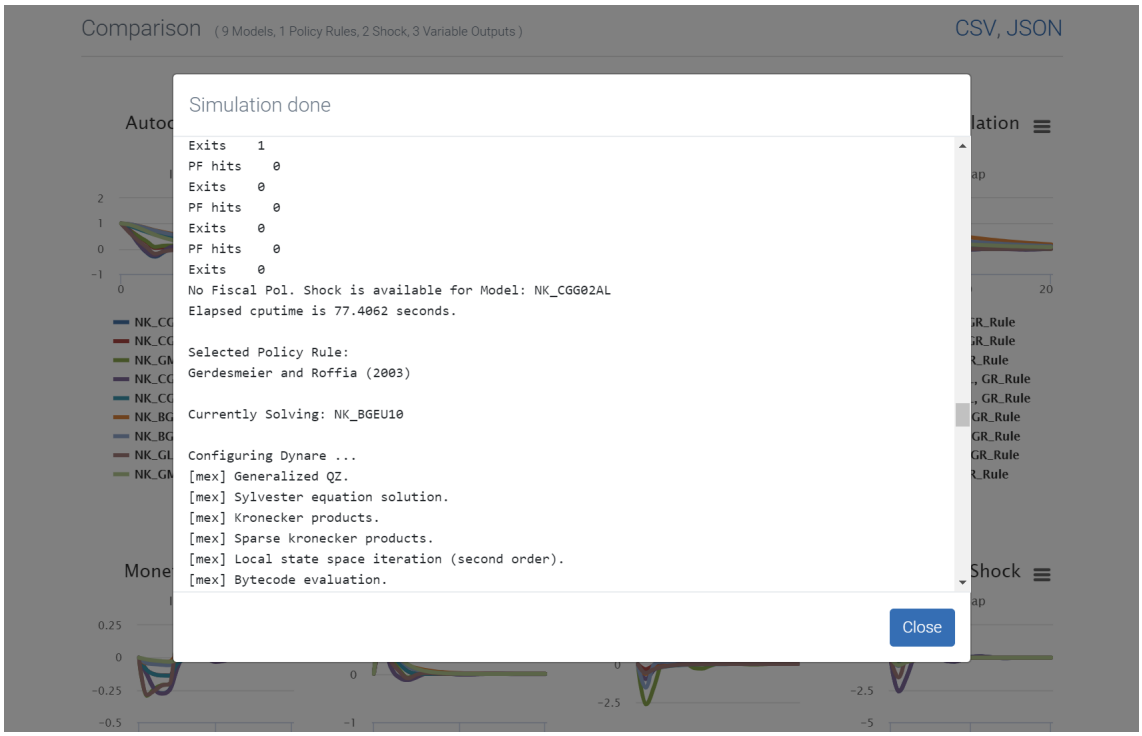
The third button 'Settings' is explained in section 1.

The last button 'Help' opens a help window.

Running a comparison exercise

Once you have selected a number of models, rules and output options, you can simply run the comparison exercise by clicking on the button ‘Compare’. When the simulations run the command window from MATLAB/OCTAVE will be embedded and you see the running output (see, Figure 7). Once the simulations are finished you can close the window by clicking on ‘Close’.

Figure 7: MODEL SIMULATION

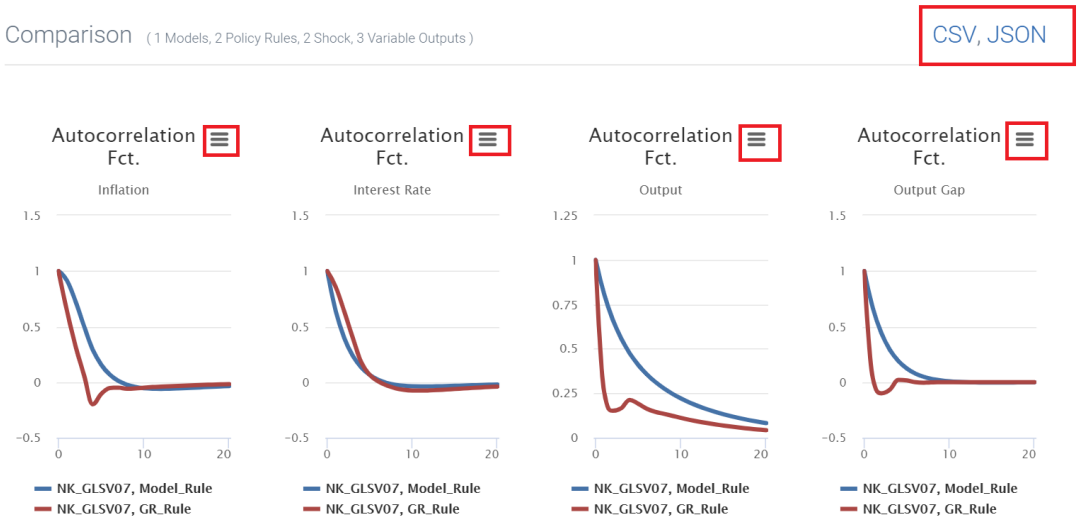


As a result, the selected Impulse response functions and autocorrelation functions are plotted, and the variances are displayed in a table at the bottom of the frontend. There are different options available to display the results (see figure 8). First, the user can choose whether results should be displayed in charts of variables, models or policy rules. This can be selected by clicking on ‘Group Data’ and then the respective option. The default is to group outputs by variable, i.e. the charts represent variables and the graphs in the charts represent model/policy rule-combinations. Further, the user can select the ‘maximum number of columns per row’, which might be useful to increase readability or clarity depending on the screen used. Finally, the user can choose which graphs are displayed in the charts by clicking on the respective entries of the caption.

