Macroeconomic Model Data Base 2.1 - User Guide[1]

This user guide describes how to install and use the Macroeconomic Model Data Base, version 2.1 (hereafter the Modelbase or MMB). After reading Sections 1 and 2 you should be able to run the software and conduct comparison exercises employing the models and options contained in the Modelbase. The next two sections explain the structure of the files used in the Modelbase in more detail. Sections 5 and 6 discuss how one can add new models and new policy rules to the Modelbase, respectively. Finally, Section 7 provides instructions on how to handle the graphical operations necessary to include a new model or a new common rule.

1 Installation and software requirements

The Modelbase is contained in a zip file called *MMB_2_1_Dyn4.zip* which you may save to any place on your computer. In order to use the Modelbase, you have to unpack the zip file to retrieve the folder *MMB_2_1_Dyn4*. This folder contains the file *MMB.m* and a set of subfolders relating to the models included in the Modelbase, to Modelbase options and to output. **Figure 1** illustrates the structure of the Modelbase. The subfolder *Models* contains a specific folder for every model included in the MMB. The specific folders contain a single Dynare mod-file in which the particular model is specified as well as related MATLAB files, some of which are created by DYNARE. The subfolder *MMB_OPTIONS* contains 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. In the folder *OUTPUT* the results from the desired exercises are stored in an Excel file.

The program is written in MATLAB[2], so some version must be installed in your computer.[3] For model solution the program utilizes DYNARE, which can be downloaded free of charge from the web.[4] Double-clicking on the downloaded DYNARE exe-file opens a set of steps that guide you through the installation. After completion, 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*. The Modelbase software is currently compatible with DYNARE 4.2, 4.3 and 4.4.[5]

[1]For further references, please visit www.macromodelbase.com, or contact the Modelbase team.
E-mail: info@macromodelbase.com

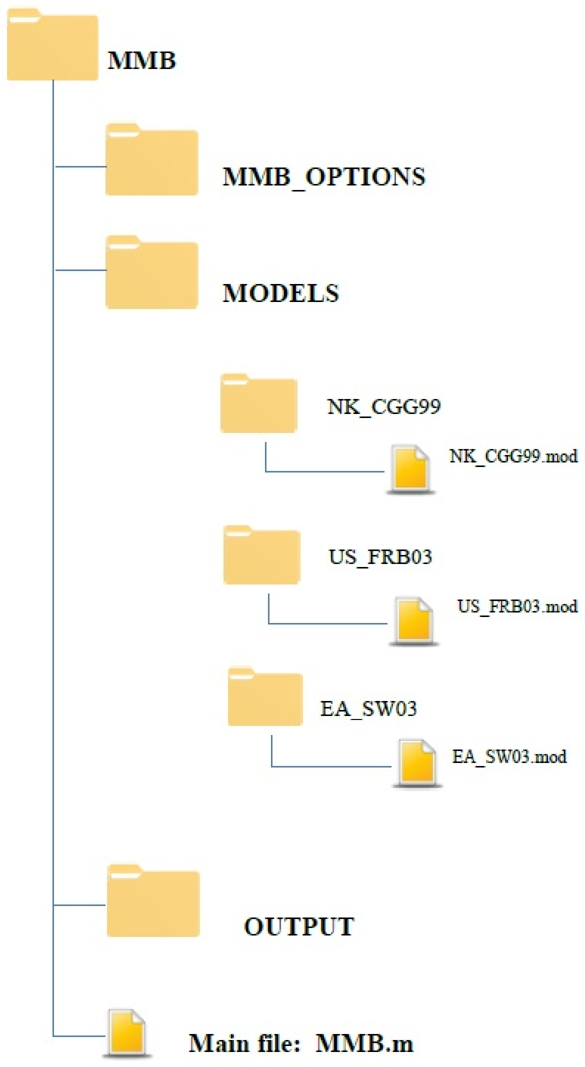
[2]MMB 2.1 is compatible with MATLAB versions 2007b and later.

[3]Solving some models might also require specific MATLAB toolboxes. This is the case, for example, with the US_CMR14 and EA_QR14 models that need the Statistics Toolbox. Also, to save some results, you will need Excel from MS Office installed on your computer

[4]The URL of the DYNARE website is *http://www.dynare.org*.

[5]We have tested that the Modelbase software works well with Dynare 4.4.0, 4.4.1 and 4.4.3, the latest versions until May 2016.

Figure 1: STRUCTURE OF THE MODELBASE



2 Using the Modelbase

MMB.m represents the main file which has to be run when using the Modelbase. In order to run *MMB.m*, you can either open the file in MATLAB and click the *run* button, which automatically adjusts the current directory of MATLAB to the correct path, or you can just start MATLAB and adjust the current directory to the path for the *MMB* folder manually. In the latter case you type afterwards *MMB* into the MATLAB command window and press the *Enter* button. In both cases the Modelbase interface shows up that will guide you through a menu of options from which you can choose. There are four exclusive options. As illustrated in **Figure 2**, those are:

1. One policy rule, many models

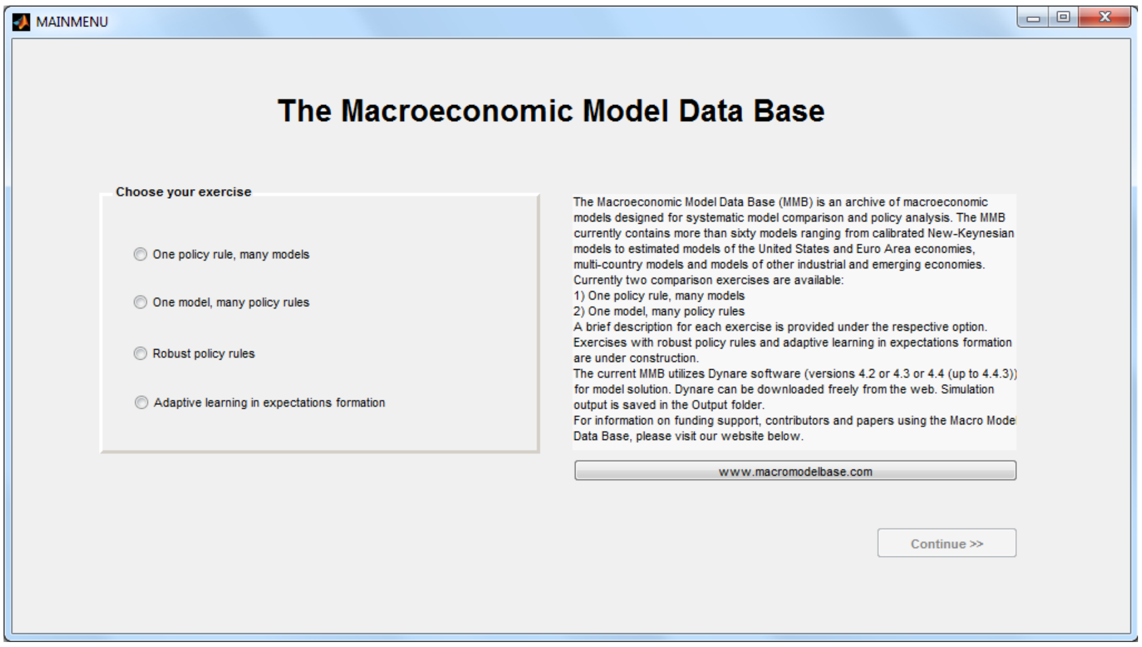
2. One model, many policy rules

3. Robust policy rules

4. Adaptive learning in expectation

In the current version of the Modelbase one can use option 1 and 2. In the following, we describe what one can do under each option and how to use them.

