Developing forecasting models using `SpecEval` add-in for Eviews*

Kamil Kovar†
CERGE-EI‡ & Moody's Analytics

April 1, 2021

Abstract

This document illustrates how Eviews user add-in `SpecEval` can be used for developing forecasting models. The add-in was developed as a tool for interactive and iterative model development process in which model developer uses information about forecasting performance of current model to propose improvements to it. Correspondingly, the illustration is made in terms of applications which start from simple model specification and then proceed to refinement of the model based on information about forecasting performance gathered from the `SpecEval` add-in outputs. The applications are chosen to be diverse so that they illustrate different functionality of the `SpecEval` add-in.

*The full title of this paper should probably be "Developing *single-equation multivariate* forecasting models with *focus on medium horizon and frequency forecasting* using `SpecEval` add-in for Eviews". While `SpecEval` can be used also for mutliequation models of the VAR family, and in future also for other multiequation models, the idea behind most of the functionality comes from single-equation multivariate model. Correspondingly, the focus of this paper is on these types of models. Similarly, while the add-in can be used for data with any frequency and forecasts of any horizon, its graphical components are most suitable for medium and low frequency and medium horizons. Correspondingly all the applications will be focused on macroeconomic time series with monthly or quarterly frequency and medium forecasting horizon.

†Email: Kamil.Kovar@cerge-ei.cz.

‡CERGE-EI is a joint workplace of Charles University and the Economics Institute of the Academy of Sciences of the Czech Republic, P.O. Box 882, Politickych veznu 7, 111 21 Prague 1, Czech Republic.

1 Introduction

The proliferation of economic data and computational power over last two decades lead to explosion in development of forecasting models. This means not only that the population of model developers has increased, but even more importantly that it diversified away from economists employed by government and international institutions and towards practitioners. This brings with itself new challenges since practitioners face very different constraints when developing forecasting models, especially in terms of time available for development. Similarly, the objectives of practitioners are different: while academia and government institutions pay close attention to causality and econometric validity thanks to focus on policy analysis, practitioners are less likely perform policy analysis and hence are more focused on capturing co-movements than causality; similarly, for institutions unconditional forecasting is often primary goal, while practitioners are often more focused on conditional and scenario forecasting.

These factors together mean that many model developers need tools to analyze model forecasting performance quickly and flexibly. The Eviews user add-in `SpecEval` is aimed at facilitating such fast and flexible evaluation of forecasting performance. However, in contrast to existing tools the main goal of the add-in is not simply reporting of summary measures of forecasting performance. The underlying idea is that model development should be *interactive* and *iterative*. It should be *iterative* in the sense that it proceeds in step, each providing improvement over previous model. It should be *interactive* in that the improvements to the model should be based on analysis of shortcoming of the initial model. Crucially, numerical summary of forecasting performance fails on the goal of interactiveness, since numerical summary does not provide (almost) any information on when and why do the model fails to produced good forecasts, and therefore how it should be improved. For this reason the `SpecEval` add-in is focused on producing graphical information about the forecasting performance, mostly in form of visualizing the forecasts themselves.

Focus on visualizing forecasts is also important for second reason: time series models are often too complex for humans to completely understand what will be the exact shape of forecasts produced by these models, and hence too complex to full understand the model behavior. Visualizing multiple forecasts helps in this regard since it provides illustration of the behavior of the model under various historical or scenario conditions. Related to this focus on graphical visualization of forecasts is, among other functionality, (a) the ability to choose between in-sample and out-of-sample forecasting, since out-of-sample forecasting often does not provide useful information about the behavior of the final

estimated model; (b) ability to perform decomposition of forecasts into forecast drivers; and (c) ability to create (conditional) scenario forecasts. Taken together, the focus and functionality of the add-in make it a novel tool for building forecasting models, not only in context of Eviews program, but in context of other programs used for statistical analysis such as R and Python.

The rest of this document proceeds as follows. First, a basic use of the add-in is presented, what provides a background for the following discussion. The following discussion is then made in terms of 8 separate applications with varying length. Most of the basic functionality of the add-in is in first application, which therefore constitutes a starting point. The next three applications then each focus on particular functionality of the add-in, but also provide additional illustration of its intended use in terms of the interactive and iterative model building process. All first 4 applications focus solely on conditional forecasting exercises; two types of unconditional forecasting are taken up in applications 5 and 6. Finally last 2 applications focus on demonstrating how the add-in can be leveraged for situations when the existing functionality is not sufficient so that the user needs to program additional functionality.

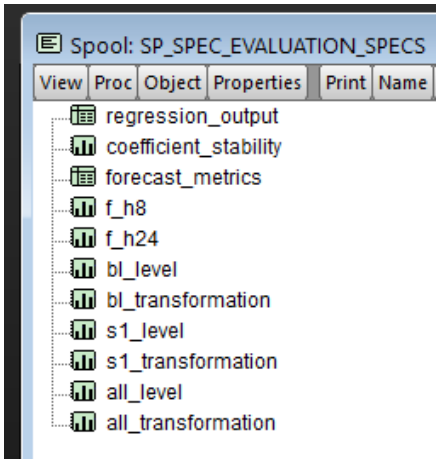
2 Basic use of `SpecEval` add-in

This section briefly describes the basic use of the `SpecEval` add-in to provide background for the more advanced applications in next section. The section is divided into two parts, one corresponding to using the add-in on single specification, and one for using it for multiple specifications. The difference between these use cases is in terms of organizing and reporting results: if multiple specifications are specified than the main spool will have results organized in a way that facilitates comparison across specifications; nothing changes in terms of execution, which is still performed one specification at a time.

Before the add-in can be used, it needs to be installed. This can be done by downloading the installer from the Eviews add-in website and executing it. The add-in can then be either executed from the GUI for any of the three supported object types (equations, VARs and strings), or as object procedure in command line or program. In the latter case one simply needs to specify the proc 'speceval' the same way one executes other object procedures:

```
{%equation}.speceval(options)
```

Figure 1: Spool with output objects - single specification case



The discussion of all options is not in scope of this paper - they are all documented in the documentation of the add-in. That said, this paper does illustrate large share of these options as part of the applications.

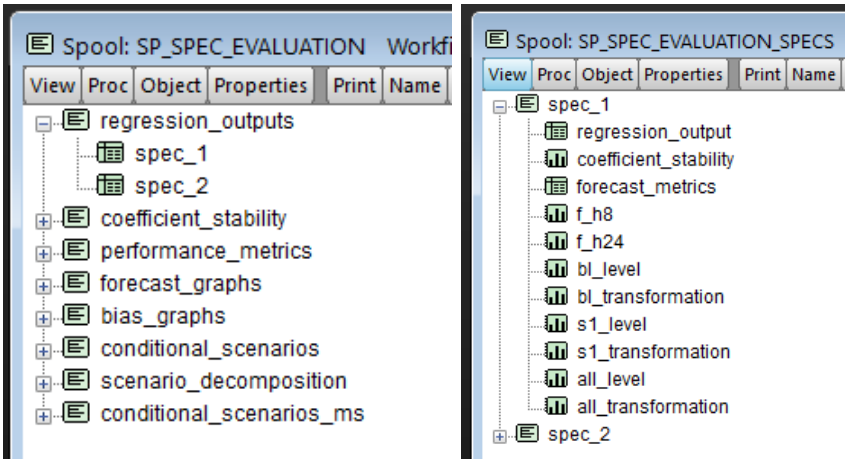
Single specification. The basic use of the `SpecEval` add-in is in terms of evaluating single specification. In such case the executing the add-in will result in a spool called sp_spec_performance being produced. This spool will contain multiple objects, both tables and graphs, that allow user to perform the key goal the add-in: evaluation of forecast performance of given model. See Figure 1 for illustration. Detailed discussion of all the output objects produced by the add-in is out of scope for this document, even those most will be illustrated in the applications below; see Table ?? for overview with short description of all the outputs, and document 'Outputs for specification evaluation add-in' for detailed explanation and illustration of all the outputs. As the table makes clear, most of the outputs are focused on visualizing forecasts, either backtest forecasts (outputs 6-8), or scenario forecasts (outputs 10-13) In addition there are tables with numerical summaries of forecast performance, as well as graph capturing bias of the forecasts. Finally, the spools will also include standard Eviews regression output with several additions and adjustments that facilitate the model development, as well as coefficient and model stability graphs.

While the discussion of the full range of settings options is out of scope for this document it is worth highlighting some key options. First, given the large number of potential outputs the add-in allows for easy and flexible customization of the execution list. The default does not contain all the output objects given that full list of output objects is unlikely to be desired in most cases. User can freely add and/or substract from the list, using the key words corresponding to each output type, which can be

Table 1: List of outputs from `SpecEval` add-in

Object name	**Description**
Regression output	Adjusted regression output table
Coefficient stability graph	Graph with recursive equation coefficients
Model stability graph	Graph with recursive lag orders
Performance metrics table	Table with values of forecast performance metrics
Performance metrics table (multiple specifications)	Table with values of forecast performance metrics for given metric for all specifications
Forecast summary graph	Graph with all recursive forecasts with given horizons
Sub-sample forecast graph	Graph with forecast for given sub-sample
Subsample forecast decomposition graph	Graph with decomposition of sub-sample forecast
Forecast bias graph	Scatter plot of forecast and actual values for given forecast horizon (Minzer-Zarnowitz plot)
Individual conditional scenario forecast graph (level)	Graph with forecast for single scenario and specification
Individual conditional scenario forecast graph (trans.)	Graph with transformation of forecast for single scenario and specification
All conditional scenario forecast graph	Graph with forecasts for all scenarios for single speciication
Multiple specification conditional scenario forecast graph	Graph with forecasts for single scenario for multiple specifications

Figure 2: Spools with output objects - multiple specifications case



found in the documentation for the add-in. Second, the add-in allows great deal of customization of the evaluation procedure. For example, one can change the horizons for which performance metrics will be calculated by including settings option 'horizons_metrics'; or horizons for which forecast summary graphs will be created by including settings option 'horizons_graphs'; or even adjust the starting and/or ending date of the evaluation procedure by including settings parameter 'tfirst_test', respectively 'tlast_test'. Lastly, user can also customize the outputs. For example, the forecast summary graphs can be changed to several different transformations by including settings option 'trans'. Similarly, one can also adjust the sample of these graphs by including settings parameter 'tfirst_graph', respectively 'tlast_graph', and include additional series useful for evaluation of forecast performance by specifying them in settings option 'graph_add_backtest'. Use of these and other settings options will be the main focus of several application below.

Multiple specifications. While the basic use of the `SpecEval` add-in is for single specification, the great advantage of the add-in lies in the fact that it is tailored so that it facilitates direct comparison across alternative specifications. This is achieved by appropriate structuring and reporting of results. Specifically, when the add-in is executed for multiple specifications then it will produce two types of spools. The main spool will be organized by type of output object, i.e. output objects of particular type for all specifications will be collected together in single subspool object. The auxiliary spool will be organized by specification, i.e. for each specification it will contain subspool corresponding to single specification execution. See Figure 2 for illustration.

To execute the add-in for multiple specifications, one just needs to provide a list of specifications as one of the settings options of the add-in, while still executing the add-in from single equation object.

For example, if there are three specifications that are being considered, `EQ01`,`EQ02` and `EQ03`, then one can evaluate the three specifications together by executing following command:

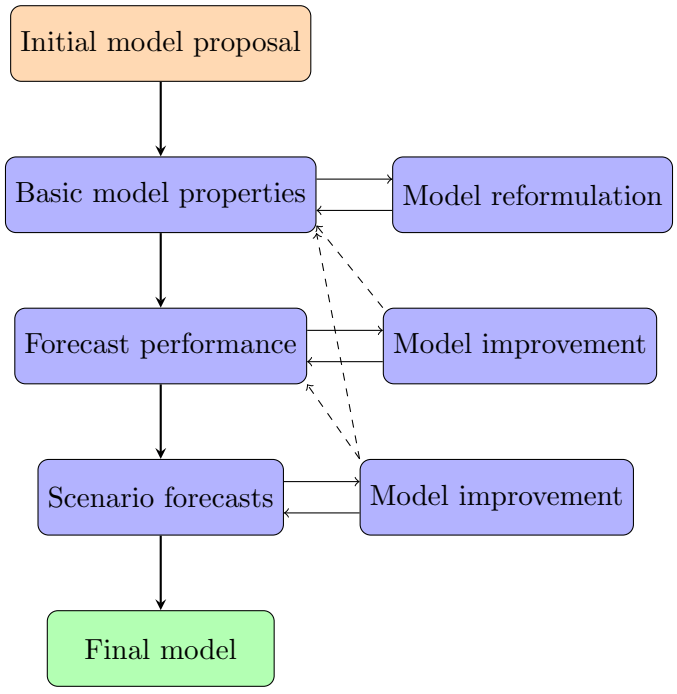
```
EQ01.speceval(spec_list="EQ01 EQ02 EQ03")
```

There are few comments in order. First, the order in the 'spec_list' matters in that the resulting outputs will follow this order. Second, in the above example the 'spec_list' argument contains the original equation, but this is not necessary; if it would not, it would be included by default as a first specification. Third, the add-in allows for use of "*" as wildcard.