Figure 8: DISPLAY OPTIONS

The results of the comparison exercise can be exported and saved in two ways. Either, one exports all the data, by clicking on CSV or JSON above the figures to receive a batch export in the respective file format, or one can export the figures one by one, by clicking on the menu on the upper right corner of each figure. Both options are marked with red boxes in Figure 9.

Figure 9: EXPORT OPTIONS



3 Structure of the Modelbase

This section describes the key folders, in which the models, options, and rules are stored and briefly sketches the structure of the key files, which are used in the execution of the modelbase.

Key Folders

Your installation of the MMB 3.1 contains a subdirectory 'resources/app/dist/electron/static/mmcicli', which contains the .m-files and .mod-files for the models, rules and options being used in the comparison exercises. The subfolder *MODELS* contains a specific folder for every model included in the MMB. The specific folders contain a DYNARE mod-file in which the particular model is specified together with related MATLAB files, some of which are created by DYNARE, as well as a json-file, which is needed to link information displayed in the user interface to the corresponding mod-file.

The subfolder *RULES* contains json files for each monetary policy rule.

The subfolder *ALTOOLS* in the folder *LIB* contains scripts for the use of models with adaptive learning.

Some key files

In earlier versions, the MMB had the subfolder *MMB_OPTIONS* containing specific MATLAB files related to the usage of the Modelbase for policy analysis and comparison, as well as explanatory notes for models and policy rules. These folders have been removed. The information is mostly contained in the .json file in the models folder.

The files discussed here are stored in the folder MODELS and are a brief introduction. For further information read section 4.

- **modelname.mod**
  This file contains the model variables, shocks and parameters as well as the model equations including the flex economy, the sticky price and wage economy and shocks. If the model is not linear, then sometimes the file also contains an *initval* block, which provides guess values for non-linear solvers and steady state computations.

- **modelname.json**
  This file contains information about the capabilities of the model. It defines, which of the policy rules, the four common variables and two common shocks are eligible for this model. It also states if unconditional variances can be calculated. Additionally, the file contains general information about the model (e.g. paper, author,...). The model specific policy rule is included if it is available and also contains the exact parameter values. It is also stated if the model includes adaptive learning and if so it shows which variables are forwards, states_short and states_long. For all models, the name and text of all shocks and variables can be seen and changed in this file.

- **optional: modelname_steadystate.m**
  In case of non-zero steady states, this file is included in the model specific folder and contains the steady state equations or values.

4 Structure of the model files

Mod files

The model files are written in the syntax of DYNARE and have a common structure. As an example we take the simple New-Keynesian model by Rotemberg and Woodford (1997) to explain the structure of the mod-files, its model specific parts and the common model data base blocks. The current example is based on the DYNARE 4.5.6 version of the Modelbase. The mod-file is shown in **Figure 10** and **Figure 11**. However, the explanations apply to all models. In the following, the two main parts of a mod-file, the preamble and the model block, are described step by step.

Part 1: The preamble

- Each model file begins with some information about the model. This should include the title, the authors, the publication etc. In front of this description you will find the symbols //, which denote a comment in DYNARE.
- The file then starts with the initialization of the model variables. In our example shown in **Figure 10** the model-specific endogenous variables are listed in line 3 after the keyword *var*: *pi*, *y*, *ynat*, *rnat*, *i*, *x*, *u*, *g* and *g_*. The latter in fact represents an exogenous government spending shock, however it has to be initialized as endogenous variable for reasons that will be explained below. It follows a Modelbase block in lines 5 to 8 in which the common variables are introduced. In general, Modelbase blocks are separated through //******* symbols from the rest of the file.
- Following the keyword *varexo* in line 10 the exogenous variables are initialized. In our example this is *u_*, a cost push shock as well as the common interest rate shock, *interest_* and the common fiscal policy shock, *fiscal_* in line 13. Note that in some models with no treatment of government spending, the latter Modelbase shock may be left out.
- Following the keyword *parameters* in line 16, the Modelbase parameters in the Modelbase block are initialized. In **Figure 10** line 20 we have, for brevity reasons, only included three policy parameters. In the actual mod-files there are many more leads and lags. These are the parameters of the general monetary policy function, except for the last one, *coffispol*, which enters the common discretionary government spending equation.
- Then the model-specific parameters are initialized in line 22.
- Afterwards numerical values are assigned to the model-specific parameters in lines 24 to 33.
- Finally a block called *Specification of Modelbase Parameters* is added. First in lines 38 to 46 the numeric values of the parameters of the selected monetary policy rule are loaded. They are contained in the file *policy_param.mat* in the subfolder *MODELS*. For models in which the original shocks are expressed in percent/100 the parameter *std_r_* has to be reset to 100 after the parameter-loading command. In our example this would have to be done in line 45. However,