Figure 2: MODELBASE MAIN MENU: OPTIONS


MAINMENU
The Macroeconomic Model Data Base
Choose your exercise
One policy rule, many models
One model, many policy rules
Robust policy rules
Adaptive learning in expectations formation
www.macromodelbase.com
Continue >>

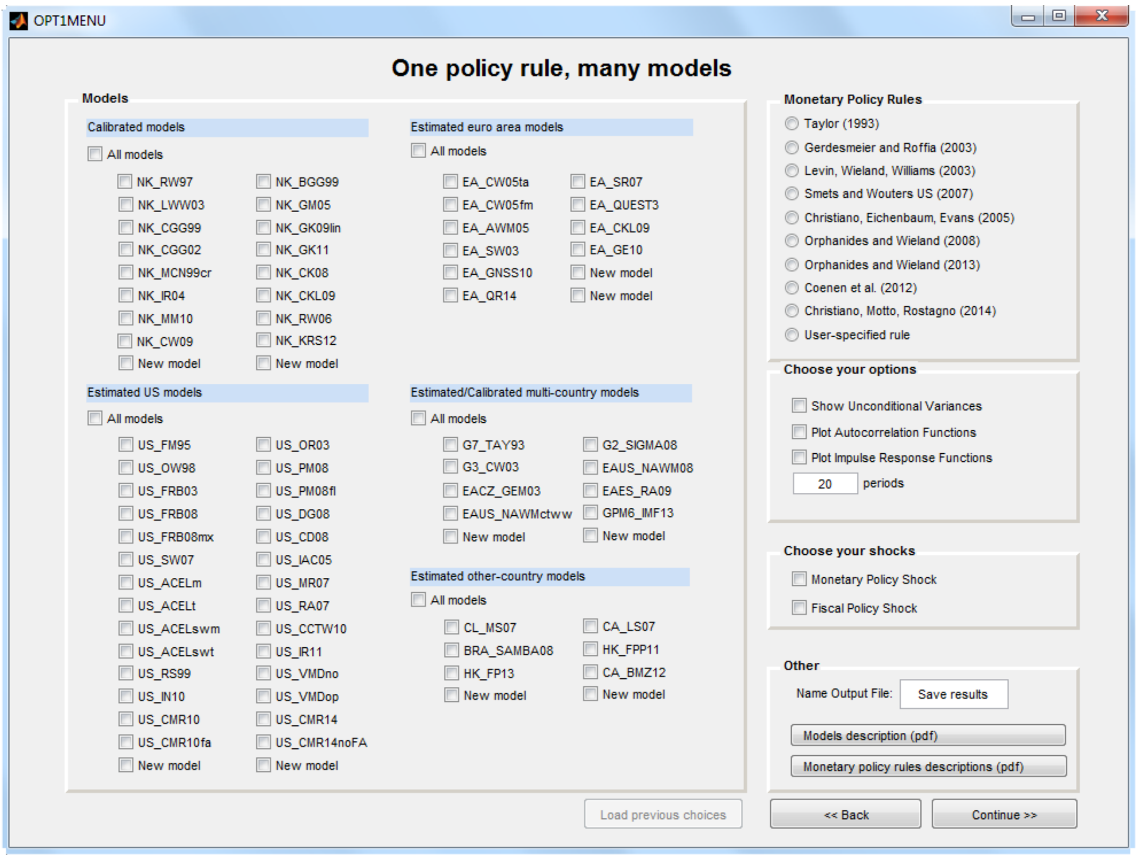
The Macroeconomic Model Data Base (MMB) is an archive of macroeconomic models designed for systematic model comparison and policy analysis. The MMB currently contains more than sixty models ranging from calibrated New-Keynesian models to estimated models of the United States and Euro Area economies, multi-country models and models of other industrial and emerging economies.
Currently two comparison exercises are available:
1) One policy rule, many models
2) One model, many policy rules
A brief description for each exercise is provided under the respective option.
Exercises with robust policy rules and adaptive learning in expectations formation are under construction.
The current MMB utilizes Dynare software (versions 4.2 or 4.3 or 4.4 (up to 4.4.3)) for model solution. Dynare can be downloaded freely from the web. Simulation output is saved in the Output folder.
For information on funding support, contributors and papers using the Macro Mode Data Base, please visit our website below.

Option 1: One policy rule, many models

Under this option users can perform model comparison and analysis under a particular monetary policy rule. Once this option is selected, a menu called *OPT1MENU* will appear. The menu for *One policy rule, many models* is illustrated in **Figure 3**. This menu has several structures that deal with the selection of models to simulate, the selection of a common monetary policy rule used in each model as well as the selection of the statistics and visual output to be displayed. More specifically, the menu for **Option 1** has the following sections: a section on the models available to choose (many models at the same time), a section on the policy rules to be chosen (only one policy rule at a time), a section on the output one would like to look at (unconditional variance of the variables, auto-correlations, impulse responses), and the choice of shocks. Under the section *Other*, one can save the output at the desired file name and also read detailed information about the models and the policy rules in the Modelbase.

Under *One policy rule, many models*, the user can compare the impulse response functions of common variables to monetary policy shock and to fiscal policy shock. The common variables are quarterly output gap, quarterly output, year-on-year rate of inflation and policy interest rate in annual terms. Currently, we consider nine monetary policy rules that are taken from Taylor (1993), Levin et al. (2003), Smets and Wouters (2007), among others. Users are also given a choice to specify their own rule. By default the results generated by the Modelbase will be stored in an excel sheet called *results.xls* in the *OUTPUT* folder. This name of the output file can be customized by the user by writing the desired name in *Name Output File*. The options selected by the user will be displayed in the MATLAB command window.

Figure 3: Modelbase Menu: One policy rule, many models



20 periods

Name Output File: Save results

In the following, each section of *One policy rule, many models* is described step by step.

Models

Under this section, one can select as many models as desired by clicking on the respective model button. When choosing more than one model, be aware that it might take quite some time until all models are solved. A detailed list of the available models in the Modelbase version 2.1 and a short description of each of them is given in a separate document, which can be opened by clicking the button of *Models description (pdf)* under the section *Other* in this menu.

Monetary Policy Rules

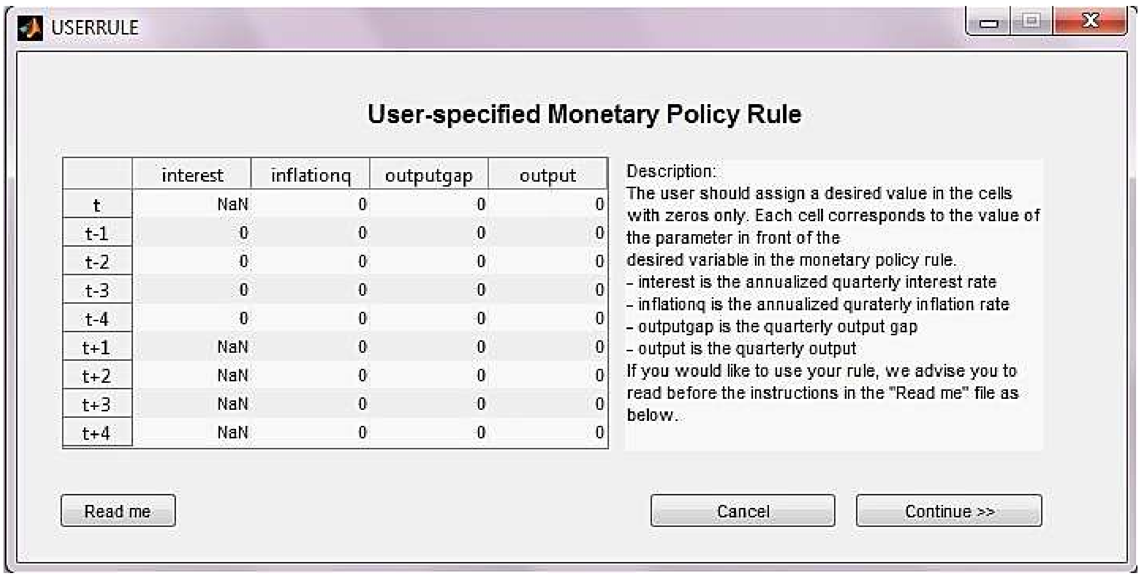
Under this section the user can select a particular interest rate rule. The original rules of the models in the Modelbase have been replaced with a fairly general monetary policy rule that allows many possible parameterizations. Currently the user can choose one of the nine policy rules. A detailed overview of available monetary policy rules is provided in the pdf *MMB_MPrule_description* accessible by clicking of the tab *Monetary policy rules description (pdf)*.[6] A particular rule is chosen by clicking on the respective button in the menu. Only one rule can be selected. The name of the chosen rule will be displayed in the MATLAB command window.

The user also has the option to specify its own parametrization for the monetary policy rule.[7] When selecting *User-specified rule*, another menu displays, as shown in **Figure 4**, called "User-specified Monetary Policy Rule". In this menu, the user should assign a desired value only in the cells with zeros. Each cell corresponds to the value of the parameter in front of the desired variable in the monetary policy rule, which are:

- interest, the annualized quarterly interest rate;
- inflationq, the annualized quarterly inflation rate;
- outputgap, the quarterly output gap;
- output, the quarterly output.

For users that would like to use their specific rule, we provide a *"Read me"* file with explanations and warnings.

Figure 4: MODELBASE MENU: USER-SPECIFIC POLICY RULE


USERRULE

User-specified Monetary Policy Rule

	interest	inflationq	outputgap	output
t	NaN	0	0	0
t-1	0	0	0	0
t-2	0	0	0	0
t-3	0	0	0	0
t-4	0	0	0	0
t+1	NaN	0	0	0
t+2	NaN	0	0	0
t+3	NaN	0	0	0
t+4	NaN	0	0	0

Description:
The user should assign a desired value in the cells with zeros only. Each cell corresponds to the value of the parameter in front of the
desired variable in the monetary policy rule.
- interest is the annualized quarterly interest rate
- inflationq is the annualized qurarterly inflation rate
- outputgap is the quarterly output gap
- output is the quarterly output
If you would like to use your rule, we advise you to read before the instructions in the "Read me" file as below.