3 Applications

This section will illustrate the full functionality of the add-in by focusing on 8 separate applications. While the primary goal is demonstration of the options[1] and outputs of the add-in in order to facilitate future use of the add-in by users, the goal in some sense is broader: this section also shows the interactive and iterative model building process that the add-in is meant to facilitate. The workflow of the model development process is in Figure 3.

Figure 3: Model development workflow



[1] The main text will include commands with the add-in options that were used to produce the outputs. In addition, Appendix A includes complete set of commands for each application. The data and programs are available on my personal website.

The process starts with initial proposed model, which is then evaluated using the outputs of the add-in. The initial model is best evaluated by looking at the forecast summary graphs which allow model developer to visualize the model forecasts and compare them with the actual values, and hence obtain some qualitative notion of forecast quality and forecast errors: for single model the quantitative forecast precision metrics are in most cases uninformative, since their range and lower bound is not known.

Next, based on this evaluation of forecast quality one can propose alternative models. Once one considers multiple model specifications, forecasting performance is more easily evaluated by using the forecast precision metric, especially if the number of considered models is large, in which situation the color coding plays an important role. That said, forecast summary graphs still can provide lot of value. Finally, when potentially large number of forecasting models is reduced to several final candidates, one can switch focus to detailed outputs from the add-in, such as the exact profiles of sub-sample or scenario forecasts.

Plan. The remainder of this section is divided into 8 sub-sections, one corresponding to each application. Each of the application is meant to demonstrate specific functionality of the add-in, as well as show how it can be leveraged as part of the iterative model building process outlines above. First application focuses on standard trending macroeconomic variable and serves to show all the basic features of the add-in. Second application highlights the ability of the add-in to use different transformation of forecasted variable, focusing on the growth rate transformation in context of variable for which growth rate transformation is more commonly of interest. The section also shows the functionality of the add-in with respect to automatic model selection. Third application focuses on different transformation - spread between two variables - and highlights its importance in case when majority of variation of particular variable arises in variations of different variable; the context used here will be standard financial variable. Final transformations will be covered in fourth application, that will focus on exponentially growing variable, in context of which logarithmic and/or ratio transformation is of special value.

Applications 5, 6 and 7 focus on how to perform different types of forecasting exercises using the add-in. Applications 1-4 focus solely on conditional forecasting; these are then complemented by applications 5 and 6 which focus (semi-)unconditional forecasting. First, application 5 considers performing unconditional forecasting when forecast for independent variables are supplied as input to the add-in. This is relevant when there is no feedback between the forecasts for the dependent

Table 2: List of applications

#	Primary focus	Secondary focus
1	Basic use of add-in and overview of key output objects	Iterative and interactive model development process
2	Basic use of transformations (growth)	Recursive automatic model selection
3	Advanced use of transformation (spread)	Interactive model development
4	Advanced use of transformation (log and ratio)	-
5	Unconditinal forecasts I - Exogenously produced forecasts	Use for identities
6	Unconditiona forecasts II - Systems of multiple individual equations	-
7	Custom re-estimation	-
8	Using intermediate objects	-

and independent variables. Next, application 6 focuses on situation when one is evaluating model consisting of multiple single-equation multivariate model, variables of which influence on each other *mutually*. Finally, while the add-in allows the user to customize the forecasting process to a great deal, there are still use cases in which the user needs to perform more complex re-estimation which cannot be dealt with within the re-estimation subroutine of the add-in itself; an illustration of such custom re-estimation is in application 7. This last application will show how user can store intermediate objects from the execution of the add-in and perform his/her own analysis on those.

3.1 Basic application

The first application considers the most basic use of the add-in. As an example our goal will be developing time series model for industrial production for Czechia. The series is displayed Figure 4.

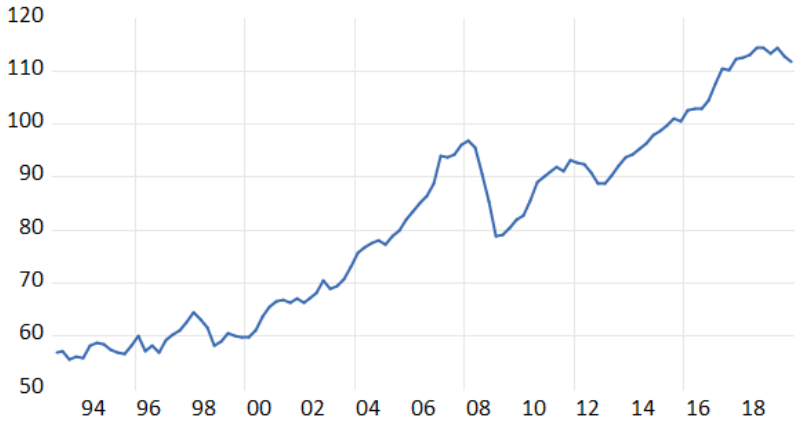
Univariate models. As preliminary step, consider using Eviews automatic ARMA model selection to chose best ARMA model. The resulting model when one uses SIC is ARIMA(0,1,2) model.[2] We can then apply the `SpecEval` add-in to the resulting equation by simply issuing following command[3]:

```
eq_ip_arma.speceval(noprompt)
```

[2] Figure 4 makes it clear that the series is both trending and that it does not have deterministic trend. At the same time, the nature of series suggests that it should be growing exponentially over long horizons, even though this is not exactly clear from the figure. Correspondingly, the natural way to model the series is in log-differences. Therefore, I specify logarithmic transformation and force at maximum 1 differencing. See Appendix A for exact command.

[3] The noprompt option is included so that there is no user dialog issued. If other options would be specified, this is unnecessary.

Figure 4: Level of Czechia industrial production



This will create and display spool that includes 4 objects.[4] Rather than discussing these, I consider estimating another 2 ARMA models and including them among the specifications: simple ARMA(1,1,0) model and ARMA model selected by AIC, ARMA(4,1,4). Once we have those equations estimated, we can run `SpecEval` add-in for all three of them as follows:

```
eq_ip_arma.speceval(spec_list="eq_ip_arma*")
```

As a result the program will create and display spool that among other things includes table with forecast precision metrics for all three specifications, see Table 3.[5] The table shows that the more complex model selected by AIC leads to substantially worse forecasting performance at both selected horizons, but there there is little difference between the two more parsimonious models. The spool also includes the regression results for each of the equations, so that one can quickly check and compare them. For example, in present case this would reveal that the ordering of the models by RMSE is exact opposite from what one would conclude from looking at adjusted R-squared.

Apart from the performance metrics tables the spool also includes forecast summary graphs. These are displayed in Figure 5. Among other things, the figure makes it clear that neither of the models provides good forecasts during the stress periods, and specifically during the Great Recession. This will be mine raison d'entre for multivariate models I consider next.

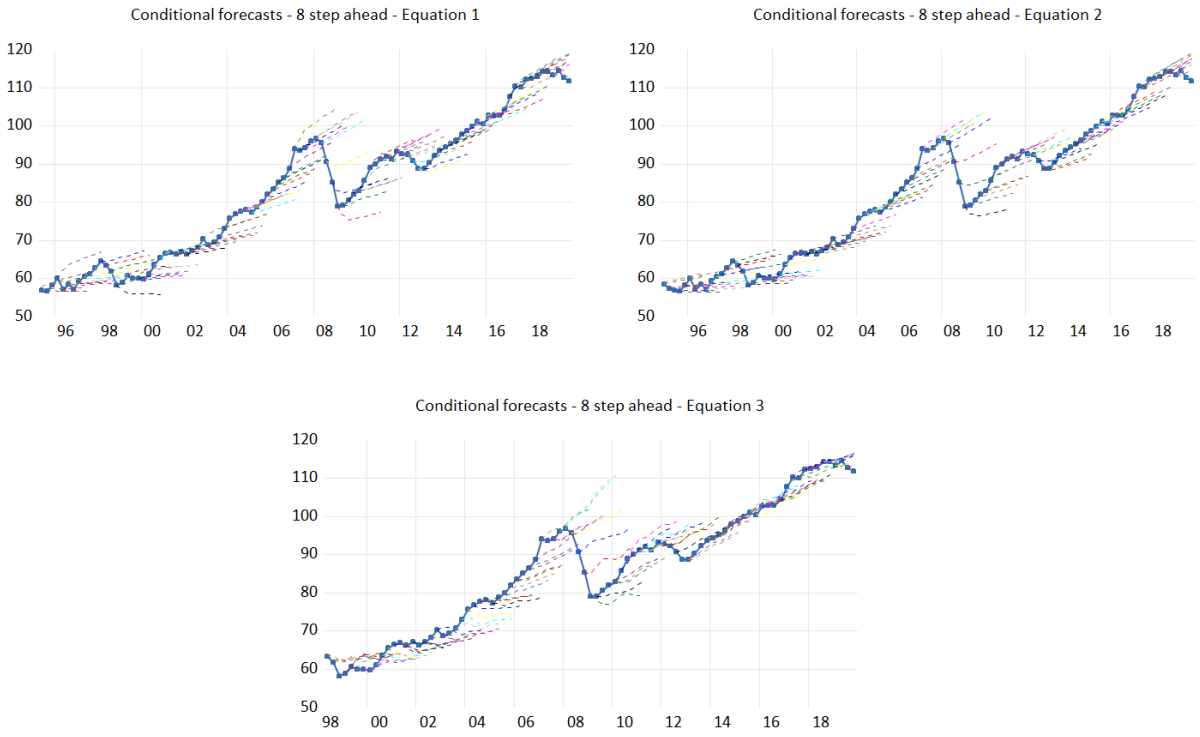
[4] The 4 objects are regression output, forecast summary metrics and 2 forecast summary graphs corresponding to two default horizons, 8 and 24 peirods; see Table ?? for brief description. This corresponds to the default execution list. Later executions will include additional outputs corresponding to the changes to the default settings.

[5] The document 'Outputs for specification evaluation add-in includes detailed discussion of the values and the colors. Briefly, the numbers are average RMSE for all available backtest forecasts, while colors go from green to red, with green corresponding to lowest values and red to highest.

Table 3: Forecast precision metrics - RMSE - ARMA models for industrial production

Specification	Forecast horizons 8	24	Avg.
1	7.52	12.9	10.2
2	7.41	12.6	10.0
3	8.45	14.8	11.6

Figure 5: Forecast summary graph - ARMA model for industrial production



Multivariate models. If one considers using other variables for forecasting industrial production, it is natural to start wit GDP: after all, industrial production is part of GDP, so GDP should provide a lot of information relevant for forecasting industrial production.[6] Consider starting with simple static regression linking log-difference of industrial production to log-difference of GDP, as in equation (1):

$$dlog(IP_t) = \beta_0 + \beta_1 dlog(GDP_t) \tag{1}$$

We can either evaluate the model on its own, or include one of the ARMA models as benchmark:

```
eq_ip_static.speceval(noprompt)
```

```
eq_ip_static.speceval(spec_list="eq_arma", use_names="t",
   graph_add_backtest="gdp[r]")
```

In the second case I have specified that I want to use equation names in the output objects, as opposed to equation aliases (numbers), since this allows me to easily figure out which specification is which. And I have also included the GDP series in the backtest graphs, assigning it to right axis. The resulting forecast summary graphs are in Figure 6. The figure shows that while the static equation does better job forecasting industrial production during Great Recession compared with the univariate models, it does not improve the forecasts that much. Moreover, Table 4 shows that the RMSE of the forecasts is actually higher than for the benchmark model. However, looking at Figure 6, one can see that this might be artifact of the large errors in beginning of sample, when the coefficient estimates are based only on few observations. To explore this issue further, one could do several different things:

- Include coefficient stability in the execution list of the program, executed by following command:

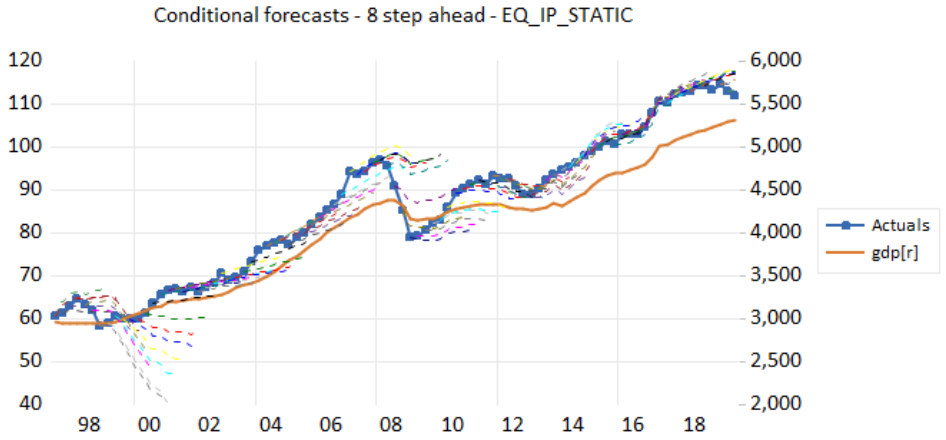
```
eq_ip_static.speceval(exec_list="medium stability")
```

- Cut the backtesting sample to start in 2000q1 for both specifications, executed by following command:

[6] Few warning comments are in place. The focus here is on conditional forecasting, as opposed to true forecasting. This means that in contrast to standard VAR models, the models below include concurrent GDP as explanatory variable, rather than lag of GDP. While this might seem strange from the perspective of current academic econometrics, the alternative would lead to very bad forecasts from conditional perspective; and it would also have very little value in scenario forecasting. For example, it would lead to forecasts where industrial production would be roughly unchanged in presence of large changes in GDP.