Figure 10: STRUCTURE OF THE MODEL FILES: THE PREAMBLE

```
// Model: NK_RW97

var pi y ynat rnat i x u g g_

//***************************************************************************
// Modelbase Variables                                                      //*
   interest inflation inflationq outputgap output fispol;                   //*
//***************************************************************************

varexo u_
//***************************************************************************
// Modelbase Shocks                                                         //*
      interest_ fiscal_;                                                    //*
//***************************************************************************

parameters
//***************************************************************************
// Modelbase Parameters                                                     //*
                                                                            //*
       cofintintbl cofintintb2 ... coffispol                                //*
 /**************************************************************************
 beta sigma alpha theta omega kappa rhou rhog stdinflation_ stdfiscal_;

beta = 1/(1+0.035/4);  // 0.9913
sigma= 6.25;
alpha= 0.66;
theta= 7.66;
omega= 0.47;
kappa= (((1-alpha)*(1-alpha*beta))/alpha)*(((1/sigma)+omega)/(1+omega*theta));
rhou=0;
stdinflation_=0.154;
rhog= 0.8;
stdfiscal_=1.524;

//***************************************************************************
// Specification of Modelbase Parameters                                    //*
                                                                            //*
// Load Modelbase Monetary Policy Parameters                                //*
thispath = pwd;
cd('..');
load policy_param.mat;
for i=1:33
    deep_parameter_name = M_.param_names(i,:);
    eval(['M_.params(i)  = ' deep_parameter_name ' ;'])
end
cd(thispath);

// Definition of Discretionary Fiscal Policy Parameter                      //*
coffispol = 1;                                                              //*
//***************************************************************************
```

Figure 11: STRUCTURE OF THE MODEL FILES: THE MODEL BLOCK

```
52  model(linear);
53
54  //*****************************************************************************
55  // Definition of Modelbase Variables in Terms of Original Model Variables //*
56
57  interest   = i*4;                                                          //*
58  inflation = (1/4)*(4*pi+4*pi(-1)+4*pi(-2)+4*pi(-3));                       //*
59  inflationq  = pi*4;                                                        //*
60  outputgap  = x;                                                            //*
61  output = y;                                                                //*
62  fispol = g_;                                                               //*
63  //*****************************************************************************
64
65  //*****************************************************************************
66  // Policy Rule                                                             //*
67                                                                             //*
68  // Monetary Policy                                                         //*
69                                                                             //*
70  interest =   cofintintb1*interest(-1)                                      //*
71             + cofintintb2*interest(-2)                                      //*
72              ...
73             + cofintoutpf4*output(+4)                                       //*
74             + std_r_ *interest_;                                            //*
75                                                                             //*
76  // Discretionary Government Spending                                       //*
77                                                                             //*
78  fispol = coffispol*fiscal_;                                                //*
79  //*****************************************************************************
80
81  // Original Model Code:
82
83  pi   =  beta * pi(+1)+ kappa*x+ u;
84  u=rhou*u(-1)+u_;
85  x  =  x(+1) - sigma *( i - pi(+1) - rnat) ;
86  rnat = sigma^(-1)*((g-ynat)- (g(+1)-ynat(+1)));
87  ynat = sigma^(-1)*g /(sigma^(-1)+omega);
88  x = y-ynat;
89  g = rhog*g(-1) + g_;
90  // i=phipi*pi + phix*x;
91  end;
92
93
94  shocks;
95  var fiscal_= 1.524^2;
96  var u_=0.154^2;
97  end;
98
99  //stoch_simul (irf = 0, ar=100, noprint);
```

the shocks in this model are already expressed in percentage terms. Secondly, the discretionary fiscal policy parameter *coffispol* is defined as a function of the model-specific parameters in order to obtain a government spending shock of one percent of GDP. The exact implementation of the common fiscal policy shock will be described below. In our example no adjustment is needed and hence *coffispol* is set equal to one.

Part 2: The model block

- The model block starts in line 52 of **Figure 11** as indicated by the keyword *model* followed by *linear*, which tells DYNARE that the equations are already linearized and thus reduces computing time.
- In the Modelbase block going from lines 54 to 63 the common variables are defined in terms of the original model variables. The variable *interest* denotes the annualized short-term interest rate, *inflation* is annual inflation, *inflationq* represents annualized quarterly inflation, *outputgap* and *output* denote the output gap and output, respectively. The common variable *fispol* represents discretionary fiscal policy. It is set equal to the model-specific government spending shock variable, which in the case of our example is *g_*. Note again, that this model-specific shock has to be initialized as an endogenous variable. This allows us the keep the original model equation for government spending unchanged.
- It follows the common *Policy Rule* block. In lines 70 to 74 the common monetary policy rule is specified. Again for reasons of brevity we have not displayed the complete general policy rule in **Figure 11**. Below in line 78, the common equation for discretionary government spending is specified.
- The original model equations are then specified in lines 83 to 90. Note that the model-specific monetary policy rule is commented out because the common policy rule is introduced. On the contrary, the government spending equation in line 89 has remained unchanged. The model section ends in line 91 with the required keyword *end*.
- Finally the variance covariance matrix is specified in lines 95 and 96 between the keywords *shocks* and *end*. Importantly, the variance of the original model-specific government spending shock has been assigned to the common fiscal policy shock variable *fiscal_*. Hence, the common shock *fiscal_* affects the fiscal policy variable *fispol* through the common discretionary government spending expression in line 78 which is set equal to the model-specific government spending shock *g_* in line 62.
- The *stoch_simul* command in line 99 is commented out. Alternatively one can also delete this command.