Choose your options and your shocks

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

[6]For a more detailed definition and the implementation of these rules in the Modelbase, please refer to the main file MMB.m

[7]This option is available only for MATLAB versions 2007b or more recent.

to twenty periods as a default. 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). The choices will again be documented in the command window. Impulse responses and autocorrelation functions of common variables will be plotted in MATLAB figures while the unconditional variances of common variables of each models will show in the MATLAB command window. The numerical results for the autocorrelation functions and impulse responses will be stored in an Excel file called *results.xls* in the *OUTPUT* folder. The user can also specify its desired name for this file of results.

There are a couple of things that the user should be aware of in order to correctly understand the received output after performing a desired MATLAB exercise. First, for certain model and policy rule combinations, a solution might not exist. In this case, the model neither uniquely satisfies Blanchard-Kahn conditions nor determines the steady state values of certain variables. Second, 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. Lastly, for certain models, the unconditional variances are zero and the autocorrelation functions do not exist. This is the case for some calibrated models in which variances of all shocks are set to zero.

Option 2: One model, many policy rules

Under this option users can do analysis for a particular model under several monetary policy rules. Once this option is selected *OPT2MENU* will appear. The menu for *One model, many policy rules* is illustrated in **Figure 5**. The structure is the same as in **Option 1**. Again, one can select desired models to simulate, desired policy rules as well as statistics and visual output to be displayed. However, in comparison to ***Option 1*** there are the following differences:

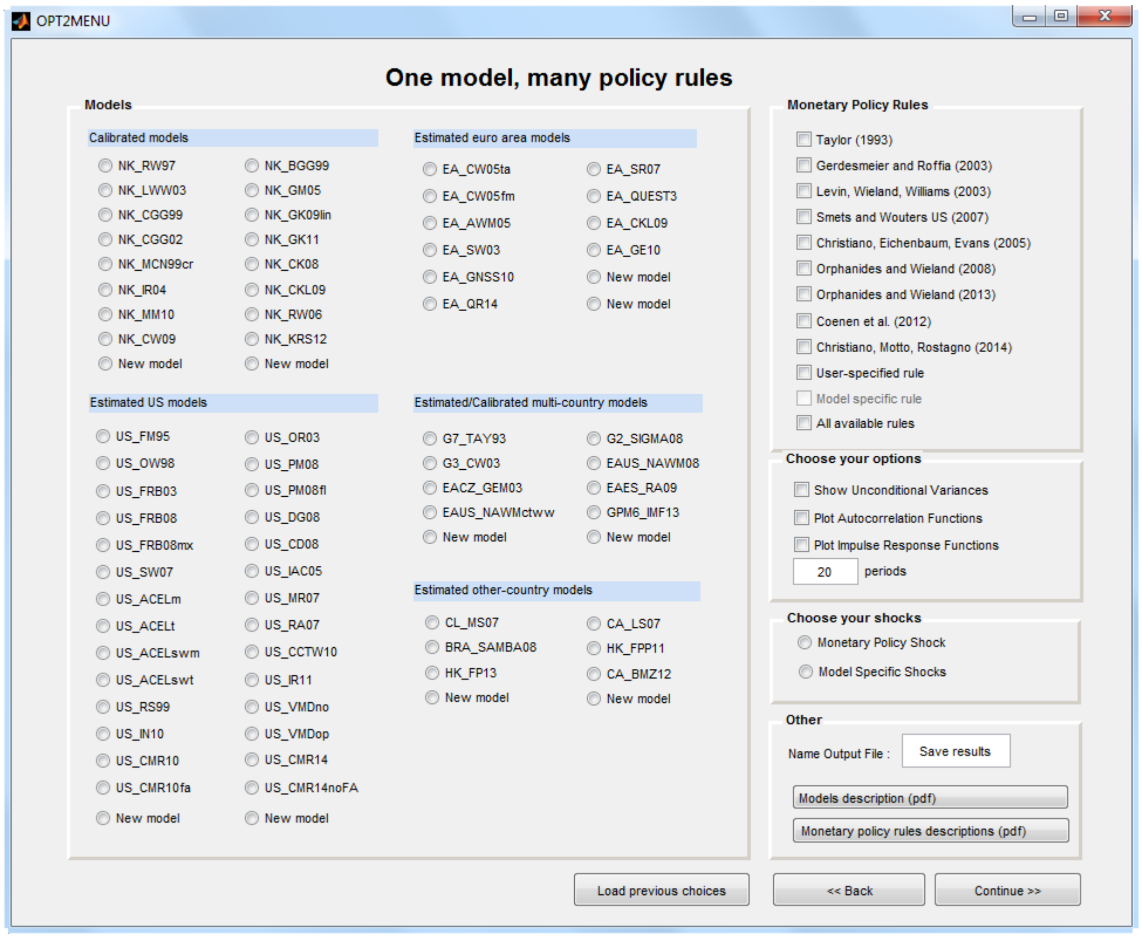
1. The user chooses only one model, but may choose several policy rules at a time.

2. In the *Monetary Policy Rules* section, there is an additional rule, *Model specific rule*. This corresponds to the original monetary policy rule in each of the models. Only if the original policy rule of a model is available, the choice for *Model specific rule* is active. Otherwise, this option is muted.

3. In the *Choose your shocks* section, the user can select either *Monetary Policy Shocks* or *Model Specific Shocks*. When choosing the former, the impulse responses to a one-percent contractionary monetary policy shock are computed. We provide the impulse response functions of all model variables as well as common Modelbase variables in the *all IRFs Mon.Pol.Shock* sheet in the output file in Excel as a default.

In case *Model Specific Shocks* is selected, another menu appears with the names of the shocks used in the selected model and several output choices. For example, if one would choose model NK_RW97, a menu as illustrated in **Figure 6** appears. This model has only three specific shocks: a cost push shock, a monetary policy shock and a fiscal policy shock. The user can choose to plot the impulse responses of the common Modelbase variables, all the variables of the model or alternatively he or she may opt to plot only impulse responses for which the maximum effect, in absolute value, after the shock is larger than a given threshold. The default value of this field is set to 5, meaning that only variables whose maximum absolute deviation from the initial, steady state, value is larger than 10^{-5} will be displayed or saved. For these three cases the same numerical results will be stored in the output file. If a cost push shock, labeled u_ as per the mod file, is selected, the impulse response

Figure 5: MODELBASE MENU: ONE MODEL, MANY POLICY RULES



functions of all variables to a unit cost push shock are computed. The names of model-specific shocks are not self-explanatory. In order to understand which shocks they pertain to, one should consult the Dynare file of the model and the respective paper. Under this menu, the user can also choose to obtain the contemporaneously combined impulse responses of selected shocks.

Simulating a model with the *Model specific rule* allows the user to replicate qualitatively some of the results in the respective paper. If the user wants to replicate the results quantitatively, he or she has to consider the size of a shock used in the paper. For example, let us consider the US_CD08 model, which is a medium-scale New Keynesian model with financial frictions and is estimated on quarterly U.S. data. In De Graeve (2008), the impulse response functions to a one standard deviation (1SD) productivity shock are given in Figure 10 in the paper. Suppose we want to replicate this result. First of all, we have to consult the Dynare mod-file of this model and the paper to find out that the variable name for this shock is epsinno_A and that the estimated standard deviation of the productivity shock is 0.4695. Then, we check the impulse responses in the output excel file and open the *'all IRFs epsinno_A'* sheet. The Modelbase always simulates impulse response functions to one unit shock, therefore, every impulse response function of the US_CD08 model variables needs to be multiplied by 0.4695 (one standard deviation of the productivity shock), in order to replicate the results of the paper.