Similarly, no consideration is given to econometric validity and/or causality of the coefficients estimates. The sole focus here is on forecasting performance, from which perspective causal interpretation of coefficients is not important; simply, coefficients capture co-movement paterns.

Figure 6: Forecast summary graph - static regression for industrial production




```
eq_ip_static.speceval(spec_list="eq_arma", use_names="t",
   graph_add_backtest="gdp[r]", tfirst_test="2000q1")
```

- Perform the backtesting for both specifications in-sample rather than out-of-sample, executed by following command:

```
eq_ip_static.speceval(spec_list="eq_arma"), use_names="t",
   graph_add_backtest="gdp[r]", oos="f")
```

Table 4: RMSE - ARMA models for industrial production (restricted sample)

	Forecast horizons (# of steps ahead)		
Specification	8	24	Avg.
EQ_IP_STATIC	8.22	22.2	15.2
EQ_IP_ARMA	7.52	12.9	10.2

Figure 7 shows the recursive coefficients for the static equation, and demonstrates that indeed the coefficients in beginning of sample are nonsensical, since the coefficient on GDP is negative. Presumably, such model would not be used and hence these forecast should not be included in evaluation of the forecasting model. Meanwhile, Figure 8 shows the forecast summary graphs when either the initial periods are excluded (left panel) or when the forecasts are created in-sample (right panel). In either case the conclusions about quality of the forecasts are improved. This is also confirmed by looking at Tables 5 and 6, which show that when either only forecasts from 2000q1 onward, or when in-sample forecasts are considered then the static equation produces better forecasts than the benchmark ARMA

Figure 7: Coefficient stability graph - static equation for industrial production

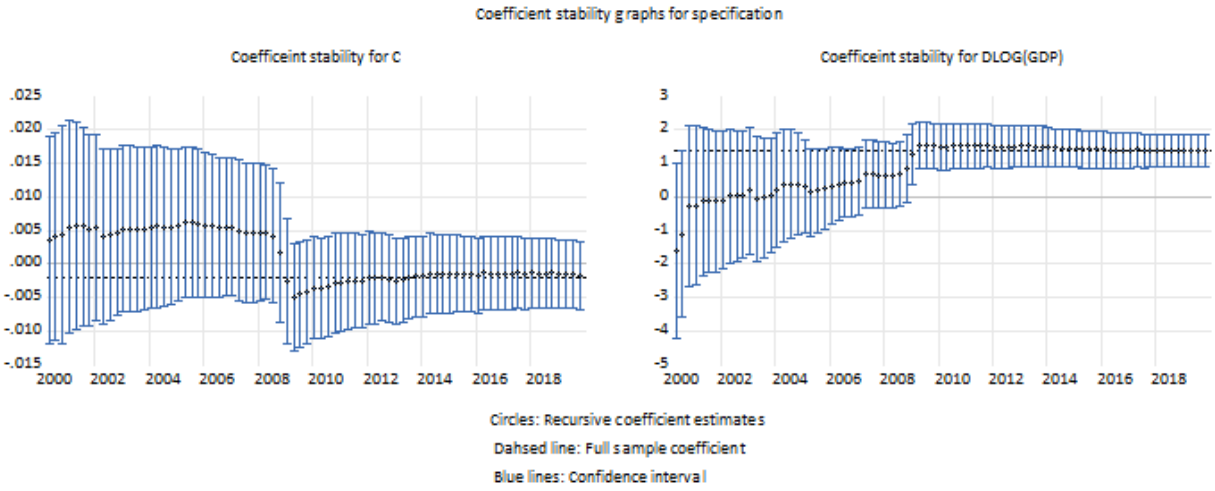
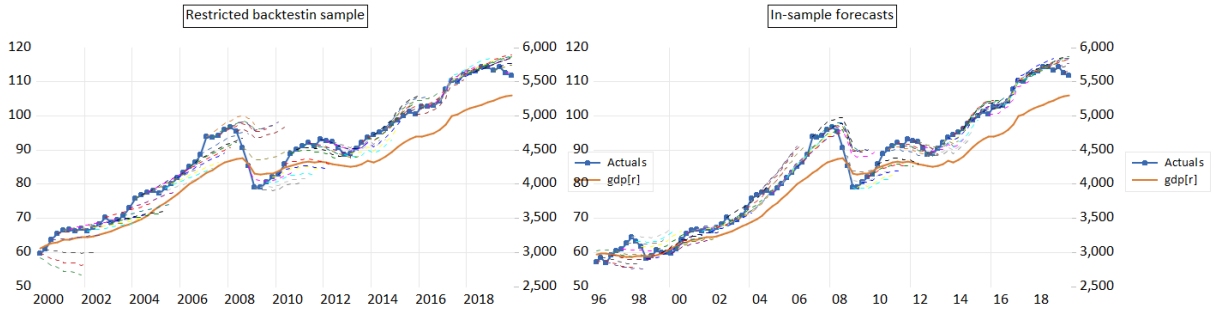


Figure 8: Forecast summary graph - static regression for industrial production (Adjusted)



model.

Table 5: RMSE - static regression for industrial production (in-sample)

	Forecast horizons (# of steps ahead)		
Specification	8	24	Avg.
EQ_IP_STATIC	4.36	5.55	4.96
EQ_IP_ARMA	6.70	9.25	7.98

Table 6: RMSE - ARMA models for industrial production (restricted sample)

	Forecast horizons (# of steps ahead)		
Specification	8	24	Avg.
EQ_IP_STATIC	6.79	14.5	10.6
EQ_IP_ARMA	8.18	13.8	11.0

Before proceeding to including additional variables, it is interesting to consider modification of

model in equation (1): modeler might consider dropping the constant from the model, since the constant allows the two series to grow at different growth rate. `SpecEval` add-in allows the modeler to quickly asses whether such model change is a good idea. Since the answer might differ substantially with forecast horizons, it is sensible to include more horizons than just the two considered above. Specifically, by issuing following command I include 6 different forecast horizons, while creating graphs just for 3 our of them:[7]

```
eq_ip_static.speceval(spec_list="eq_ip_static*", horizons_forecast="1 2 4 8
   16 40 80", horizons_graph="4 8 40", alias="with without")
```

Table 7 shows the RMSEs for different horizons for the two specifications. Note that the command above also included the aliases for the two specifications; like that one can easily distinguish which one is which in the output objects. The table shows that the specification with constant is slightly worse at forecasting at short and medium horizons, but better at very long horizons. In either case, the differences do not seem large.

Table 7: RMSE - static regression for industrial production (effect of constant)

	Forecast horizons (# of steps ahead)							
Specification	1	2	4	8	16	40	80	Avg.
with	1.56	2.77	4.84	8.22	13.7	37.7	103	24.6
without	1.51	2.59	4.38	7.44	13.5	39.9	104	24.7

The forecast summary graphs make it clear that while including GDP does help with forecasts during recessionary periods - especially when one considers in-sample forecasts, which are more relevant for understanding behavior of forecasts in future recessionary periods - the model is far from good enough. Moreover, the model has tendency to miss not only during the recession, but also during the recovery. This suggests that the forecast quality could be systematically improved by including additional information.

Figure 7 suggest one possible way to adjust the model. It is clear that there is a break in coefficient estimates at around Great Recession. One option to address this would be use model with break. An alternative that is probably more sensible is to allow the relationship between industrial production

[7] If 'horizons_graph' would not be specified then the graph horizons would correspond to the default value, which is equal to 'horizons_forecast'.

and GDP to be different during recessionary periods.[8] Consider then estimating following equation

$$dlog(IP_t) = \beta_0 + \beta_1 dlog(GDP_t) + \beta_2 D_t^{recession} dlog(GDP_t) \tag{2}$$

where $D_t^{recession}$ is a recession dummy indicator.[9] We want to compare this specification with equation (1). Since the motivation for the modification is the performance during the recessionary periods, it is useful to get detailed information about those. I will focus here on two such periods, 2008q3-2009q4, and 2011q3-2012q4, and specify them as the two subsamples via following command:

```
eq_ip_static.speceval(spec_list="eq_ip_static_dummy",
   subsamples="2008q3-2009q4,2011q3-2013q2", horizons_forecast="1 2 4
   8",oos="f", alias="normal dummy")
```

Specifying sub-samples has two consequences. First it creates additional tables with forecast performance metrics for given sub-samples, see tables 8 and 9.[10] Second, it creates graphs of single forecast that starts at beginning of the sub-sample; see Figure 9, top panels of which show forecast from the basic static regression, while bottom panels show forecast from the static regression with recession dummy. The tables (as well as the figure) make it clear that including dummy in the regression helps with the forecasting performance during the two recessionary periods. Meanwhile, the figure suggests that this is due to making the industrial production more sensitive to movements in GDP, something which can be confirmed by checking the coefficients in the regression outputs (not included): the inclusion of the interaction with regime dummy leads to lower coefficient on the stand-alone GDP term, but to higher combined coefficient. Investigating the table with overall RMSE (not included) shows that this is not at the cost of worse forecasting performance overall, since RMSE for whole sample is lower.[11]

While allowing industrial production to be more sensitive to GDP movements during recessionary periods helps with the forecasting performance during these periods, the forecast during Great Recession is still not satisfactory. Another approach is to consider including GDP components in

[8] The reason why this is more sensible is because one does not discard the observations before the break.

[9] The dummy is equal to one during following samples: 1996q4-1997q4, 1998q4-1999q1, 2008q4-2009q3, 2011q3-2013q1. Another alternative to using exogenously specify regime is estimating the regimes based on data (e.g. by threshold autoregression or markov-switching).

[10] Note that for forecast to be included in calculation of forecast performance metric, both its start and end have to be in within the sub-sample.

[11] The forecast evaluation here is done in-sample to highlight the use of the tools. The results for the first sub-sample, and for sample before 2008 in general, are reversed in out-of-sample forecasting performance. That said, model with dummy variable is better even out-of-sample for sample from 2010Q1 to 2019Q4.

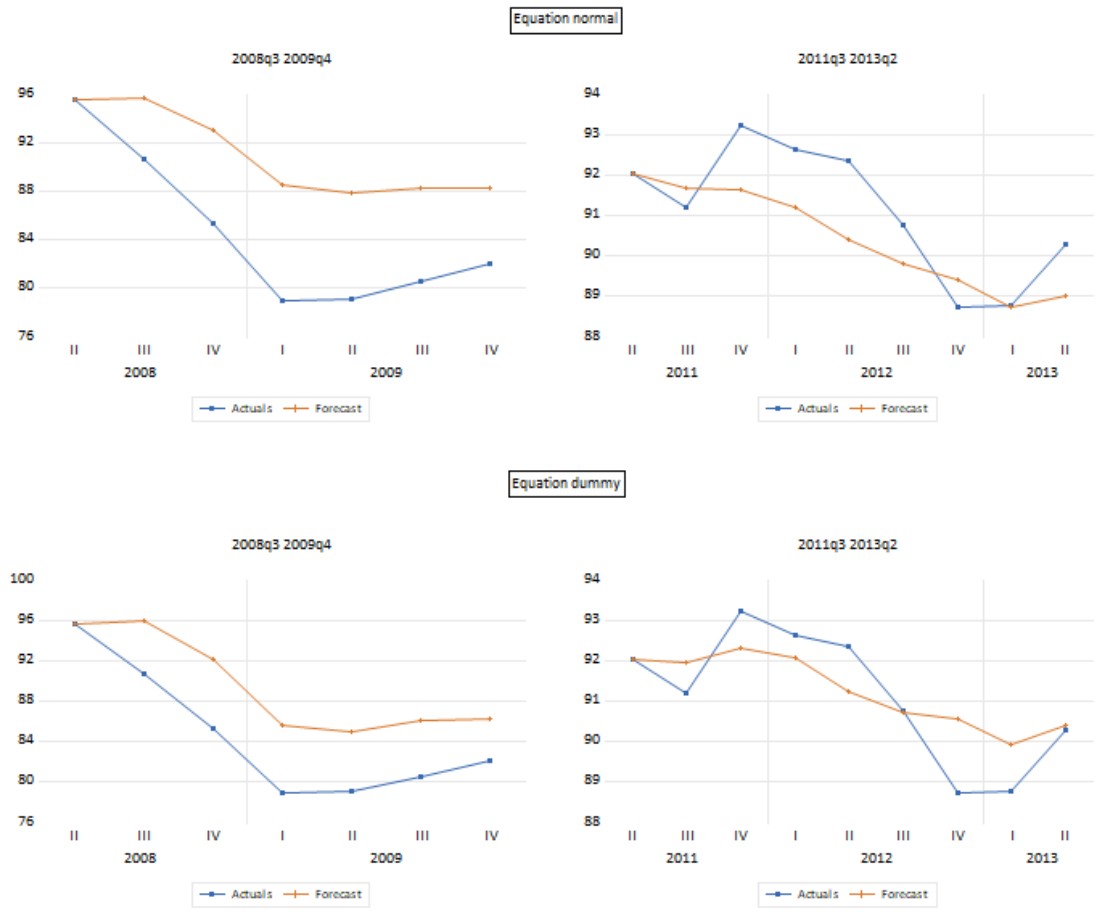
Table 8: RMSE - static regression for industrial production (2008q3-2009q4)