Json files

The json files have a common syntax. As take the same example as above to explain the structure of the json files. It can be seen in figure 12.

Every json file starts with a curved bracket. All objects need to be specified in " ". Firstly, it is specified what the json file includes, here it is a model, followed by the name of the model (*NK_RW97*). Here, the first letters determine which type of model it is (e.g. New Keynesian model: NK, Euro Area: EA, United States: US, or others), followed by a '_'. The next letters are set according to the

last names of the authors (here: RW stands for Rotemberg and Woodford). The name is completed by the year of publication (97 stands for 1997, while 03 or 13 stands for 2003/2013). Optionally, AL is added to the model in case the model includes adaptive learning.

The first block you need to specify are the *"capabilities"* (lines 4-22). It defines, which of the four common variables and two common shocks are eligible for this model. If they are included, the value is set to *true*, else it is set to *false*. After that *"rules"* contains all policy rules, which are available for this model. Note that the set of available rules is written in square brackets. All unavailable policy rules appear in gray in the GUI and can not be chosen. In step 3 of section 5 there is a list showing which rule number belongs to which monetary policy rule. In the json file it also states if unconditional variances can be calculated (line 21). So for NK_RW97, all common variables, shocks as well as unconditional variances are available. Rules 3-7 and 9-11 can be chosen.

The next block, *"description"* (lines 23-36), contains general information about the model, like reference, paper title, journal and year published, replicants name, keywords, description, category and authors. Again all sets are written in square brackets. This includes key words and authors. General information is shown when holding the mouse over a model. It is important to specify the category of the model because it specifies in which model block the model appears in the GUI. Type *"Calibrated model"* for calibrated models, *"Estimated euro area model"* for EA models, *"Estimated US model"* for US models, *"Estimated other-country model"* for other countries or *"Adaptive learning model"* for adaptive learning models. Note that these entries are case sensitive.

Figure 12: STRUCTURE OF THE JSON FILES


```
{
  "$schema": "model",
  "name": "NK_RW97",
  "capabilities": {
    "fiscal_shock": true,
    "inflation": true,
    "interest": true,
    "mp_shock": true,
    "output": true,
    "outputgap": true,
    "rules": [
      3,
      4,
      5,
      6,
      7,
      9,
      10,
      11
    ],
    "unconditional_variances": true
  },
  "description": {
    "ac_ref": "Rotemberg & Woodford (1997)",
    "paper_title": "An Optimization-Based Econometric Framework for the Evaluation of Monetary Policy",
    "journal": "NBER Macroeconomics Annual 12, pp. 297-346",
    "replicants_name": "S. Schmidt",
    "pub_date": "1997",
    "keywords": [],
    "description": "",
    "category": "Calibrated model",
    "authors": [
      "Rotemberg, Julio",
      "Woodford, Michael"
    ]
  },
  "msr": null,
  "variabledim": 1,
  "al": false,
  "shocks": [
    {
      "name": "fiscal_",
      "text": "Fiscal Policy Shock"
    },
    {
      "name": "interest_",
      "text": "Monetary Policy Shock"
    },
    {
      "name": "u_",
      "text": "Price markup shock"
    }
  ],
  "al_info": null,
  "variables": [
    {
      "name": "fispol",
      "text": "fispol"
    },
    {
      "name": "g",
      "text": "g"
    },
```


Next, there is a block for the model specific policy rule. The New Keynesian model by Rotemberg and Woodford does not have a model specific policy rule, so this entry is set to *null*. It is the policy rule used in the paper. Consider NK_CGG02AL as an example for msr and adaptive learning. Figure 13 (left) shows the syntax for the msr entry. As it is a set, you need to specify the policy rule coefficients inside of square brackets.

Afterwards, the dimension of the shock is specified using *"variabledim"*. It is set to 1 if the shock is in percent, so the number 1 next to the graph means 1%, and 2 if it is in percent/100. In this case the number 1 next to the graph would mean 0.01%. However, MMB 3.1 takes this into account and standardizes it to %.