Figure 6: MODELBASE MENU: MODEL SPECIFIC SHOCKS

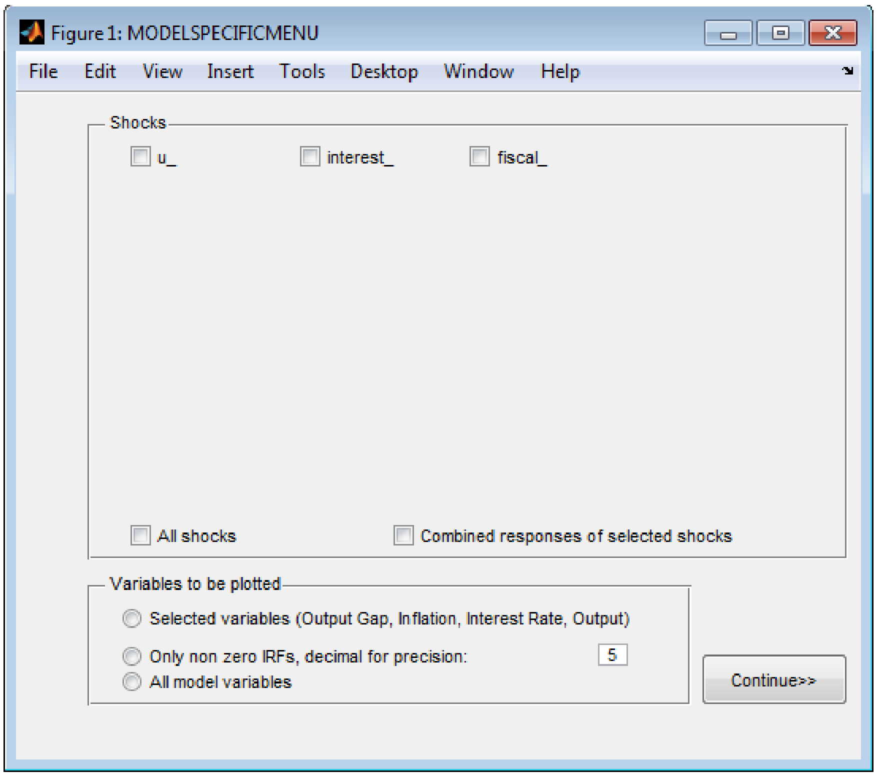


Figure 1: MODELSPECIFICMENU
File Edit View Insert Tools Desktop Window Help
Continue>>

Shocks

- [ ] u_
- [ ] interest_
- [ ] fiscal_
- [ ] All shocks
- [ ] Combined responses of selected shocks

Variables to be plotted

- ○ Selected variables (Output Gap, Inflation, Interest Rate, Output)
- ○ Only non zero IRFs, decimal for precision: 5
- ○ All model variables

In both ***Option 1*** and ***Option 2***, the user can go back or load the previous choices.

3 Structure of the *MMB.m*, *MMBOPT1.m*, *MMBOPT2.m* files

In the following we give a detailed overview of the main MATLAB files corresponding to the Modelbase options discussed above. This subsection and the following ones are designed especially for the Modelbase users who would like to learn more about the technical details of the Modelbase and also add new models and new rules.

MMB.m consists of four parts. In the first part, primary global variables which will be used throughout the Modelbase exercises are declared. The so-called global variables collect the user's choices from the Modelbase options. In the second part, variables relating to models available in the Modelbase are defined. The variable *names* lists all models and is used to produce proper legends of the graphs. In addition, an identification number is assigned to each model in the comment line just next to a model name. We classify all models into five sub-groups: Calibrated models, Estimated US models, Estimated euro area models, Estimated/Calibrated multi-country models and Estimated other-country models. The variables for each sub-group are defined as a collection of identification numbers of models belonging to the respective sub-group. The variable *variabledim* denotes for each model the dimension unit of original model shocks where 1 indicates shocks in percentage and 2 shocks in percent/100. The variable *mycolor* contains a color and line specification for each model, helpful when displaying output in graphs and making comparison among different models. Whenever a new model is added in the Modelbase, the user should make the respective initialization (add entries) in all three variables.

The third part is defining monetary policy rules considered in the Modelbase. In this part of the code, one can observe the vector of policy rules, the original specification of the rule and a description of the rule in terms of the modelbase general rule specification. When adding a new rule, the user should make the appropriate initializations in this part of the code as explained in Section 6. The variable *rulenames* lists all rules implemented in the Modelbase and its variants *rulenamesshort* and *rulenamesshort1* are used as legends in the plots and to display results on the screen. The identification numbers are also given to each rule in *rule_number*. A text block follows which describes a very general monetary policy rule that is used for each model and that nests the policy rules listed in Wieland, Cwik, Mueller, Schmidt, and Wolters (2012), Appendix B. The generalized policy rule is composed of lagged interest rates, the lags and leads of inflation, output gap and output. For each common rule it is shown how the original representation is transformed into the respective general rule representation. For simplicity we create the matrix *common_rule* of which the row number corresponds to the identification number of each rule. A set of policy rule coefficients is set in each row of *common_rule* according to the mapping scheme which is shown in the text block.

One of the policy rules that users employ in ***Option 2*** is the original model-specific rule. The variable *model_with_rule* has a collection of the identification numbers of the models of which the original rule is included in the Modelbase. The variable *model_without_rule* contains the numbers of the models of which the original rule is not implemented. For example, the value 1 in *models_without_rule* means that for model NK_RW97 (which is identified with 1 in the list of models) an original policy rule is not implemented in MMB. Same as for models, the *myrulecolor* variable contains a color and line specification for each policy rule, helpful for displaying output in graphs. In the last part of the *MMB.m*, the main menu of the Modelbase (MAINMENU) is called.

MMBOPT1.m and MMBOPT2.m

The computation for each Modelbase option, in terms of solving the models using DYNARE and computing and displaying results, is done through the *MMBOPT1.m* and *MMBOPT2.m* files, respectively. The structure of *MMBOPT1.m* and *MMBOPT2.m* is illustrated in **Table 1**.[8]

[8]A cell starts with two comment sign, i.e. with %%. When setting the cursor in a cell, the background color of this cell turns to yellow and can thus easily be spotted.

Table 1: STRUCTURE OF THE *MMBOPT1.m* AND *MMBOPT2.m*

MMBOPT1.m	**MMBOPT2.m**
1. Initialization	**1. Initialization**
Storing information obtained from *OPT1MENU*	Storing information obtained from *OPT2MENU*
Specification of a choice set of policy rules	**2. Initializing a loop over selected rules**
Saving parameters for the chosen rule	Specification of a choice set of policy rules
2. Solving the model and computing statistics	Saving parameters for the chosen rule
Initializing a loop over selected models	Solving the Model
Stepwise model solution	Stepwise model solution
Storing model solutions and statistics	Storing model solutions and statistics
3. Obtaining results	**3 Obtaining results**
Extracting statistics for common variables	Extracting statistics for common variables
Plotting results as chosen by the user	Plotting results as chosen by the user
Saving results in an Excel file	Saving results in an Excel file

Before starting with a detailed description, it is important to mention that the information collected from the Modelbase interface (Modelbase options) is saved in a structure variable called *modelbase*. This structure is saved as a MAT-file and is accessible even after a workspace clearing. For example, *modelbase.horizon* saved in *modelbase*, contains the number of periods to be plotted for impulse responses or other graphs. The default value is fixed to 20 periods and one can change the number of periods that are plotted by modifying this entry in the Modelbase interface. **Table 2** lists the remaining most important variables in the structure *modelbase*.

In the following, the specific parts of *MMBOPT1.m* file are explained in detail. *MMBOPT2.m* is written with a similar structure to *MMBOPT1.m*. The major difference is that a loop is initialized over selected rules instead of over selected models because the user chooses only one model but many rules in ***Option 2***.

Part 1 of MMBOPT1.m: Initialization

This first block of the *MMBOPT1.m* stores all the information obtained from the menu of ***Option 1***, *OPT1MENU*. Afterwards, the coefficients of the chosen policy rule are stored in the file *policy_param.mat*. They are loaded later on in the specific model files to initialize the monetary policy equation. Next, all option choices for an exercise are displayed. As in the model files the monetary policy shock has the name *interest_* and the fiscal policy shock has the name *fiscal_*. This convention is important to address the shocks of the right equations after having solved the model.

Table 2: KEY VARIABLES IN *MMBOPT1.m* AND *MMBOPT2.m*

modelbase.totaltime	total CPU time (in seconds) used by the modelbase
modelbase.savepath	path for the Excel file that contains the output
modelbase.names	names of all models
modelbase.variabledim	dimension of model-specific shocks
modelbase.horizon	number of periods to be plotted
modelbase.mycolor	color vector for the graphs
modelbase.rule	chosen rule
modelbase.models	chosen models
modelbase.option(1)	autocorrelation functions
modelbase.option(2)	impulse response functions
modelbase.option(3)*	shock several innovations contemporaneously
modelbase.option(4)*	plot impulse responses for all model variables
modelbase.option(5)	unconditional variances of common variables
modelbase.option(6)*	choose model specific shocks
modelbase.data*	stores the user-specified rule
modelbase.homepath	path of the modelbase folder
modelbase.namesshock	names of shocks contained in the modelbase
modelbase.innos	chosen shocks
modelbase.modeltime	CPU time (in seconds) used for solving each model
modelbase.setpath	paths of the modelfolders of chosen models
modelbase.epsilon	counts the number of loops
modelbase.info	equals 1 if a model has no determinate solution; otherwise 0
modelbase.AUTR	contains the autocorrelation functions
modelbase.AUTlgy_	contains variable names that correspond to the autocorrelation functions
modelbase.IRF	contains the impulse response functions
modelbase.IRFlgy_	contains variable names that correspond to the impulse response functions

** used in MMBOPT2.m only.*

Part 2 of MMBOPT1.m: Solving the model and computing statistics

Before solving the models using DYNARE, all choices made so far have to be saved in the file *Modelbase* since DYNARE clears the workspace before solving a model. A loop over selected models, which are saved in the vector *modelbase.models*, is initialized. For every turn, the current working directory is adjusted to the subfolder of each model. Using the command *dynare* followed by the model name, c.f. *dynare NK_CGG99* to solve the model of Clarida, Gali, and Gertler (1999), calls the software DYNARE to translate the DYNARE syntax in a convenient way. Afterwards the function *stoch_simul_modelbase.m* is called to solve the model and compute autocorrelation functions, impulse response functions and unconditional variances. The results are appended to the file *Modelbase* before we return to the beginning of the loop to solve the next model.