Specification	Forecast horizons (# of steps ahead) 1	2	4	8	Avg.
normal	2.65	4.39	5.40	NA	4.15
dummy	2.36	3.33	3.66	NA	3.12

Table 9: RMSE - static regression for industrial production (2011q3-2013q2)

Specification	Forecast horizons (# of steps ahead) 1	2	4	8	Avg.
normal	1.14	1.61	1.62	1.28	1.41
dummy	1.10	1.50	1.65	0.13	1.09

Figure 9: Sub-sample forecasts graph - static regression for industrial production



addition to GDP as such. Different recessions can be characterized different composition of decline in GDP, and some declines can be associated with more or less decline in industrial production. A prime example are exports: Czechia relies heavily on export of industrial goods, and hence industrial production should be sensitive to foreign demand. As such, including exports could help with forecasting performance, especially in periods of large movements in global demand for industrial goods, such as the Great Recession. Consider therefore estimating following equation

$$dlog(IP_t) = \beta_0 + \beta_1 dlog(GDP_t) + \beta_2 D_t^{recession} dlog(GDP_t) + \beta_3 dlog(Exports_t) \tag{3}$$

and then calling `SpecEval` as follows:

```
eq_ip_static.speceval(spec_list="eq_ip_static_dummy eq_ip_static_exports",
   subsamples="2008q3-2009q4,2011q3-2013q2", horizons_forecast="1 2 4 8",
   oos="f", alias="normal dummy exports", keep_forecasts="t")
```

Note that this time around I have specified that I want to keep the forecasts in the workfile so that I can investigate them together, rather than one specification at a time as in Figure 9. We are specifically interested in series *IP_F2008Q3_NORMAL*, *IP_F2008Q3_DUMMY* and *IP_F2008Q3_EXPORTS*, which contain forecasts starting in 2008q3 for each of the three specifications. The forecasts, together with actuals, are in Figure 10. It shows that including exports does help with the forecast, but not dramatically so. This suggests that while exports might be useful for for forecasting industrial production - indeed full sample and sub-sample RMSE again decreases - the decline during Great Recession goes above and beyond what one would conclude by looking at GDP and exports.

The model with exports already includes three regressors, making understanding forecasts a potentially complicated task. `SpecEval` add-in contains functionality that helps with this task, namely the forecast decomposition tool.[12] To perform forecast decomposition, just add it to the execution list as follows:

```
eq_ip_exports.speceval(exec_list="medium decomposition",
   subsamples="2008q3-2009q4,2011q3-2013q2", oos="t")
```

As result the spool will now include forecast decomposition graph for each sub-sample, such as Figure 11. The figure shows how each of the regressors contributes to the overall forecasts (in terms

[12] To be more precise, `SpecEval` calls separate add-in called `FcastDecomp`. This add-in has to be installed in order for forecast decomposition to be used in `SpecEval` . See associated documentation for more details about the forecast decomposition graphs.

Figure 10: Sub-sample forecasts graph - static regression for industrial production (multiple specifications)

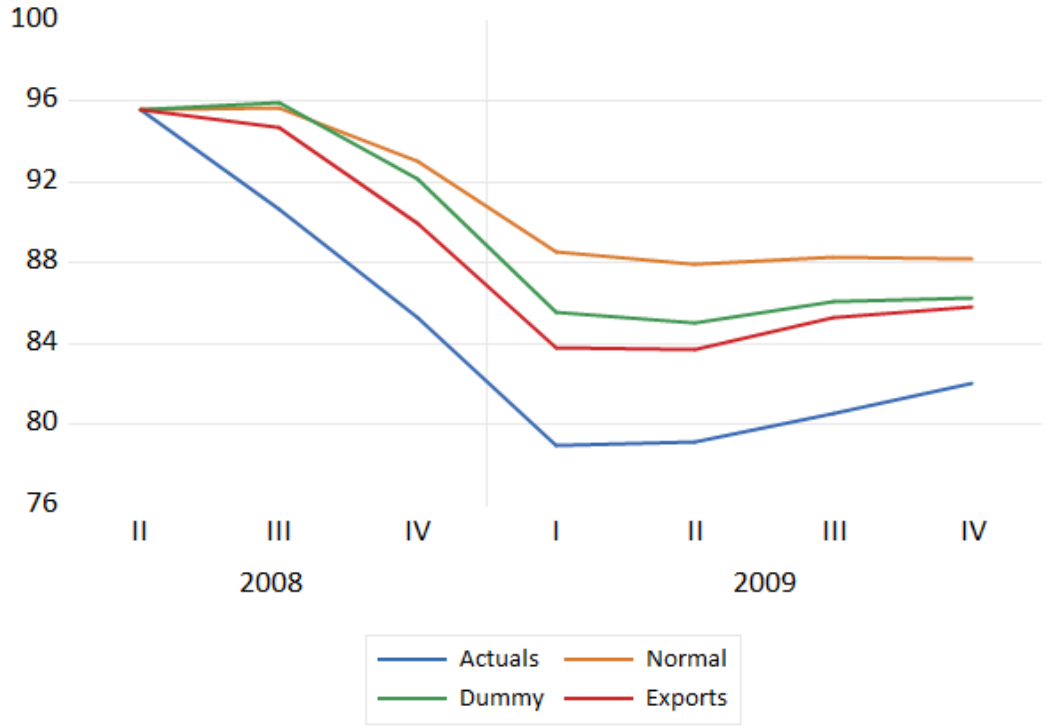
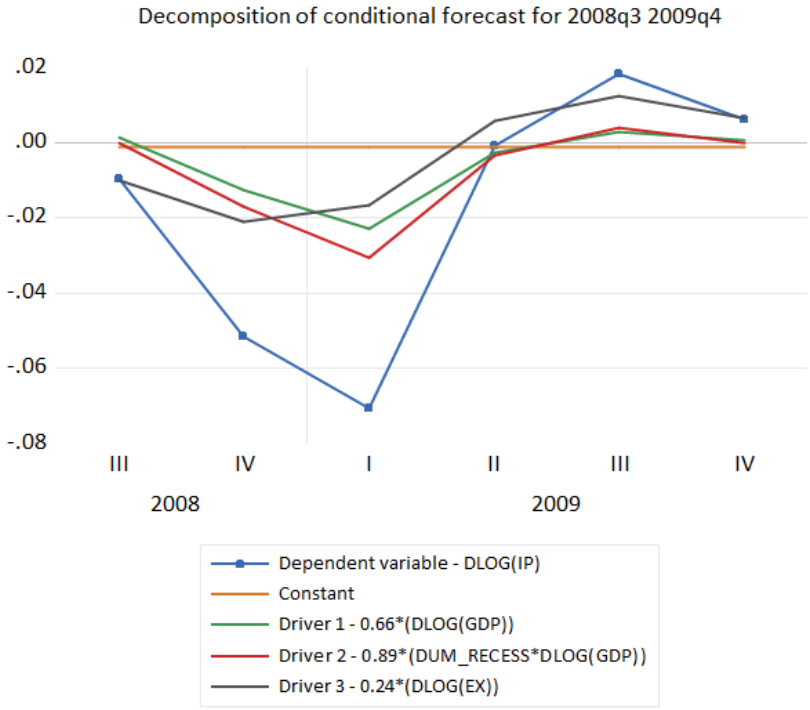


Figure 11: Sub-sample forecasts graph - static regression for industrial production (multiple specifications)



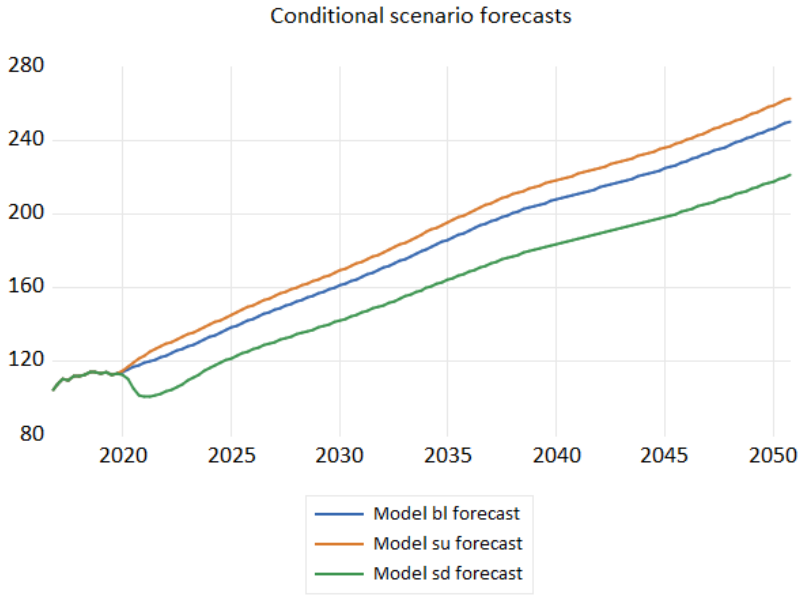
of the dependent variable) in each of the forecast periods. Among other things this graph can be used to figure out source of problematic forecast, or understand understand un-intuitive forecasts. Here, it shows how exports contribute to the overall decline in 2008q4 more than the other regressors, but less in 2009q1.

So far I have focused solely on overall forecasting performance, or on forecasting performance during particular historical period(s). In either case this amounts to analyzing the specifications in terms of their backtesting performance. The other focus of `SpecEval` is scenario forecast performance. While this will be in bigger focus in later application, it is worth introducing some basic aspects of it here, to highlight how scenario forecasting performance can be also useful as source of information when overall forecasting performance is the focus of model building. Following command creates forecasts for 3 scenarios - baseline, su (upside scenario) and sd (downside scenario)[13] - based on equation (2), and includes graph that contains all scenario forecasts together in the resulting spool, see Figure 12.

```
eq_ip_dummy.speceval(scenarios="bl su sd")
```

[13] The scenarios are taken from Moody's Analytics.

Figure 12: Scenario forecasts graph - static regression for industrial production