Figure 13: STRUCTURE OF THE JSON FILES: MSR (LEFT) AND AL (RIGHT)

```
"msr": [
  "0",
  "0",
  "0",
  "0",
  "1.5",
  "0",
  "0",
  "0",
  "0",
  "0",
  "0",
  "0",
  "0",
  "0",
  "0",
  "0",
  "0",
  "0",
  "0",
  "0",
  "0",
  "0",
  "0",
  "0",
  "0",
  "0",
  "0",
  "0",
  "1",
  "0.25"
],
```

```
"variabledim": 1,
"al": true,
"al_info": {
  "forwards": [
    "inflationq",
    "infstar",
    "ytildestar",
    "ytilde",
    "infl",
    "ybar",
    "ystar",
    "ybarstar",
    "y"
  ],
  "states_long": [
    "astar",
    "a",
    "inflationq",
    "inflationql",
    "inflationql2",
    "infstar",
    "infstarl",
    "infstarl2",
    "rstar",
    "interest"
  ],
  "states_short": [
    "astar",
    "a",
    "inflationq",
    "infstar",
    "rstar",
    "interest"
  ]
},
```

The next block is *"al"*, in which adaptive learning properties are specified. If the model does not include this feature, then the entry is set to *false*. Otherwise, it is set to *true*. Figure 13 (right) shows the structure of the code in the case of adaptive learning. In this case *"al_info"* is added, which contains three features, namely *forwards*, *states_short* and *states_long*. All of these features are a set of variables. Forwards are variables appearing with a lead in the model (i.e. var(+1) in the mod file). State variables, which appear with a lag in the model. In case of Euler equation learning, the agents forecast only immediate future variables. States_short entails the state variables used for this forecast and are in the limited information set of the agents. Following Slobodyan and Wouters (2012) Slobodyan and Wouters (2012), for every forward, the information set is a small subset of the state variables and also includes a constant as well as two lags of the forward. Then a regression of the forward is run on the information set.

Alternatively, long-horizon learning considers agents forecasting economic variables until infinity (suggested by Marcet and Sargent (1989)). States_long entails state variables used for this case.

However, in this version of the MMB we focus on Euler equation learning only, so only forwards and states_short are used.

Lastly, there are blocks for *"shocks"* and *"variables"*, in which all shocks and variables of the model are listed. As the two objects are lists, the user needs to use square brackets. Every entry in the list is in curved brackets and separated by a comma. An entry contains a name and a text. The name is the same as in the mod file, while the text appears in the GUI. Note that the text is case sensitive.

5 Adding models to the Modelbase

Adding a new model to the data base consists of four steps. First, the original model has to be translated into a DYNARE mod-file and the common Modelbase variables have to be defined as functions of the original model variables. Second, the mod-file must be stored under the model name in a folder with exactly the same label. Third, a json-file, which builds the link between the mod file and the GUI, has to be created. It also has to have the same name and be stored in the same folder. The folder with both files has to be stored in the subdirectory 'resources/app/dist/electron/static/mmci-cli/models'. Fourth, the GUI needs to be reloaded to incorporate the new model.

Step 1: Creating the mod-file

- The first task when adding a new model to the Modelbase is to create a DYNARE mod-file. The file should start with a comment section giving some information about the associated reference paper(s) for the model.
- The file must have the usual structure of a DYNARE mod-file. That is, one starts with the initialization of variables, shocks and parameters. Then the equations describing the model follow and finally the variance-covariance structure of the shocks is specified.
- However, each of the sections mentioned before has to be augmented by a Modelbase block. This Modelbase block should be visually separated from the original model sections through a comment line //*******.
- After the initialization of the original model variables, the common block *Modelbase Variables* follows. It consists of the six common variables *interest*, *inflation*, *inflationq*, *outputgap*, *output*

and *fispol*. Those variables will be described below. If output is not specified in the model, then the common variable *output* has to be left out. Furthermore, in some small models, one may have to leave out the *fispol* variable. This common block corresponds to lines 4 to 7 in **Figure 10**

- The common block *Modelbase Shocks* is added after the initialization of the original model shocks as in lines 10 to 13 of **Figure 10**. It consists of a common monetary policy shock, *interest_*, and of a common fiscal policy shock, *fiscal_*.

- The third common block is the *Modelbase Parameters* section. Following the initialization of the original model parameters, the common Modelbase parameters are preset, consisting of the monetary policy rule parameters and the discretionary fiscal policy parameter *coffispol*. For the Dynare 4 version of the Modelbase, one first defines the Modelbase parameters and afterwards the original model-specific parameters.

- It follows the numeric specification of the parameters. This is done first for the model-specific parameters and then separately for the common Modelbase parameters in the block called *Specification of Modelbase Parameters*. First, the parameter values of the selected monetary policy rule are loaded. They are contained in the file *policy_param.mat* in the subfolder *WORK*. For models in which the original shocks are expressed in percent/100, the parameter *std_r_* has to be reset to 100 after the parameter-loading command. This specification is required for the proper calculation of impulse response functions. In our example this would have to be done after line 44. However, the shocks in the example are already expressed in percentage terms. Secondly, the discretionary fiscal policy parameter *coffispol* is defined as a function of the model-specific parameters such that a unit government spending shock has a unit impact on output. In our example no adjustment is needed and hence *coffispol* is set equal to one.

- At the beginning of the model section, a *model-specific* Modelbase block has to be added in order to define the common Modelbase variables in terms of original model variables. This is done in lines 52 to 59 in our example. The variable *interest* is defined as the annualized short-term interest rate set by the policy maker. The variable *inflation* denotes the year-on-year inflation rate in percent and *inflationq* denotes the annualized quarter-to-quarter inflation rate in percent. If for instance the original model variable representing quarterly inflation is not annualized, then *inflationq* would have to be specified as four times the original quarter-to-quarter inflation variable. The common variables *outputgap* and *output* represent the output gap and output, respectively.