Part 3 of MMBOPT1.m: Plotting the results

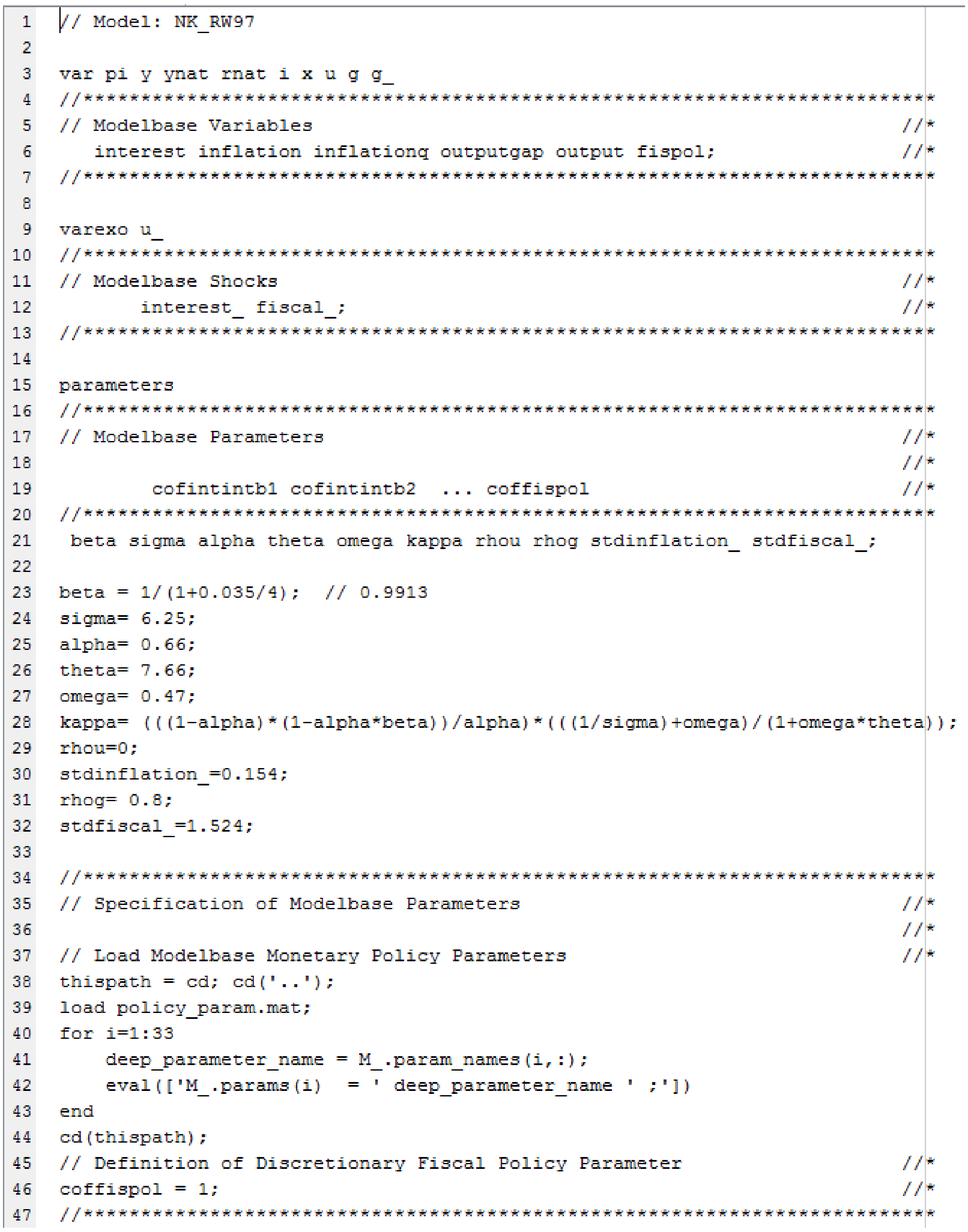
The last part of the *MMBOPT1.m* is devoted to processing and presenting the results. Figures for impulse response functions of the common variables to each shock and autocorrelation functions of the common variables are set up and plotted for each model. The common variables can be easily identified within the whole output of each model using the function *loc* that searches for positions of string variables in vectors like *modelbase.IRFlgy_*.[9]

4 Structure of the model 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 version of the Modelbase. The mod-file is shown in **Figure 7** and **Figure 8**. 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.

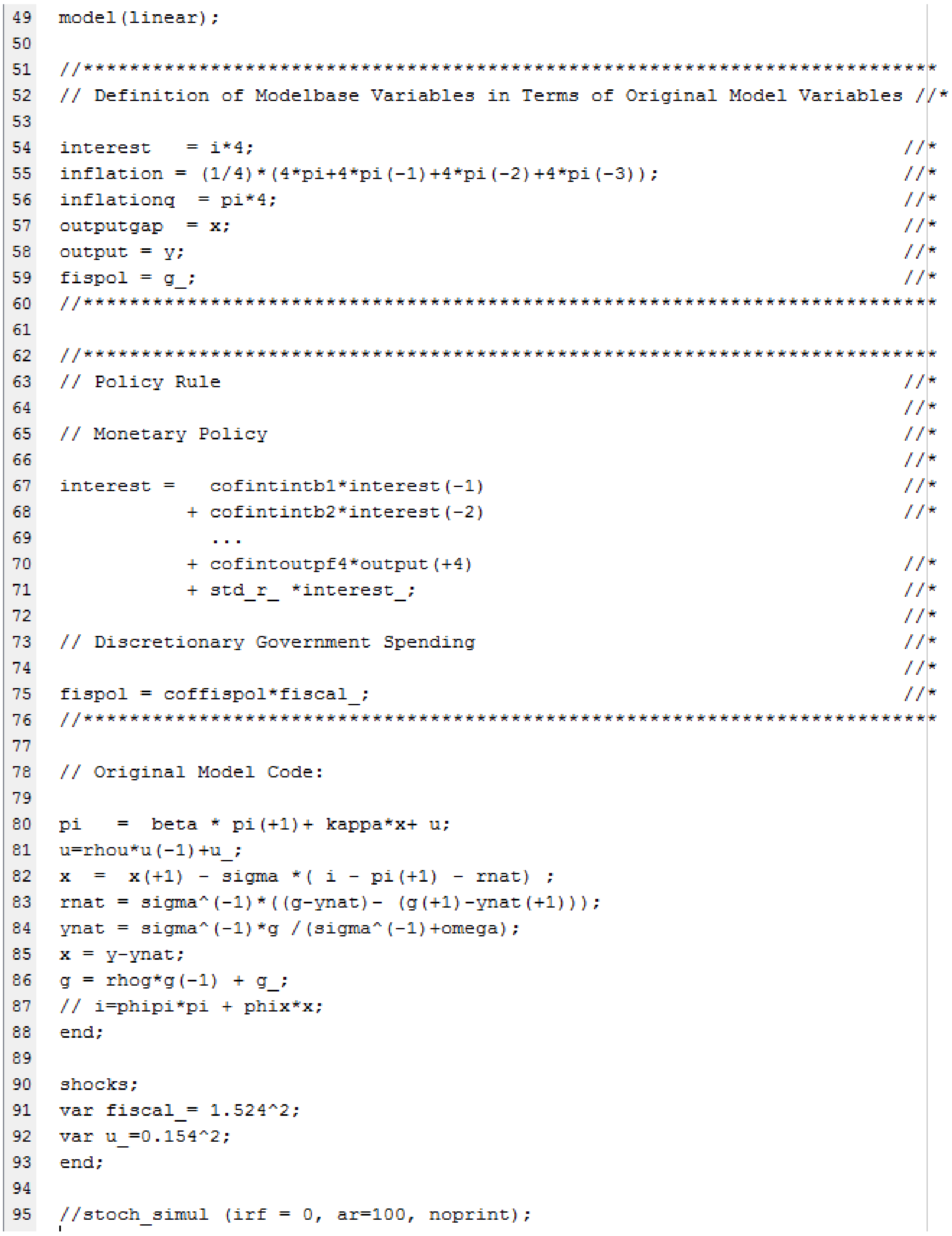
[9]*MMBOPT2.m* makes use of additional functions to process other results and the user should refer to their description in the respective MATLAB file.