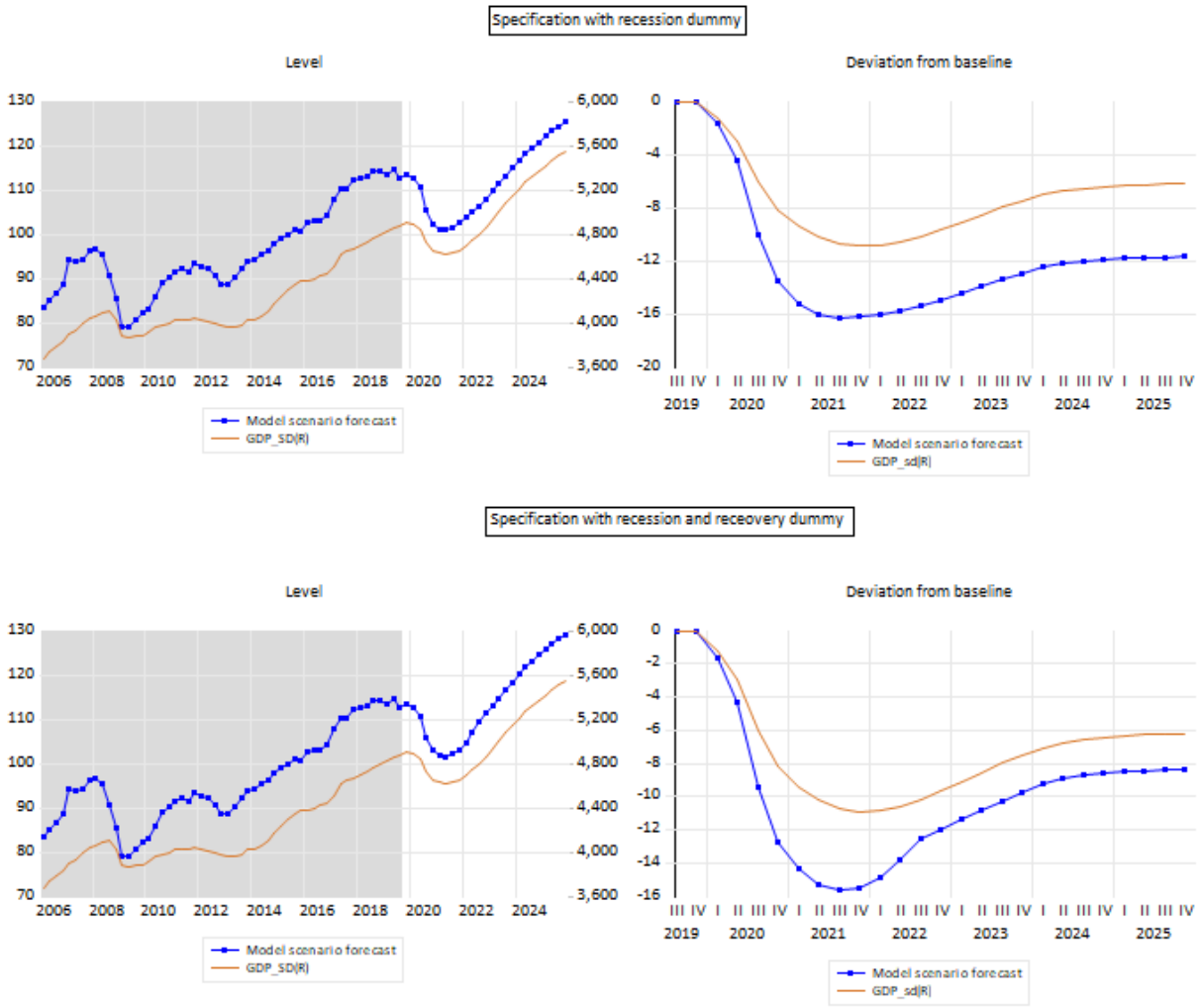


While containing the basic information, Figure 12 has some important drawbacks. First, the sample before the start of scenarios is too short to judge the magnitude of the movements, while the sample after the start of scenarios is too long. Second, it is hard to judge movements in industrial production without knowing the size of movements in GDP. We can address these issues by adjusting the starting period of the graph, ending period of scenario forecasts and including GDP as comparison variable, and then specifying that we want to produce individual scenario graphs rather than all scenario graphs. The resulting command is following:

```
eq_ip_dummy.speceval(exec_list="medium scenarios_individual",
  scenarios="bl su sd", tfirst_sgraph="2006q1", tlast_scenarios="2025q4",
  graph_add_scenarios="gdp[r]")
```

Now the resulting spool will include graph for each scenario individually, in additional to single graph with all scenarios. For example, top left panel of Figure 13 shows the individual scenario forecast graph for the downside scenario. Among other things the figure suggests that while the industrial production does fall more than GDP in beginning of the scenario, as desired based on previous analysis, the recovery seems to be too weak compared to GDP. This can be best seen if we use scenario transformation graph while using deviations from baseline as our transformation, as following command does:

Figure 13: Scenario forecasts graph - static regression for industrial production




```
eq_ip_dummy.speceval(exec_list="medium scenarios_individual",
   scenarios="bl su sd", tfirst_sgraph="2006q1", tlast_scenarios="2025q4",
   graph_add_scenarios="gdp[r]", trans="deviation")
```

The desired chart is shown in top right panel of Figure 13. The model results into industrial production falling permanently and substantially behind the GDP, which is likely not reasonable. Instead, one would expect both the drop and rebound in industrial production to be larger so that the permanent effect on industrial production is only slightly larger than for GDP. The reason why the model fails to make such forecast is because it makes industrial production more sensitive movements in GDP only during recessions, not during recoveries. One simple way to address this is to replace the

dummy indicating recession by dummy that is equal to 1 also 4 quarters after the end of recessions.[14] The downside scenario forecasts for the resulting specification are in bottom panels of Figure 13. The graphs make it clear that adjusting the model in such way addresses the problems raised above. Moreover, Table 10 shows that this modification motivated by improving scenario forecasts also leads to improvement in overall forecasting performance. This then constitutes an example of how analysis scenario forecasting can be useful in improving overall forecasting performance, irrespective of whether scenario forecasting is of importance or not.

Table 10: RMSE - static regression for industrial production (different dummy variables)

	Forecast horizons (# of steps ahead)		
Specification	8	24	Avg.
short_dummy	3.94	4.46	4.20
long_dummy	3.91	4.04	3.97

3.2 Application with growth rate transformation

Previous application showed most of the core functionality of `SpecEval` , and how it can be used in interactive iterative model building process. The application presented in this section will be more limited and focused than the previous one. The primary goal will be to show how `SpecEval` can be used for situations when it is not the level of underlying variable that is of focus, but rather different transformation, and specifically the growth rate of underlying variable. Consider the example of consumer price index (CPI). While the level of prices is often of interest, it is also common for modelers to be interested in inflation, i.e. growth rate of CPI.[15] Apart from this primary goal, the first part of this application will also show how to use recursive automatic model selection functionality of `SpecEval` .

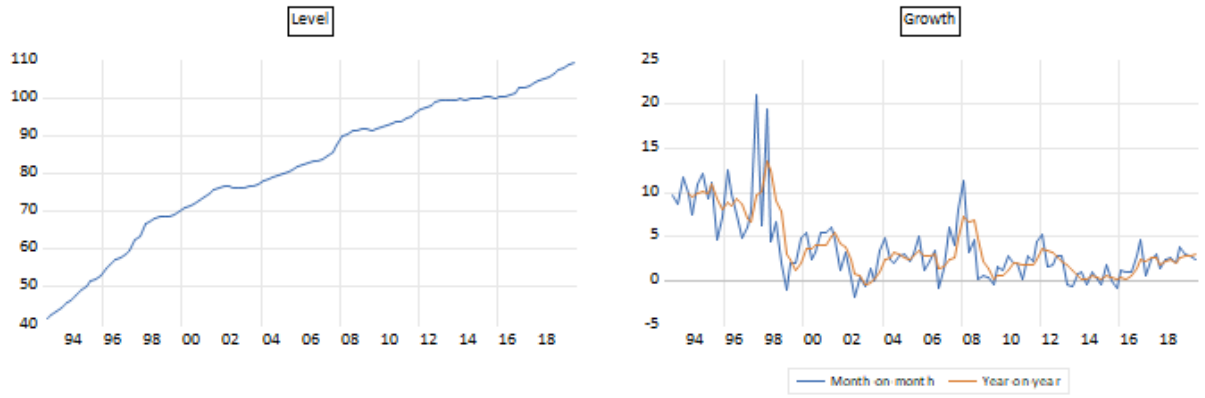
Figure 14 shows the level and growth rate of Czechia CPI, which will be our focus in the first part of this application. Given that the series is clearly trending but not trend-stationary then it is natural to model the series in terms of growth rate. Since the growth rate itself will be of focus, I will use the

[14] One simple way to do this in Eviews is to replace $D_t^{recession}$ by $(@movav(D_t^{recession}, 4) > 0)$, i.e. indicator whether 4-quarter moving average of recession dummy is positive.

[15] While it might seem that it is almost always the case the inflation rather than level of CPI is of interest, this is not necessarily the case. If inflation is directly or indirectly used in later forecasting, then what should matter for overall forecasts is the inflation over the whole forecasting period that should matter, not the exact timing of high/low periods of inflation. This would correspond to cumulative rise/decline in CPI. In contrast, focusing on inflation as the main variable can often lead to wrong conclusions because the model in consideration does not produce movements in inflation with exactly correct timing, despite producing correct forecasts for the cumulative rise/decline in CPI. In such case, model which will produce less volatile forecast for inflation, but will lead to wrong forecasts for CPI, can be wrongly preferred.

annualized quarterly growth rate as dependent variable rather than the log-difference as in application 1.[16]

Figure 14: Czechia consumer price index



Typical starting point of modeling inflation is model of from the ARMA family. As in application 1 I will rely on the automatic ARMA model selection to select the specific type of ARMA model used for CPI. However, in contrast to application 1 I will make this selection recursively rather than establishing it ex ante: notice that in application 1 the model used for backtesting was selected based on the full data sample, rendering the the model *structure* to be in-sample rather than out-of-sample in nature (while actual model *coefficients* were out-of-sample in nature). `SpecEval` includes functionality to allow user to determine model structure recursively always on data available at the start of the forecast and hence to make proper out-of-sample forecasting.[17]

To use this functionality one needs include information about the automatic model selection that should be performed. Consider first using automatic ARMA model section for annualized growth rate of CPI based on whole sample:[18]

[16] Note that I will use *quarter-on-quarter* growth rate rather than the year-on-year growth rate, which is more common measure of inflation. The reason is because using year-over-year growth rate as dependent variable introduces artificial seasonality into forecast for level of CPI. While in application when only year-over-year inflation is ever used this does not constitute a problem, it causes problems when level CPI is also used.

[17] Note that for some estimation methods the automatic model selection is what will happen irrespective of `SpecEval` settings. For example, for `ARDL` estimation command Eviews will automatically select the model whenever the equation is re-estimated and hence also during out-of-sample backtesting. That said, the `SpecEval` functionality has additional advantages in such situation, namely the automatic removal of perfect colinearity issues. For example, if an equation contains constant and some independent variable has zero variance in some shortened estimation sample then the constant and the variable are perfectly colinear, preventing estimation to proceed. In such situation the `SpecEval` will remove the constant, allowing estimation to proceed. However, for equations estimated via `ARDL` command this happends only if automatic selection option is specified. Apart from this feature, using the in-build model selection also has the advantage of allowing the user to leverage the coefficient and model stability functionality, see discussion below.

[18] Notice that Eviews does not currently support using automatic ARMA model selection for series with transformation. Hence one needs to first create the transformed series and perform automatic ARMA model selection on the transformed series. Once the selection was performed the dependent variable in the equation needs to re-specified in implicit terms.

```
series s_cpi_growth = @pca(cpi)
```

```
s_cpi_growth.autoarma(tform=none, diff=0, maxar=4,maxma=4, select=aic,
   eqname=eq_cpi_arma) s_cpi_growth_f c
```

```
equation eq_cpi_arma.LS @pca(CPI) C AR(1 to 3) MA(1 to 4)
```

The resulting selected model is ARMA(3,4). However, instead of using ARMA(3,4) model for the backtesting exercise - and hence imposing the model structure based on full data sample - the modeler wants to select the bast ARMA model according to the criteria based on the data available at the start of each forecast. For this he/she needs to specify the selection information, namely the maximum number of AR terms, maximum number of MA terms and the selection criterion. To do this, he/she needs to add attribute to the equation with name 'selection_info' and include the terms 'maxar', 'maxma' and 'info'. In the example above this is dones as follows:

```
eq_cpi_arma.setattr("selection_info") maxar=4,maxma=4,info=aic
```

Once the equation includes this information one use `SpecEval` to perform recursive automatic model selection:

```
eq_cpi_arma.speceval(auto_select="t", exec_list="medium stability")
```

As a result, the usual spool will be created. Before looking into the forecasting performance, it is useful to highlight that when the 'stability' execution option is specified, the spool will also include graph with recursive ARMA orders as well as the recursive coefficient. The latter is displayed in Figure 15 while the former is in Figure 16.[19] The key value of Figure 16 is that it succinctly summarizes the model structure that would be selected at each point in time. This is useful for two reasons. First, it allows the model builder to ascertain the stability of the selected model order over time, much like it is commonly done for coefficient estimates. Second, it allows the model builder to spot shifts in the model structure and possibly link them to changes in the underlying data.[20]

So far I have focused on the secondary goal of this subsection; it is now time to turn to the primary goal of transformation graphing. Figure 17 shows the forecast summary graph for 8-quarter horizon. The key conclusion from the figure is that it is not extremely informative, especially if inflation is

[19] Notice that the recursive coefficient series now can include gaps, since particualr ARMA term does not have to be included at for given estimation sample.

[20] See **?** for example of such interpretation of shift in model structure.