- The variable *fispol* specifies the common discretionary fiscal policy variable. For implementation of the discretionary fiscal policy variable, one does not have to change the original model equations. The original shock that should represent the common fiscal policy shock has to be initialized as endogenous variable, i.e. following the command *var* instead of *varexo*. In our example the original government spending shock *g_* is initialized in this way. Furthermore, in the section in which the shock variances are specified, this original shock has to be replaced by the common shock *fiscal_*. The *fispol* variable has to be set equal to the original shock variable. If there does not exist a fiscal policy shock in the original model, *fiscal_* and *fispol* should not be initialized.

- Afterwards the common *Policy Rule* block is added to the mod-file, specifying the general monetary policy rule, as it is done in lines 62 to 72 in **Figure 11**. For the sake of brevity we have

not displayed the complete general policy rule in our example. The original monetary policy rule has to be commented out in the original model code. In case the model contains a fiscal policy shock, common discretionary government spending is also specified in the *Policy Rule* block, expressing *fispol* as a function of the *fiscal_* shock, as in line 75 of **Figure 11**. Hence, the common shock *fiscal_* affects the fiscal policy variable *fispol* through this common discretionary government spending expression and *fispol* is set equal to the model-specific government spending shock *g_* in line 59. The original model equations following this block remain unchanged.

- The variances of the two common shocks are specified together with the variances/covariances of the model-specific shocks. Specifically, the variance of the monetary policy shock *interest_* is set equal to zero and therefore it does not have to show up explicitly. For the fiscal policy shock *fiscal_* one adopts the original covariance specification of the replaced shock if available. Otherwise one sets the variance of the fiscal policy shock equal to zero.

- Finally, one has to delete or out-comment the commands for finding the steady state and solving the model as it is done in line 95 of our example.

- NOTE: if your mod-file draws on other files, e.g. a '*steadystate.m*' file, put these into the same folder.

Step 2: Storing the mod-file

- Next, the file has to be stored as mod-file under the model name. In the example, the *NK_RW97* model is stored as *NK_RW97.mod*. The name of calibrated New Keynesian models should start with *NK*, models of the US economy should start with *US* and models of the Euro area should start with *EA*. The full model name should allow for the identification of the specific model among the other Modelbase models. The file must be stored in a folder that has to be created under exactly the same model name and that is positioned in the subfolder *MODEL*, i.e. the created mod-file would be on the path 'resources/app/dist/electron/static/mmci-cli/models/NK_RW97/NK_RW97.mod'.

Step 3: Creating the json-file.

- The json creates the link between the mod-file and the MMB. The structure is as explained in section 4 and shown in figure 12. The contents of the json have to be in curly brackets ({...}) and blocks are separated by commas.

- First, specify that your json refers to a model by setting `"$schema": "model"`,

- Second, specify the name of your model `"name":` *`name`*, which should be the same you have given to the folder, the mod-file and the json-file.

- Third, the `"capabilities"` block specifies which of the common variables and shocks are available in the model; you can enable them by setting them to `true` or disable by `false`.

- Next, enter the policy rules that can be chosen to be run with this model. You have to enter the rule ids in square brackets ([...]) and separate them by commas. The number coding is as follows:

Rule number	Abbreviation	Reference paper
3	Taylor	Taylor (1993)
4	GR	Gerdesmeier and Roffia (2004)
5	LWW	Levin et al. (2003)
6	SW	Smets and Wouters (2007)
7	CEE	Christiano et al. (2005)
8	OW08	Orphanides and Wieland (2008)
9	OW13	Orphanides and Wieland (2013)
10	Coenen	Coenen et al. (2012)
11	CMR	Christiano et al. (2014)

- Fourth, you can specify whether the variances shown are unconditional. For the moment being this is true for all model in the MMB, so set `"unconditional_variances": true`.
- Fifth, add the `"description"`. `"category"` defines in which category the model falls and hence where it will be displayed in the interface. The option are `Calibrated model`, `Estimated euro area model`, `Estimated US model`, `Estimated other-country model` and `Adaptive learning model`. Note that these entries are case sensitive.
- Sixth, specify whether a model-specific rule is available. If the model-specific rule cannot be expressed in terms of the common variables, set `"msr": null`. If a compatible model-specific rule is available, specify the parameters in square brackets. See section 6 for details.
- Seventh, specify the dimension of the shock by setting `"variabledim"` to `1`, if the shock is expressed in percent or to `2` if the shock is expressed in percent/100.
- Eighth, specify whether the model features adaptive learning. If not, set `"al": false` and `"al_info": null`, if yes set `"al": true` and use the block `"al_info": {...}` to specify the `forwards`, `states_long` and `states_short` (see figure 13 for an example).
- Ninth, specify the shocks available in your model by setting the `"shocks": [...]` block. Shocks need to be in curly brackets ({...}) and be separated by commas. For each shock you need a `"name": "`*`name`*`"` which is the same as in the mod-file and a `"text": "`*`displayed_name`*`"` which will be shown in the MMB interface. This block should at least include the common Monetary Policy Shock.
- Tenth, use the `"variables"` block to specify all variables in your model, i.e. both, model-specific variable as well as common variables. This block follow the same syntax as the `"shocks"` block.
- As mentioned before, save the json-file in the same folder as the mod-file.