Figure 7: 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                                                  //*
                                                                         //*
        cofintintb1 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 = cd; 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 8: Structure of the model files: The Model Block



```
model(linear);

//***************************************************************************
// Definition of Modelbase Variables in Terms of Original Model Variables //*

interest    = i*4;                                                        //*
inflation = (1/4)*(4*pi+4*pi(-1)+4*pi(-2)+4*pi(-3));                      //*
inflationq  = pi*4;                                                       //*
outputgap   = x;                                                          //*
output = y;                                                               //*
fispol = g_;                                                              //*
//***************************************************************************

//***************************************************************************
// Policy Rule                                                            //*
                                                                          //*
// Monetary Policy                                                        //*
                                                                          //*
interest =    cofintintb1*interest(-1)                                    //*
            + cofintintb2*interest(-2)                                    //*
              ...
            + cofintoutpf4*output(+4)                                     //*
            + std_r_ *interest_;                                          //*
                                                                          //*
// Discretionary Government Spending                                      //*
                                                                          //*
fispol = coffispol*fiscal_;                                               //*
//***************************************************************************

// Original Model Code:

pi    =  beta * pi(+1)+ kappa*x+ u;
u=rhou*u(-1)+u_;
x   =   x(+1)  - sigma *( i - pi(+1)  - rnat)  ;
rnat = sigma^(-1)*((g-ynat)-  (g(+1)-ynat(+1)));
ynat = sigma^(-1)*g /(sigma^(-1)+omega);
x = y-ynat;
g = rhog*g(-1)  + g_;
// i=phipi*pi + phix*x;
end;

shocks;
var fiscal_= 1.524^2;
var u_=0.154^2;
end;

//stoch_simul (irf = 0, ar=100, noprint);
```

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 7** 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 4 to 7 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 9 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 12. 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 15, the Modelbase parameters in the Modelbase block are initialized. In **Figure 7** line 19 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 21.
- Afterwards numerical values are assigned to the model-specific parameters in lines 23 to 32.
- Finally a block called *Specification of Modelbase Parameters* is added. First in lines 37 to 44 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 43. However, 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 49 of **Figure 8** 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 51 to 60 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*

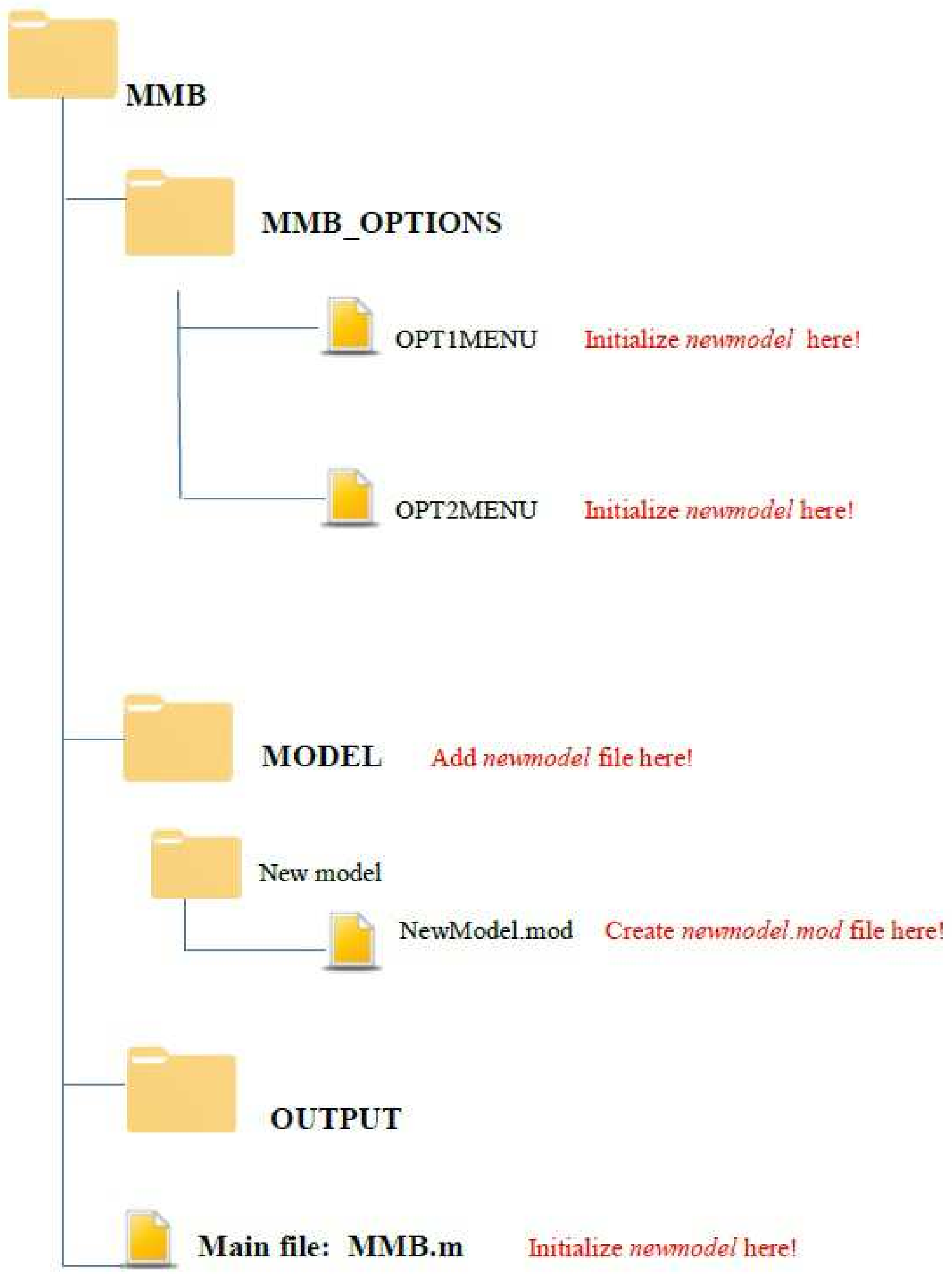
and *output* denote the output gap and output, respectively. The common variable *fispol* represents discretionary fiscal policy. It is set equal to the model-specific government spending shock variable, which in the case of our example is *g_*. Note again, that this model-specific shock has to be initialized 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 65 to 71 the common monetary policy rule is specified. Again for reasons of brevity we have not displayed the complete general policy rule in **Figure 8**. Below in line 73, the common equation for discretionary government spending is specified.
- The original model equations are then specified in lines 78 to 87. 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 86 has remained unchanged. The model section ends in line 88 with the required keyword *end*.
- Finally the variance covariance matrix is specified in lines 91 and 92 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 75 which is set equal to the model-specific government spending shock *g_* in line 59.
- The *stoch_simul* command in line 96 is commented out. Alternatively one can also delete this command.

5 Adding models to the Modelbase

Adding a new model to the data base consists of three 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, the new model has to be initialized in the Modelbase interface. **Figure 9** illustrates the Modelbase folders and in red we attract attention to the folders/files where one should initialize the new model. In the following, each of these steps is described in detail.

Figure 9: ADDING MODELS IN THE MODELBASE


MMB
MMB_OPTIONS
OPT1MENU
Initialize *newmodel* here!
OPT2MENU
Initialize *newmodel* here!
MODEL
Add *newmodel* file here!
New model
NewModel.mod
Create *newmodel.mod* file here!
OUTPUT
Main file: MMB.m
Initialize *newmodel* here!

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 7**

- The common block *Modelbase Shocks* is added after the initialization of the original model shocks as in lines 10 to 13 of **Figure 7**. 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 *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. 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 8**. 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 8**. 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.

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*.

Step 3: Initializing the model in the Modelbase interface.

- As the final step, one initiates the model in the main file *MMB.m* as well as in the Modelbase interface, namely, adding new entries in *MMB.m*, *OPT1MENU.m* and *OPT2MENU.m* files in the MMB_OPTIONS folder.
- In *MMB.m*, the model name has to be added at the corresponding position to the vector *names*. Currently, one can substitute the New_Model with the name of the model one is adding. Next, a new entry has to be added at the corresponding position to the vector *variabledim*. This entry has to be *1* if the standard deviations of the model-specific shocks are expressed in percent and it has to be *2* if the standard deviations are expressed in percent/100. Lastly, a new entry has to be added as well at *mycolor*. Also, the model should be assigned to one of the five model categories in Modelbase.
- The new models should be also initialized in *OPT1MENU.m* and *OPT2MENU.m* files under the MMB_OPTIONS folder. Please consult Section 7 for instructions how to do this.