Figure 15: Recursive coefficients

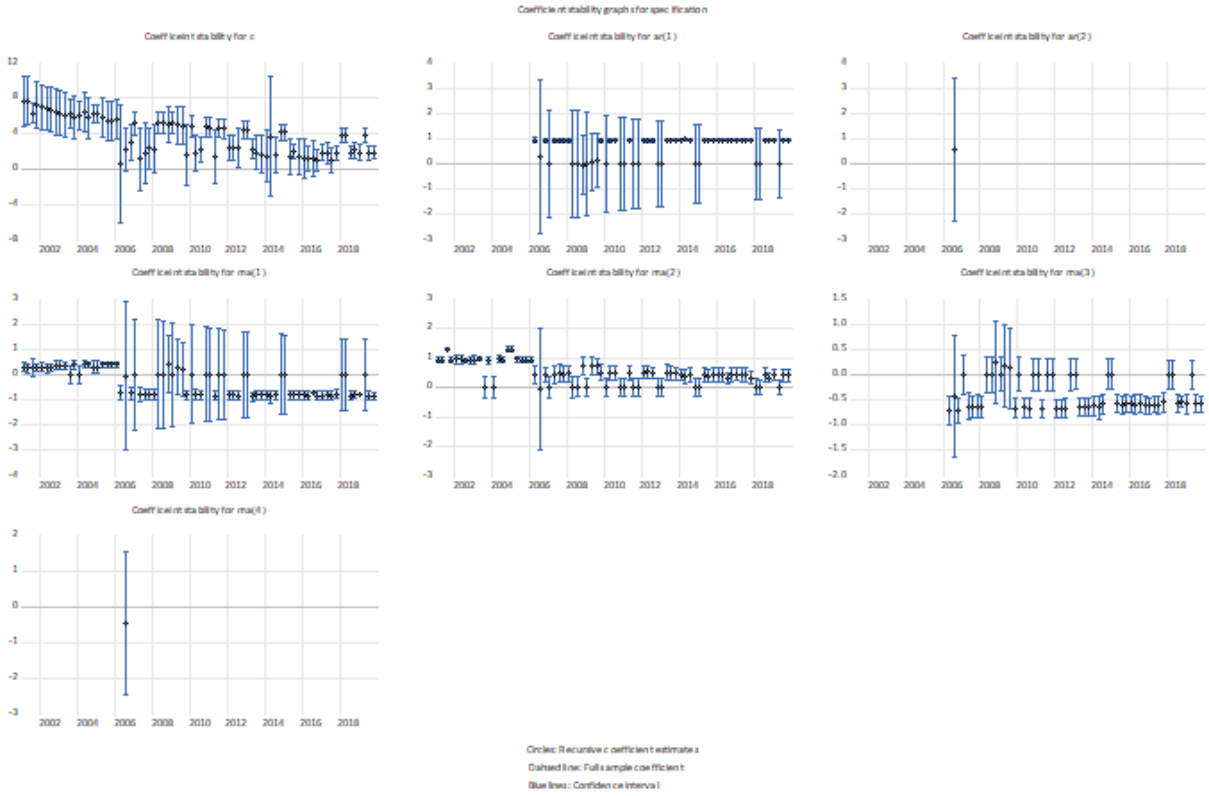
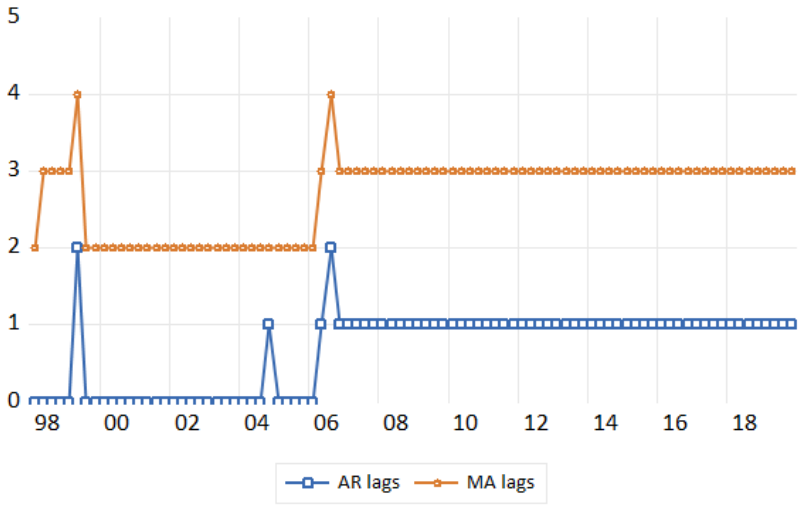
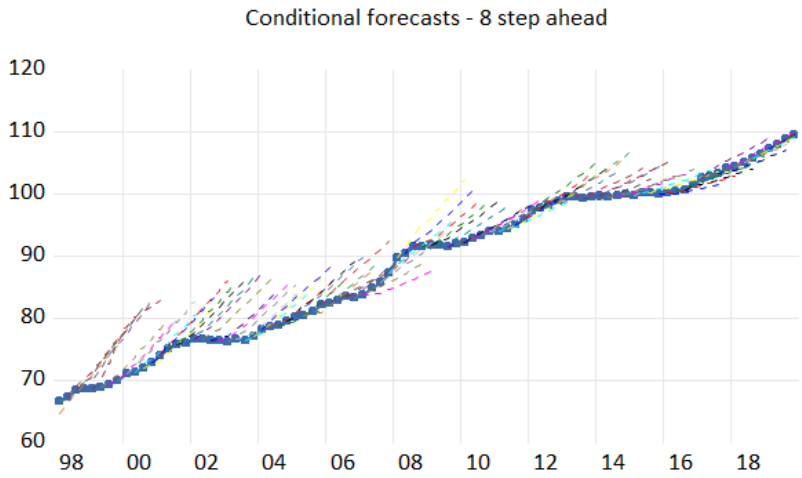


Figure 16: Recursive ARMA lags



of interest: there is relatively little variation in the forecast for inflation produced by the model and hence the forecasts for *level* of CPI are very close to straight lines starting from the last pre-forecast value. While this on its own is useful information as it suggests that the model should include more amplification and/or persistence, model developer would likely be more interested in graphs that would be in terms of inflation itself. To do that one just needs to specify that different transformation for graphs should be used as follows:

Figure 17: Recursive ARMA lags



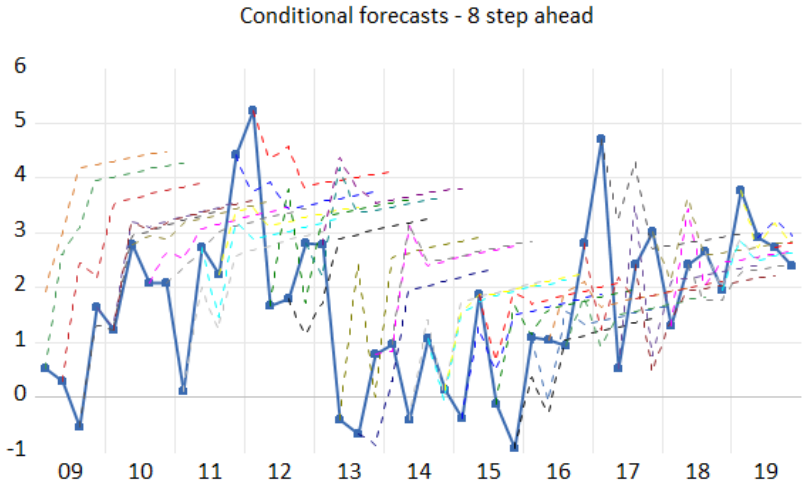

```
eq_cpi_arma.speceval(auto_select="t", trans="growth", tfirst_graph="2009Q1")
```

Note that the command has restricted the sample to start in 2009Q1 to limit the number of observations in the graph to make it legible, and to focus on the period after Great recession.[21] The resulting forecast summary graph is in Figure 18. There are two key takeaways from the Figure, both that were already visible in Figure 17. First, the model shows too little persistence, with inflation jumping in the first two quarters of forecast. Second, the inflation rises to too high values, reaching 4% in the first half of the displayed sample. This has manifested in previous figure by the fact that almost all the forecasts had upward bias: forecasted level of CPI was higher than the actual future values. Of course this reflects the fact that inflation rate was much higher in earlier part of the sample, which influences the model estimate of constant and hence inflation rate.

decomposition

[21] This affects only the graphs ample, not the backtesting sample.

Figure 18: Recursive ARMA lags



3.3 Application with spread transformation

The previous application considered the case when specific transformation is used because specific transformation is of interest. This and next application consider a different motivation for use of transformation: situation, when graphs only in specific transformation provide useful information for developing forecasting model. Specifically, in many situations majority of variation in variable of interest originates not in the shocks to the variable, but reflect variations in some independent variable. A typical example are market interest rates: all market interest rates can be theoretically decomposed into current risk free interest rates, expectations of future risk free interest rates and (compensation for) risks associated with market lending. Out of these factors, it is the risk free interest rates which vary the most at medium to long horizons.[22]

This means two things. First, in conditional forecasting the performance of multivariate models that include risk free rates as independent variable will always look good in *terms of level*. Hence, the proper way to evaluate such models is by eliminating the effect of risk free rates by using spread transformation.[23] Second, in unconditional forecasting majority of forecast errors will originate in forecasts for the risk free rates. Correspondingly, the overall size of forecast errors will not be very informative about the model of interest; and the relative value of forecast errors will reflect not only how

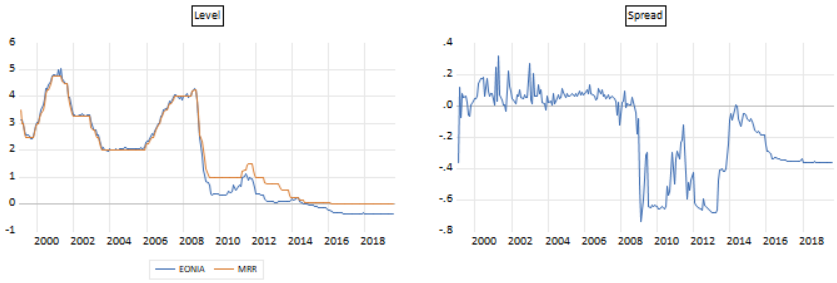
[22] Of course, this is different for different horizons. For example, at short horizons most of the variation will come from the risk factors.

[23] Of course, this applies to qualitative (graphical) analysis. In quantitative analysis the good performance will result in low error metrics for all models, so that relatively to each other one will still be able to select better performing models.

large are the errors originating in the model of interest, but also how are they correlated with errors originating in the model for risk free rates. It can easily happen, that model that is conditionally wrong will be selected because *in given evaluation sample* its errors are negatively correlated with errors from model for risk free rates. For these reasons, using spread transformation is useful in such situation for the development of forecasting models, irrespective of whether the spread transformation is of interest itself or not. This application will focus on the first aspect; I will return to the second aspect in application 5.

Eonia rate. This application will use Eonia rate as the variable of interest.[24] Figure 19 shows the Eonia rate, both in terms of level and in terms of spread from the main refinancing rate (MRR) set by the ECB. The figure suggest natural starting point for modeling the Eonia rate: static regression linking Eonia rate to MRR:

Figure 19: Eonia rate

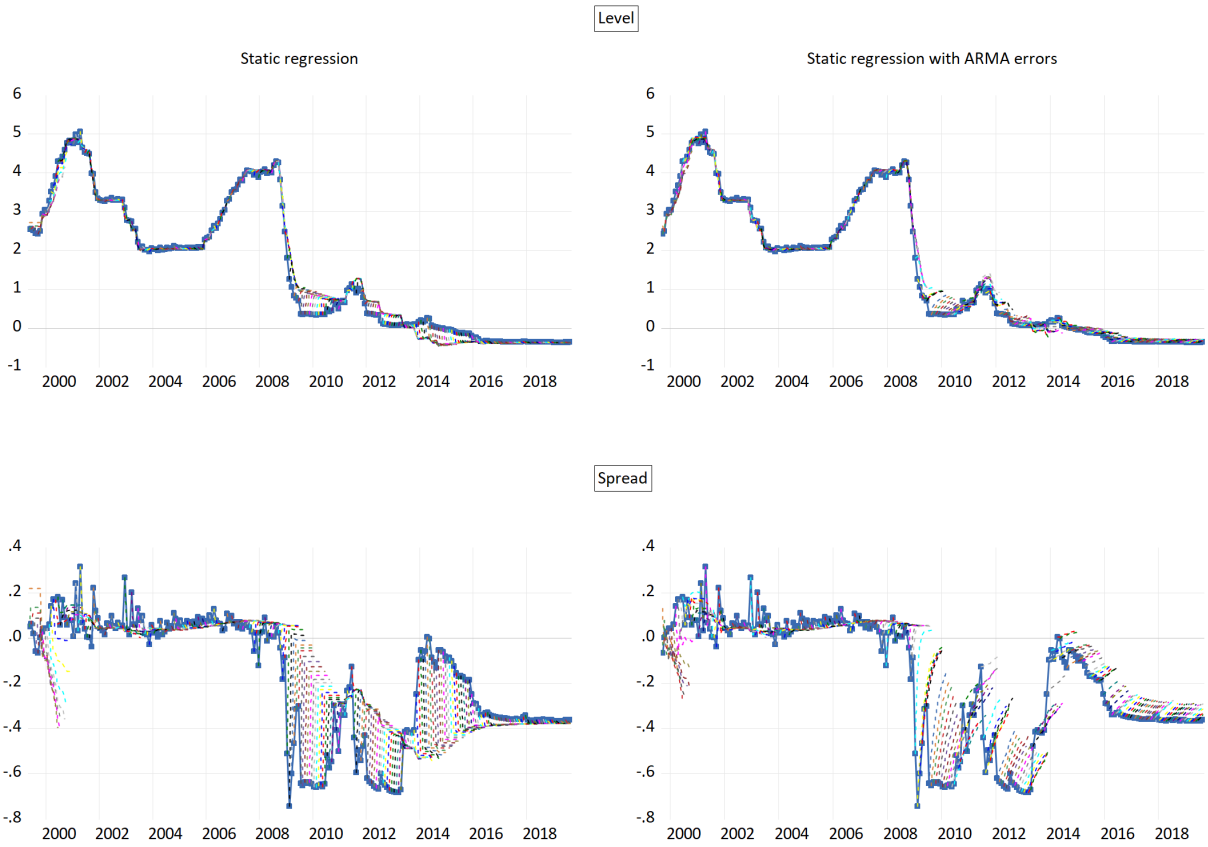


$$Eonia_t = \beta_0 + \beta_1 MRR_t \tag{4}$$

Estimating such equation running thespeceval add-in on it leads to forecast summary graph displayed in top left panel of Figure 20. The figure seems to suggest that the model leads to good forecasting performance, with forecasts close to the actual values for most of the sample. However, one might be concerned about the few periods when forecasts diverge from actual values given that in those periods the model predicts large jump in Eonia rate in begining of forecast. Natural way to address this would be by including ARMA errors in the model. Top right panel of Figure 20 shows the resulting forecasts. The results are better, with fewer periods of deviation and with the deviations appearing more gradually.