Step 4: Initializing the model in the Modelbase interface.

- As the final step, one initiates the model in the Modelbase interface.
- Open the MMB interface and click on 'Menu' in the upper right corner as indicated in 1. Then click on 'Reload Data'. The new model should now appear in the list of models under the specified category.

6 Adding rules to the Modelbase

There are three ways to add a new rule to the Modelbase: 1) add a rule to a list of common monetary policy rules, 2) include the model-specific rule calibrated or estimated by the original model authors and 3) specify a rule using the Modelbase user-specified rule option. Below we discuss each case in detail. Keep in mind that policy rules in the Modelbase have to be reformulated in terms of common Modelbase variables such as the annualized quarterly interest rate, the annualized quarter-to-quarter rate of inflation, the quarterly output and the quarterly output gap. This is explained in detail in Wieland et al. (2012) and in the separate document, called 'MMB_MPrule_description.pdf'.

1) Add a common monetary policy rule

- In the new version of the MMB, policy rules are entirely defined via json-files. Common policy rules are stored in the subfolder 'resources/app/dist/electron/static/mmci-cli/rules'.
- The first task is to choose a suitable name for the new policy rule and to create a json-file with the same name. This json-file is to be stored in a subfolder with the exactly same name in the *RULE* folder.
- The json-file for a rule is in general similar to those for models; the whole entry has to be in curly brackets ({...}) and blocks are separated by commas. Figure 14 shows an exemplary json-file for the Taylor (1993) rule.
- First, you have to indicate that the json-file describes a policy rule by setting `"$schema": "rule"`,
- Second, you have to give the new rule an id-number by specifying `"id":` *`number`*. The number 3 to 11 are already taken by the common policy rules shipped with the MMB.
- Third, give your rule a name by specifying `"name":` *`name`*, which should be the same you have given to the folder and the json-file.
- Fourth, you have to actually specify the parameters of the rule by setting `"coefficients":` [...] to the respective values.
- Fifth, you need to add a description of the rule, very much like you do in the json-files for models as described in section 4. Here, you can set `"replicants_name": null`.
- Safe the json-file and go to the MMB interface, click on 'Menu' and 'Reload Data'. The new rule should now be displayed under 'Policy rules'
- However, in order to use the new rule for comparison exercises, you also need to add its `id` to the `rules` section of all model-jsons for which the rule can/shall be used. This is necessary because not all models can be solved with all parameterizations (see subsection 3 of this chapter) and the new MMB is designed such that only common policy rules can be selected in combination with models in which they actually work. When you have done this, again reload data in the interface. The new rule should now be fully implemented in your MMB.

Figure 14: TAYLOR (1993) RULE AS DEFINED IN JSON-FILE


```
{
  "$schema": "rule",
  "id": 3,
  "name": "Taylor",
  "coefficients": [
    "0",
    "0",
    "0",
    "0",
    "1.5/4",
    "1.5/4",
    "1.5/4",
    "1.5/4",
    "0",
    "0",
    "0",
    "0",
    "0",
    "0.5",
    "0",
    "0",
    "0",
    "0",
    "0",
    "0",
    "0",
    "0",
    "0",
    "0",
    "0",
    "0",
    "0",
    "0",
    "0",
    "0",
    "1",
    "0.25"
  ],
  "description": {
    "ac_ref": "Taylor (1993)",
    "paper_title": "Discretion versus policy rules in practice",
    "journal": "Carnegie-Rochester Conference Series on Public Policy, pp. 195-214",
    "replicants_name": null,
    "pub_date": "1993",
    "keywords": [],
    "description": "",
    "authors": [
      "Taylor, John B."
    ]
  }
}
```