- The model name should be updated also in the Modelbase graphical interface with Graphical User Interface Developing Environment (in short GUIDE) available in MATLAB. Currently we show the reserved places for new models, under the name *New Model*. For instructions how to do this please consult Section 7.

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 'Monetary policy rules description'. Also, one can see how it practically works by looking at common monetary policy rules already implemented in the *MMB.m* file.

1) Add a common monetary policy rule

- The first task is to choose a suitable name for the new policy rule and to include it into an array for a list of rule names, *rulenames*. One also has to add its acronyms in character arrays, *rulenamesshort* and *rulenamesshort1*.[10] Please make sure that the new rule name is of the same size of character array as the already existing one. Next, one assigns color to the rule with the array *myrulecolor*.
- Then, one adds the specification of the new-rule coefficients right after the last common rule's specification.
- The remaining work is to create a button (or a checkbox) for the new rule in the section of rules in the menu file *OPT1MENU.m* (or *OPT2MENU.m*) using GUIDE. For how to include a new common rule using GUIDE, please refer to Section 7.

2) Add the model-specific monetary policy rule

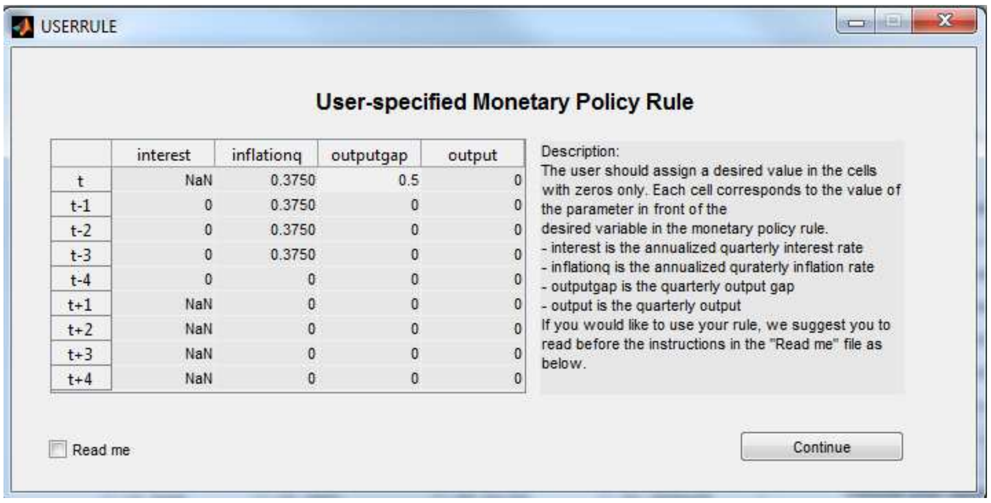
- When adding a new model, it is possible to include its policy rule as long as the rule can be rewritten in terms of Modelbase common variables. If this is the case, the user should add the model identification number to the variable vector *model_with_rule* in the main file *MMB.m* such that a model-specific rule is activated when its corresponding model is chosen in **Option 2**. Otherwise, the user should add the model number to the variable vector *model_without_rule*. For example, if the policy rule is set for the interest rate to react to exchange rate or credit growth, the user cannot include the original rule in the Modelbase.
- Finally, the user has to insert the specification of the model-specific rule into the part of *switch* statement in the file *MSR_COEFFFS.m* using the model's identification number as the case expression.

[10] As *rulenamesshort1* are used for displaying simulation outcomes in Matlab console, any blank is not allowed in the rule name.

3) User-specified monetary ruleAvailable only for MATLAB version 2007b or more recent

- When *User-specified rule* is chosen, 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 10 illustrates how to use the option for a user-specified rule with the example of Taylor (1993) rule.

Figure 10: TAYLOR (1993) RULE USING THE OPTION OF USER-SPECIFIED RULE



USERRULE

User-specified Monetary Policy Rule

	interest	inflationq	outputgap	output
t	NaN	0.3750	0.5	0
t-1	0	0.3750	0	0
t-2	0	0.3750	0	0
t-3	0	0.3750	0	0
t-4	0	0	0	0
t+1	NaN	0	0	0
t+2	NaN	0	0	0
t+3	NaN	0	0	0
t+4	NaN	0	0	0

Description:
The user should assign a desired value in the cells with zeros only. Each cell corresponds to the value of the parameter in front of the
desired variable in the monetary policy rule.
- interest is the annualized quarterly interest rate
- inflationq is the annualized qurarterly inflation rate
- outputgap is the quarterly output gap
- output is the quarterly output
If you would like to use your rule, we suggest you to read before the instructions in the "Read me" file as below.

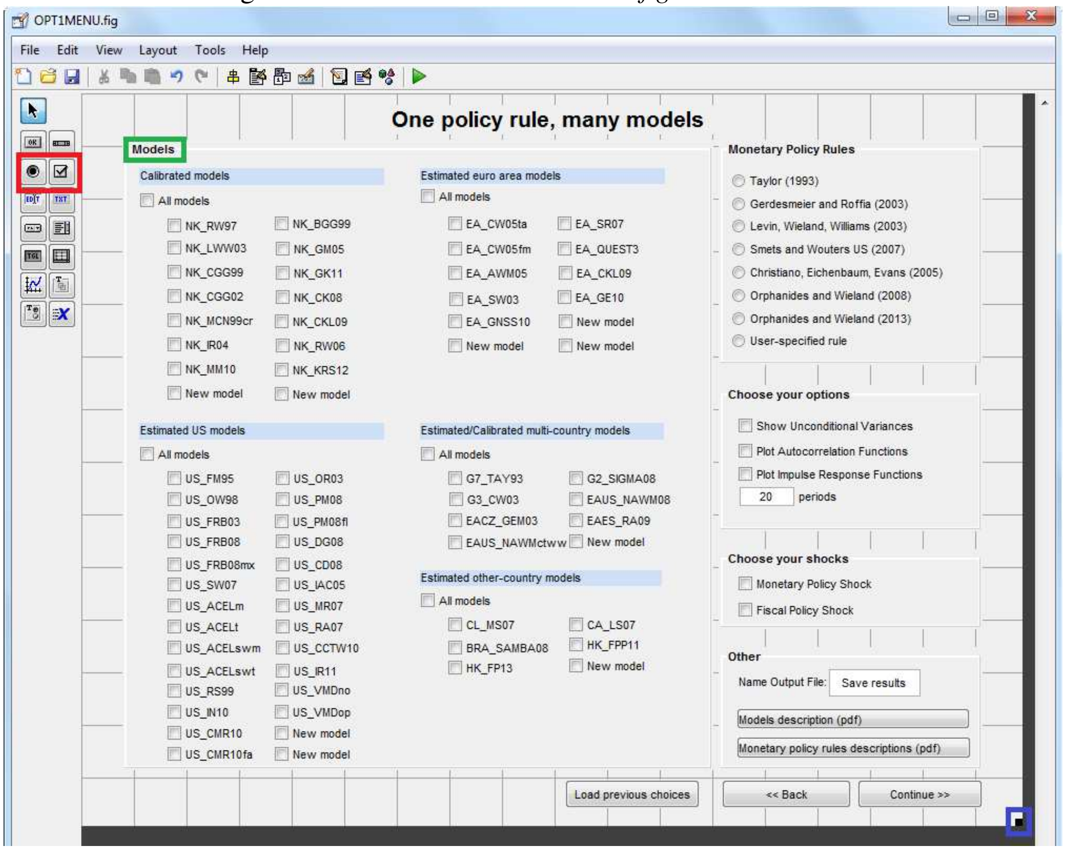
- Note that with certain rule parameterizations, it may happen than some models cannot be solved. This may be the case for several reasons. For example, the system of equations may violate the Blanchard-Kahn conditions so a model does not yield a unique stationary rational expectations equilibrium. There is no clear guideline for conditions for determinacy, but Levin, Wieland, and Williams (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.

7 Instructions on how to use the Modelbase graphical operations

The Graphical User Interface (GUI) embodied in MMB 2.1 is developed in the so-called Graphical User Interface Developing Environment (in short GUIDE) available in MATLAB. To add a new model or a new monetary policy rule, you will deal with some interesting and simple objects that GUIDE offers in a user-friendly way. Implementing a GUI is an object-oriented programming exercise and as such tools that are used and procedures to follow might be sensitive to the MATLAB version. The graphical operations presented next are compatible with MATLAB versions from 2008b onwards. At the same time, they have also been tailored to be as consistent as possible with versions 2007a through 2008a. Specific issues related to each of these versions and instructions on how to solve them are indicated as footnotes.