[24] This application is taken from **?**. See that paper and its companion paper for more details.

Figure 20: Eonia rate forecasts



However, the bottom panels of Figure 20 show that once one looks at the spread transformation then forecasting performance of neither model is good: both models have tendency to forecast Eonia rate to increase from the current values to historically more normal values, while in reality this forecast was almost never borne out. To obtain this graph, one needs to specify different transformation and also indicate what series to use as transfromation benchmark[25]

```
eq_eonia_arma.speceval(trans="spread", graph_bench="mrr")
```

One could object that if the modeller cares about Eonia rate not the spread from MRR - which, however, often is not the case - then the model still performs well enough. However, such argument misses the key point illustrated by Figure 20. The reason why forecasts in terms of level look good is because most variation in level of Eonia rate comes from variation in MRR. Hence using model that links Eonia rate to MRR guarantees that the forecasts for level of Eonia rate will look good in terms of level of Eonia rate. A bigger question than whether one is able to forecast level of Eonia rate conditionally on MRR is whether one can forecast the residual variation after accounting for MRR. And this is something the `SpecEval` functionality is crucial for: by eliminating the main source of variation in Eonia rate, the spread transformation allows him/her to visualizing the forecasting performance on top of that variation. In some sense, the (graphical) evaluation of the forecasting performance is hindered by the presence of dominant source of variation and only the functionality of `SpecEval` is suitable to address this problem.

It would seem that the above arguments are really relevant only if one uses graphical evaluation of forecast performance, and as long as one focuses only on performance metrics then the above problem does not arise. While this is true, it means that one has to give up the value of graphical forecast performance evaluation, with the ability to identify missing factors being the main one. For example, the bottom panels of Figure 20 suggest 4 periods when spread between Eonia rate and MRR changed dramatically while the model was not able to explain that, as evidenced by large forecast errors: period right after the Global Financial Crisis, when the spread dropped; period around 2011-2012, when spread first increased and then decreased; period during 2013, when the spread recorded two jumps; and finally period from 2015 until 2016, when spread gradually decreased. A good forecasting model would be able to explain those periods and the shifts that happen during them.

A natural way to go about building such model would be to see which variables could help with

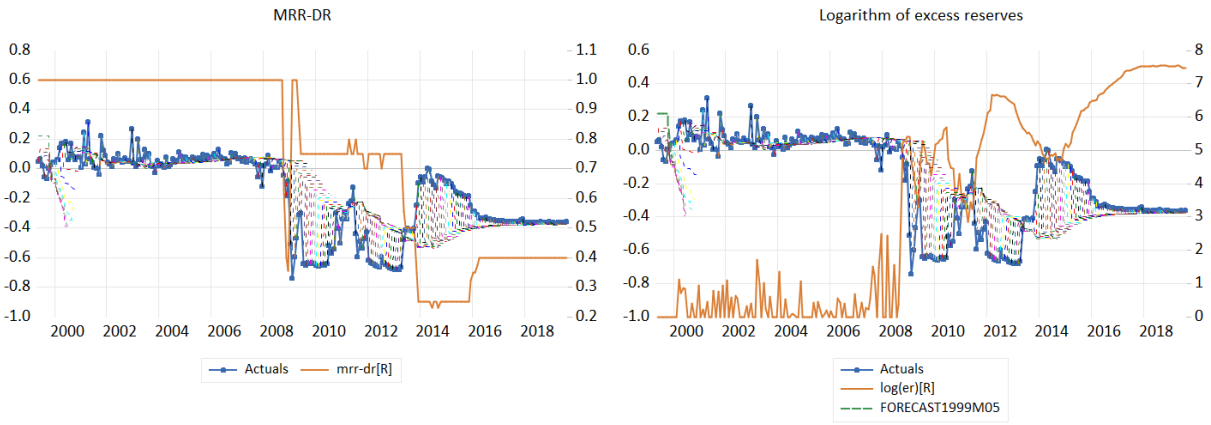
[25] If no transformation benchmark is used then user dialog in which user will specify it is issued.

improving the forecasting performance during those periods. `SpecEval` allows modeler to include additional variables in the forecast summary graph, and Figure 21 does just that.[26] The left panel shows the forecast summary graph for equation (4), but this time includes the spread between MRR and deposit rate as additional series. Meanwhile, the right panel shows the same graph, but this time includes logarithm of excess reserves. The graphs were created using following commands[27]

```
eq_eonia_arma.speceval(trans="spread", graph_bench="mrr",
   graph_add_backtest="mrr-dr[R]")
```

```
eq_eonia_arma.speceval(trans="spread",graph_bench="mrr",
   graph_add_backtest="log(er)[R]")
```

Figure 21: Eonia rate forecasts - Additional variables



The left panel show that the two-step increase during 2013 (as well as part of the decrease during the 2015-2016 period) can be explained by the change in spread between MRR and DR. **?** explains that this is due to the fact that Eonia rate underwent change in regime in 2008 and that since then it is anchored by deposit rate rather than main refinancing rate. The right panel then shows that the periods around 2011-2012 and from 2015-2016 are periods when excess reserves were either increasing

[26] Of course, one has to know which variables are possible candidates. The point here is not the `SpecEval` solve this problem, but rather that the ease of including additional series facilities exploring whether such particular variable is a good candidate or not; and in broader terms, that evaluating forecast performance in terms of graphs is important tool in model building.

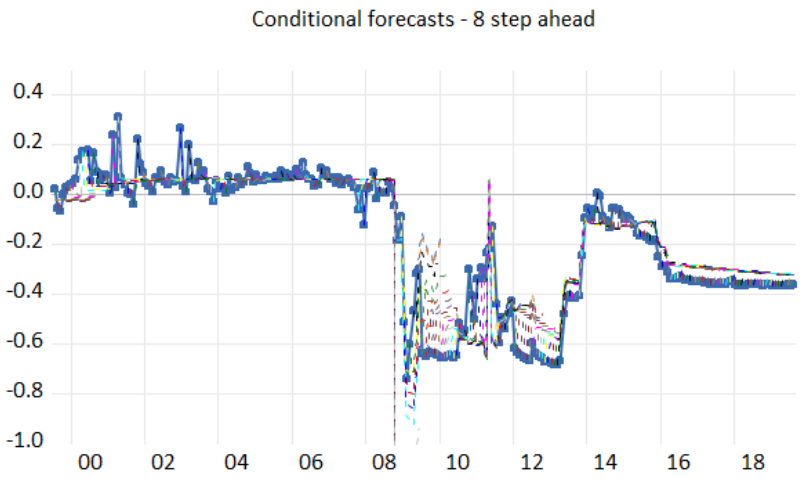
[27] The `[R]` in `graph_add_bactest` option controls the assignment of axis to secondary (right) axis.

or decreasing. **?** explains that increase in excess reserves leads to decrease in spread between Eonia rate and deposit rate, which then explains the forecast misses in the periods when excess reserves were different from their previous values. As a result of this analysis, **?** explains that the appropriate model for Eonia rate is following structural model that includes not only MRR, but also DR and excess reserves:

$$IIR_t = \begin{cases} \beta_{10} + MRR_t + \epsilon_t & \text{if } D_t^{ER} = 0 \\ \beta_{20} + \beta_{21} DR_t + \beta_{22} log(ER_t) + \epsilon_t & \text{if } D_t^{ER} = 1 \end{cases} \tag{5}$$

where D_t^{ER} is dummy indicating presence of excess reserves. Figure 22 show the forecasts from the resulting model, while Table **??** shows the RMSE of the the static regressions, static regression with ARMA errors and the structural model. ...

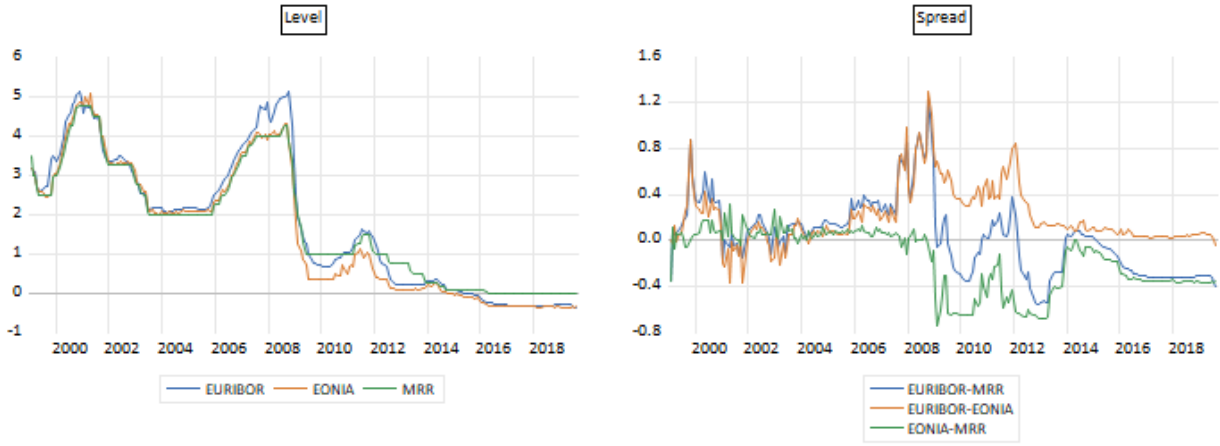
Figure 22: Eonia rate forecasts - Structural model



Euribor. The previous application used Eonia rate to demonstrate the value of transformation graphing in case of forecast summary graphs. Now I turn to the value of transformation graphing in subsample forecast graphs and scenario graphs. To fully appreciate the `SpecEval` functionality I will use Euribor series, rather than the Eonia series as object of my analysis. Figure 23 displays the level of 3-month Euribor together with Eonia rate and main refinancing rate, both in terms of level (left panel) and in terms of spread between Euribor and either of the two other rates (right panel). The figure highlights the reason why for this application I chose the Euribor rather than the Eonia rate: while the two rates feature the same medium-term movements which which were above linked to

excess reserves, the Euribor also features large fluctuations at short-term horizon in periods of financial stress, namely the Global Financial Crisis of 2007-2009 and Sovereign Debt Crisis of 2011-2012. The former movements are seen in the fact that broadly speaking the spreads between either of the two interbank rates and the MRR follow each other, so that the spread between the two interbank interest rates is more stable than the spreads from the MRR. The latter can be seen in the fact that during the two mentioned periods Euribor rises well above the Eonia rate. This of course reflects that fact that Euribor has longer maturity and hence includes substantially larger risk components.[28]

Figure 23: Euribor



Given the analysis above, I will study use model linking the Euribor to the Eonia rate. The starting point again is static regression linking those series.

$$Euribor_t = \beta_0 + \beta_1 Eonia_t \tag{6}$$

To analyse this model I will directly focus on the spread transformation, this time using the Eonia rate as the benchmark variable in the spread:

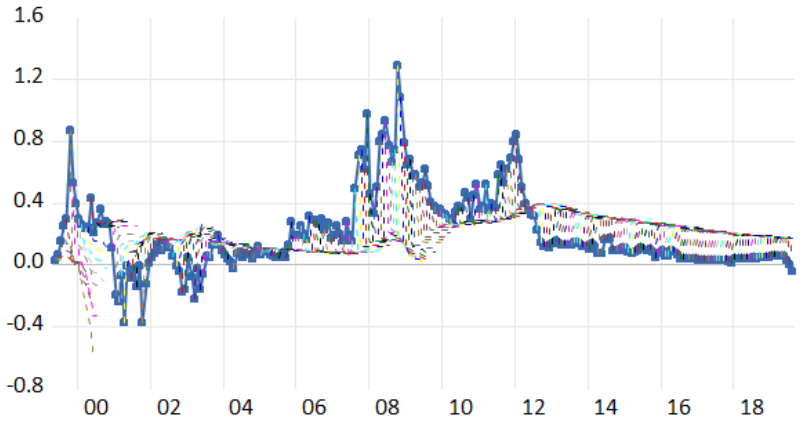
```
eq_euribor_static.speceval(nodialog,trans="spread", graph_bench="eonia")
```

The resulting forecast summary graph in Figure 24. The figure shows two related problems with the forecasts. The model predicts immediate drop from elevated value during the periods of financial

[28] One can also see the expectations component in the figure: for example, during 2005-2006, policy rates were increasing, and were also expected to further increase, so that the 3-month Euribor was substantially above the Eonia rate. I will ignore this factor here for simplicity.