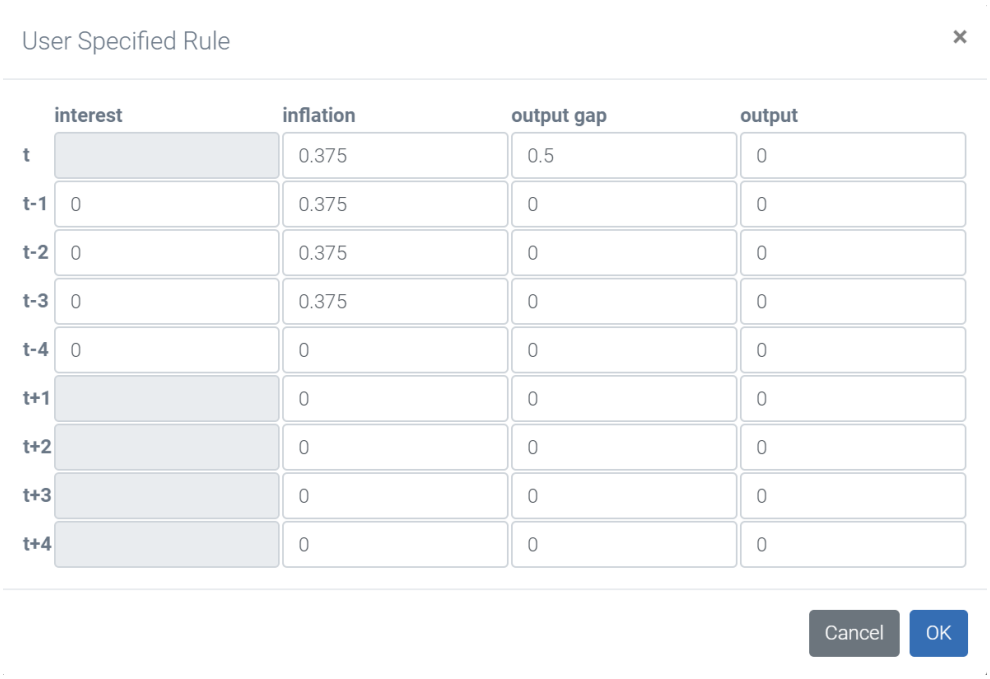
2) Add the model-specific monetary policy rule

- When adding a new model, it is possible to include its policy rule as long as the rule can be rewritten in terms of Modelbase common variables. If this is the case, the user should specify the parameters of the model-specific rule in the `"msr"` section of the model's json-file.
- Reload data in the interface to employ the changes in the MMB.

3) User-specified monetary rule

- When *User-specified rule* is chosen (see figure 5 in section 2), a menu with a general form of a monetary policy rule appears in terms of common variables. Then, one can specify desired coefficient values of each variable in columns and to the corresponding lag/lead in rows. For example, to implement the Taylor (1993) rule using the option for user-specified monetary policy rule, one should set the coefficients as following: $\rho_{\pi,0} = \rho_{\pi,-1} = \rho_{\pi,-2} = \rho_{\pi,-3} = 0.375$, $\rho_{q,0} = 0.5$ and the rest of coefficients to zero. Figure 15 illustrates how to use the option for a user-specified rule with the example of Taylor (1993) rule.

Figure 15: TAYLOR (1993) RULE USING THE OPTION OF USER-SPECIFIED RULE



- Note that with certain rule parametrization, models cannot be solved due to several reasons. For example, the system of equations may violate the Blanchard-Kahn conditions so a model does not yield a unique stationary rational expectations equilibrium. There is no clear guideline for conditions for determinacy, but Levin et al. (2003) suggest several crucial characteristics of rules that deliver a unique equilibrium: a relatively short inflation forecast horizon, a moderate degree of responsiveness to the inflation forecast, an explicit response to the current output gap, and a substantial degree of policy inertia.

References

Christiano, L.J., Eichenbaum, M., Evans, C.L., 2005. Nominal rigidities and the dynamic effects of a shock to monetary policy. Journal of Political Economy 113(1), 1–45.

Christiano, L.J., Motto, R., Rostagno, M., 2014. Risk shocks. American Economic Review 104, 27–65.

Coenen, G., Erceg, C.J., Freedman, C., Furceri, D., Kumhof, M., Lalonde, R., Laxton, D., Linde, J., Mourougane, A., Muir, D., Mursula, S., de Resende, C., Roberts, J., Roeger, W., Snudden, S., Trabandt, M., in't Veld, J., 2012. Effects of fiscal stimulus in structural models. American Economic Journal: Macroeconomics 4, 22–68.

Gerdesmeier, D., Roffia, B., 2004. Empirical estimates of reaction functions for the euro area. Swiss Journal of Economics and Statistics 140(1), 37–66.

Levin, A., Wieland, V., Williams, J.C., 2003. The performance of forecast-based monetary policy rules under model uncertainty. The American Economic Review 93(3), 622–645.

Orphanides, A., Wieland, V., 2008. Economic projections and rules of thumb for monetary policy. Fed of St. Louis Review , 307–324.

Orphanides, A., Wieland, V., 2013. Complexity and monetary policy. Journal of International Central Banking 9(1), 167–204.

Rotemberg, J.J., Woodford, M., 1997. An optimization-based econometric framework for the evaluation of monetary policy. NBER Macroeconomics Annual 12, 297–346.

Slobodyan, S., Wouters, R., 2012. Learning in an estimated medium-scale DSGE model. Journal of Economic Dynamics and Control 36, 26–46. URL: https://ideas.repec.org/a/eee/dyncon/v36y2012i1p26-46.html, doi:10.1016/j.jedc.2011.01.01.

Smets, F., Wouters, R., 2007. Shocks and frictions in US business cycles: A bayesian DSGE approach. The American Economic Review 97(3), 586–606.

Taylor, J.B., 1993. Discretion versus policy rules in practice. Carnegie-Rochester Conference Series on Public Policy 39, 195–214.

Wieland, V., Cwik, T., Mueller, G.J., Schmidt, S., Wolters, M., 2012. A new comparative approach to macroeconomic modeling and policy analysis. Journal of Economic Behavior & Organization 83, 523–541.