More generally, a GUI consists of two types of files: the figure itself (a *.fig* file) and a code file (an *.m* file). The latter contains actions to be executed once the user clicks on a tab in the interface. The Modelbase interface uses five types of objects: edit-fields, toggle buttons, panels, checkboxes and a group of radio buttons. Adding a new model or rule primarily requires dealing with the last three objects. Before delving deeper in how to include a new model and rule in the Modelbase it is worth starting with a brief overview of a GUI. **Figure 11** shows a GUI open in GUIDE with update mode. This mode allows the user to edit the GUI. Under MATLAB, opening an interface with update mode works as follows: Type *guide* in the MATLAB command window, and a dialog box will pop up. Under the tab *Open Existing GUI*, browse and choose either *OPT1MENU.fig* or *OPT2MENU.fig*.[11] A figure similar to **Figure 11** will show up.

Figure 11: CREATING *OPT1MENU.fig* WITH GUIDE



The first elements we start with are panels. Their usefulness is twofold: firstly, panels enable the user to create proper designs and to encapsulate objects with similar characteristics, secondly, they can be used to implement a group of mutually-exclusive radio buttons. One may need to resize the panel. For illustration, click on the tab framed in green in **Figure 11** to select the panel *Models* and then adjust its size by clicking and dragging on the edge.[12] When needed, each tab in the panel can

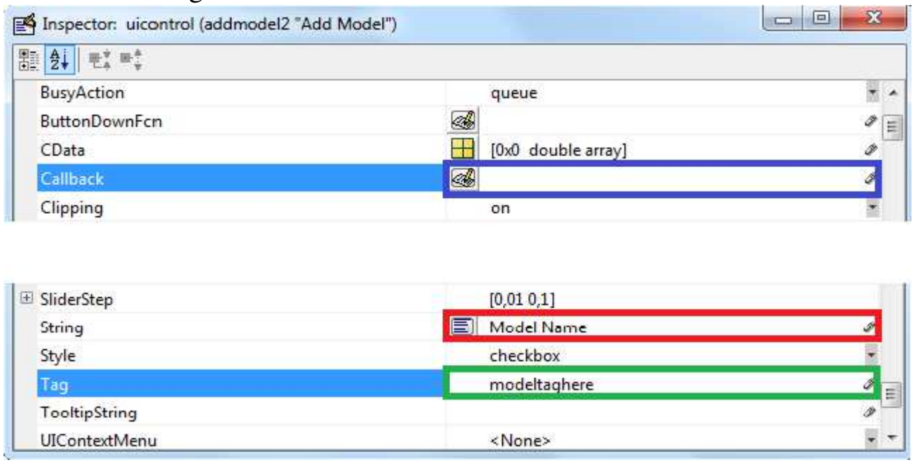
[11]For MATLAB versions older than 2008a, the user needs to launch the GUIDE and choose the option *Blank GUI* under the tab *Create GUI* and from there use the menu file to open GUIs.

[12]The same can be done for the whole GUI figure using the corner illustrated in blue in **Figure 11**.

be displaced by click-and-dragging. The next elements we look at are checkboxes and radio buttons. They are shown in the red box in **Figure 11**. The right tab is for checkboxes while the left is for radio buttons. The checkboxes are used to allow for multiple choices within a panel. This is the case with models in ***Option 1*** and policy rules in ***Option 2***. Conversely, choices are mutually exclusive in panels containing radio buttons; this is the case with models in ***Option 2*** and policy rules in ***Option 1***. One can add a new tab by clicking and dragging each object to the desired place in the GUI figure. Make sure that the object is settled in the appropriate panel. The panel is highlighted when it is hovered over.

After creating a new tab, you need to change its properties. Double-click on the new object to see its properties in the *Property Inspector*.[13] As shown in the **Figure 12**, a couple of fields need to be edited. The first one, *String* (see the box framed in red), is the name that will appear on the GUI figure. From our convention, *String* is the model/rule name for a new model/rule. The second field, the so-called *Tag*, will serve as an object unique identifier in the code file. The name of the *Tag* should only include lowercase letters and digits. According to our convention, the Tag for models implemented in the Modelbase are the model names (in lowercase letters) where the underscore is omitted. For example, the tag of model *NK_RW97* is *nkrw97*. In the case of a checkbox, the callback field, framed in blue, needs to be edited to specify the actions to be executed behind the object in the code file (see below). For radio buttons the callback function is already implemented with the panel. Hence, no need to change it. Now we come more specifically to the procedure of adding new models and new monetary policy rules into the MMB GUI.[14]

Figure 12: PROPERTY INSPECTOR FOR NEW CHECKBOX



Inspector: uicontrol (addmodel2 "Add Model")

BusyAction	queue
ButtonDownFcn	
CData	[0x0 double array]
Callback	
Clipping	on

SliderStep	[0,01 0,1]
String	Model Name
Style	checkbox
Tag	modeltaghere
TooltipString	
UIContextMenu	<None>

Add new models step by step

Henceforth, it is assumed that *modeltaghere* is the Tag of a new model.

- In ***Option 1***: modifications should be done in *OPT1MENU.fig* and *OPT1MENU.m*. Augmenting the first option of the Modelbase with a new model implies proceeding as follows:

  (i) Add a new checkbox in the panel *Models*.

[13]This also is accessible by right-clicking on the object or under the GUI menu Property Inspector.

[14]Under MATLAB 2007b, one can encounter some difficulties in editing callbacks. It is then recommended to use the tabs *New model* already provided on the GUI to implement new models. In that case, only the fields *Tag* and *String* need to be edited.

(ii) Edit the fields *String* and *Tag*.

(iii) Copy and paste the following code in Callback in the property inspector

```
OPT1MENU('models_Callback',hObject,eventdata,guidata(hObject))
```

As objects in this panel are sharing a common callback, one can alternatively copy this line from the property inspector of an existing checkbox in the panel.[15]

(iv) Add a new model Tag in the array of tags *modelslist* in the function *OPT1MENU_OpeningFcn* in *OPT1MENU.m*. In our example, you should append a new model Tag to *modelslist* as following:

$$modelslist = [\cdots, handles.modeltaghere].$$

Make sure that the Tag in the variable *modelslist* has the same ordering as the model name in the variable $names$, see *model_number* in *MMB.m*.

- In ***Option 2***: modifications should be done in *OPT2MENU.fig* and *OPT2MENU.m*. Here an user has to:

(i) Add a new radio button in the panel *Models*.

(ii) Edit the fields *String* and *Tag*

(iii) Add a new model tag in the array of Tags *modelslist* in the function *OPT2MENU_OpeningFcn* in *OPT2MENU.m*. In our example, you should append a new model Tag to *modelslist* as following:

$$modelslist = [\cdots, handles.modeltaghere].$$

Make sure that the *Tag* in the variable *modelslist* has the same ordering as the model name in the variable $names$, see *model_number* in *MMB.m*.

Add new common policy rule

Henceforth, it is assumed that *ruletaghere* is the Tag of the new rule.

- In ***Option 1***: modifications should be done in *OPT1MENU.fig* and *OPT1MENU.m*. Since only one policy rule can be chosen in this option, the steps are very similar to those of adding a model in ***Option 2***. They are the following:

(i) Add a new radio button in the panel *Monetary Policy Rules*.

(ii) Edit the fields *String* and *Tag*.

(iii) Add the rule *Tag* in the existing list of other rule tags, *ruleslist* in the function *OPT1MENU_OpeningFcn* in *OPT1MENU.m* as below.

$$ruleslist = [\cdots, handles.ruletaghere].$$

Make sure that the *Tag* in the variable *ruleslist* has the same ordering as the rule name in the variable $rulenames$ in *MMB.m*.

[15]This alternative is not available under MATLAB version 2008a.

- In ***Option 2***: modifications should be done in *OPT2MENU.fig* and *OPT2MENU.m*. Like for the models in ***Option 1***, the choice of monetary policy rules in ***Option 2*** are implemented with checkboxes. The steps of adding a new one are therefore:

  (i) Add a new checkbox in the panel *Monetary Policy Rules*.

  (ii) Edit the fields *String* and *Tag*

  (iii) In the property inspector, write the following line in the field *Callback*:

  ```
  OPT2MENU('rules_Callback',hObject,eventdata,guidata(hObject))
  ```

  Again this is the same *Callback* for all rules in this panel.

  (iv) Add the rule Tag in the existing list of other rule Tags, *ruleslist* in the function *OPT2MENU_OpeningFcn* in *OPT2MENU.m* as below.

  $ruleslist = [\cdots, handles.ruletaghere]$.
  Make sure that the *Tag* in the variable *ruleslist* has the same ordering as the rule name in the variable $rulenames$ in *MMB.m*.

References

Clarida, R., Gali, J., Gertler, M., 1999. The science of monetary policy: A New Keynesian perspective. Journal of Economic Literature 37(4), 1661–1707.

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.

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

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 (3), 523–541.