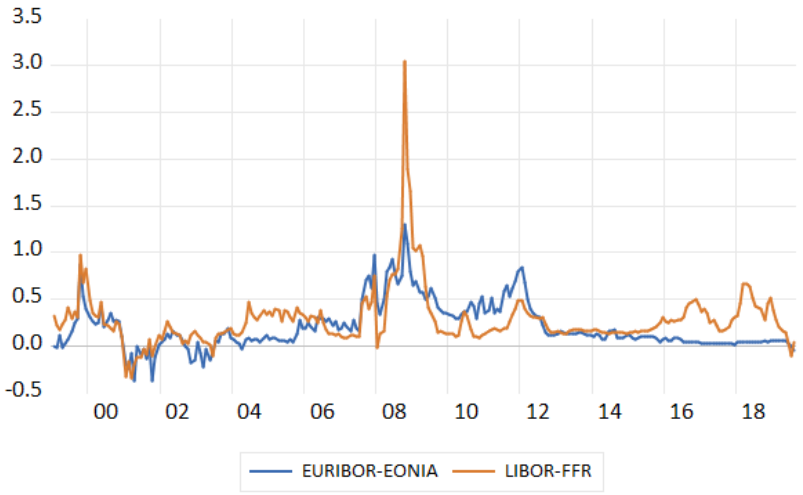
stress; and after the financial stress passes it predicts values that are much higher then the observed values. Both reflect the fact that the model does not include any stress drivers.

Figure 24: Euribor forecasts (static regression)



To logical way to improve the model is by including some stress drivers. Here I will consider a simple approach under which the stress component of Euribor is proxied by the proxy for global interbank market spread, namely the spread between US dollar Libor rate and the US risk free rate given by the federal funds rate (FFR).[29] Figure **??** displays the two spreads together.

Figure 25: Euribor and Libor spreads



First question when developing multivariate single equation model linking Euribor spread to Libor

[29] Of course, this proxy is far from perfect due to maturity mismatch. That said, the possible alternatives either come with serious data limitation, such as the Libor-OIS spread, or feature often more, not less undesirable fluctuations, such as the TED spread. In any case, the depdent variable also includes maturity mismatch.

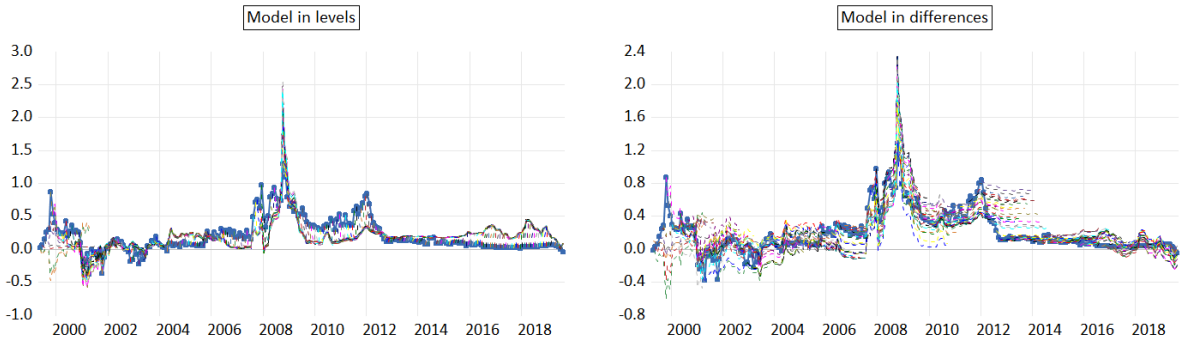
spread is what transformation should one use in such model: should level of Euribor spread depend on level of Libor spread, or should changes in Euribor spread depend on changes in Libor spread? The choice is not obvious, since each approach has its drawbacks, which can be illustrated by the `SpecEval` add-in. Therefore, I consider both options and estimate following two models:

$$Euribor_t - Eonia_t = \beta_0 + \beta_1 Libor_t - FFR_t$$

$$d(Euribor_t - Eonia_t) = \beta_1 d(Libor_t - FFR_t)$$

Figure 26 shows the forecast summary graph for both models, displaying only the spread transformation. One can see the drawbacks of each model. If level transformation is used then the secular variations in Libor spread will influence the Euribor spread, such as in last several years of the sample, when US interbank market was undergoing structural changes. On the other hand, if differences are used then the long-run forecast for Euribor spread will possibly not be stable, such as in 2012.

Figure 26: Euribor forecasts (spread regressions)



The figure captures the overall forecast performance. However, it is likely that the focus for model that would include stress driver is the periods of financial stress, such as the Global Financial Crisis and the Eurozone sovereign debt crisis. To study that, one needs to restrict the sample in some way so that there is only limited number of forecasts. `SpecEval` offers two approaches. One approach is to restrict the sample of the forecast summary graph, say to sample starting in 2007M01 and ending in 2012M12. The second approach taken here is to specify additional subsamples in addition to the main sample of analysis. To include two subsamples, one covering 2008M08-2008M12 and one covering 2011M09-2011M12, one just needs to use following commands:

```
eq_spread_level.speceval(trans="spread", graph_bench="eonia",
   subsamples="2008M09-2009M09,2011M10-2012M06")
```

```
eq_spread_diff.speceval(trans="spread", graph_bench="eonia",
   subsamples="2008M09-2009M09,2011M10-2012M06")
```

As a result of these commands the spool with results will contain graph capturing single forecast starting in beginning of the subsample and ending at the end; see Figure 27 for the subsample graphs for each subsample and each model. The figure shows that while there is not much difference between the two models, it is the model in differences for which forecasts are closer to the actuals during the Global Financial Crisis sample. As for the Eurozone sovereign debt crisis, neither of the models provides good approximation of the rise and drop in the Euribor spread, reflecting the fact that the fact that the stress in interbank markets was asymmetric, with higher stress in Eurozone than in US.

Figure 27: Euribor forecasts (subsamples)

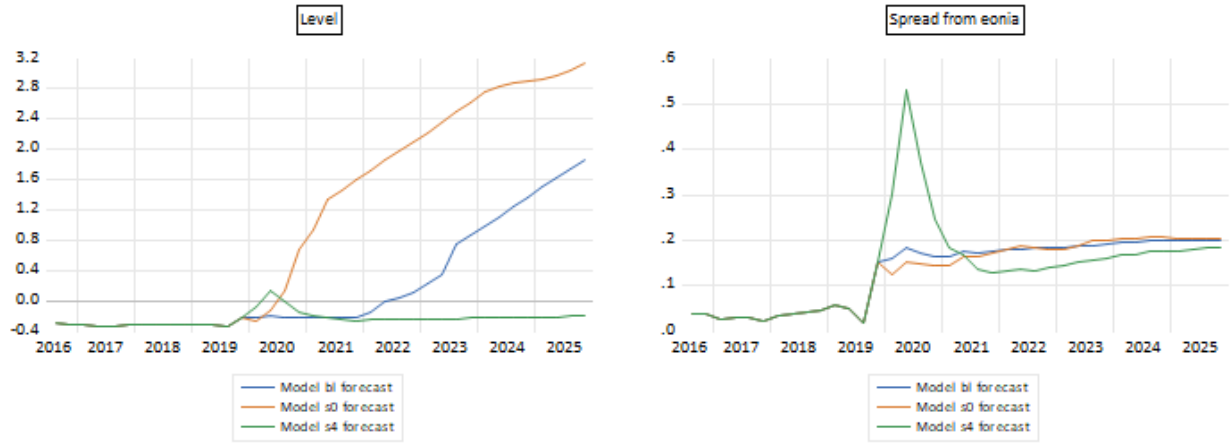
The second goal of this application is to highlight the value of transformation graphs in the analysis of scenario forecasts. For this purpose it is useful to shift focus to an alternative model that seems to avoid issues of both previous models, an error-correction mode:

This model is in terms of first differences, but also anchors the level of the forecast, so seems to offer a way to address both the issues raised above. To obtain the conditional scenario forecasts one needs to adjust the command as follows:

```
eq_spread_level.speceval(trans="spread", graph_bench="eonia", scenarios="bl
   su sd")
```

As a result the spool will contain conditional scenario forecasts graphs. Figure 28 displays the scenario forecasts for three scenarios: baseline, upside and downside. The left panel of the figure shows the level of Euribor rates in those three scenarios. While the graph is useful in so far as visualizing the actual forecasts, it does not provide great value in terms of evaluating the model for Euribor itself. The evaluation of the scenario forecast is hindered by the same factors as was the evaluation of backcasts in previous subsection: the majority of variation in forecasts originates in variation of single regressor, namely the risk-free interest rate (here the Eonia rate, before the policy rate(s)). Specifically, it is hard to judge whether the scenario forecasts are good, given that they mostly follow the profile of forecast for Eonia rate; analyzing the level of scenario forecasts for Euribor rate is in effect almost equivalent to analyzing the level of scenario forecasts for Eonia rate.

Figure 28: Euribor conditional scenario forecasts



To address this, right panel shows the transformation graph, which in present situation is the

graph of spread from Eonia rate.[30] The key takeaway from the figure is that even though the forecasts for Eonia rate look good in terms of level, looking at the spread reveals a crucial problem: the spread in the downside scenario, which includes stress in the financial markets, drops below the spread in baseline scenario. At first this might seem plausible if it reflects some of the drivers of the forecast, but further analysis can reveal that it actually reflects the (faulty) structure of the model. To see this, consider adding individual scenario graphs to the execution list and specifying libor spread together with baseline conditional scenario forecast as additional series to include in scenario graphs as follows[31]

```
eq_spread_level.speceval(exec_list="medium scenarios_individual",
  trans="spread", graph_bench="eonia", scenarios="bl su sd",
  graph_add_scenarios="libor_spread", baseline_forecast="t")
```

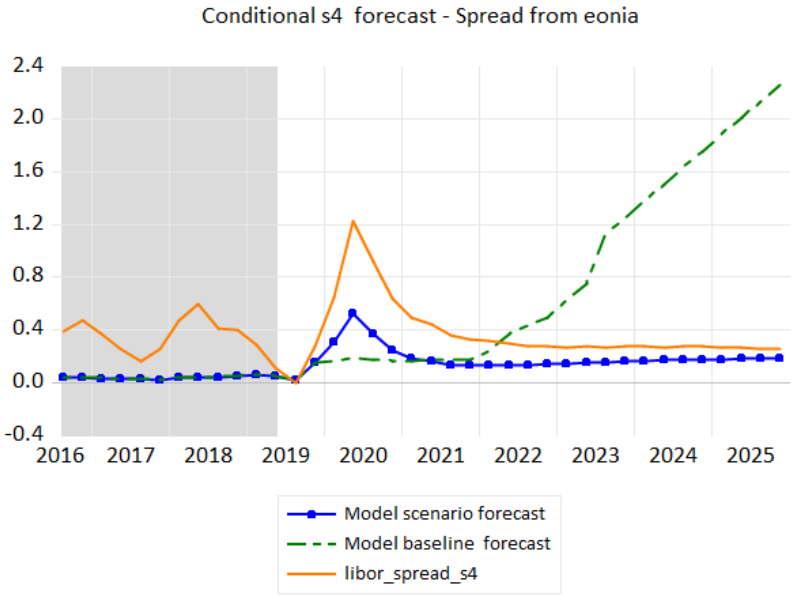
Figure 29 displays the resulting individual scenario graph, which includes the forecast for downside and baseline scenarios, as well as the Libor spread series. One can see that while the profile for Euribor spread is similar to the Libor spread, it is different in the key aspect I have raised in previous paragraph: there is no reversal between downside scenario and the baseline scenario. This demonstrates that the reversal originates in the structure of the model for Euribor. Specifically, the error-correction structure means that the the series has tendency to decrease to its normal values whenever it is above them, as it is during the period of stress. However, since the stress is mean reverting - Libor spread decreases to starting value after the initial increase - there is eventually an overshooting, as the decrease due to error-correction structure comes on top of decrease due to reversal in movement of the stress driver. Therefore, this application highlights how the transformation graphing can be important tool in constructing models that need to satisfy particular requirements in scenario forecasts *in terms of the given transformation.*

Concluding remarks. As a final note, let me highlight that this application demonstrates one important aspect of graphing transformations. In previous application, the transformation we were using corresponded more or less to the dependent variable. This raises question why does one need functionality to separately control the transformation used in graphs: after all, one could just change the way the dependent variable is specified, e.g. define new series that would be equal to growth rate

[30] Two comments are in order. If no transformation would be specified then the transformation graph would be in terms of first differences, given that the base variable is not trending (for trending base variable it would be percentage annualized growth rate). Second, the spread is calculated from the scenario value of the benchmark variable, i.e. scenario values of eonia rate here.

[31] Currently, `SpecEval` does not allow transformations in additional series and therefore the additional series - here spread between libor and FFR - has to be created as standalone series.

Figure 29: Euribor downside conditional scenario forecast - spread from Eonia rate



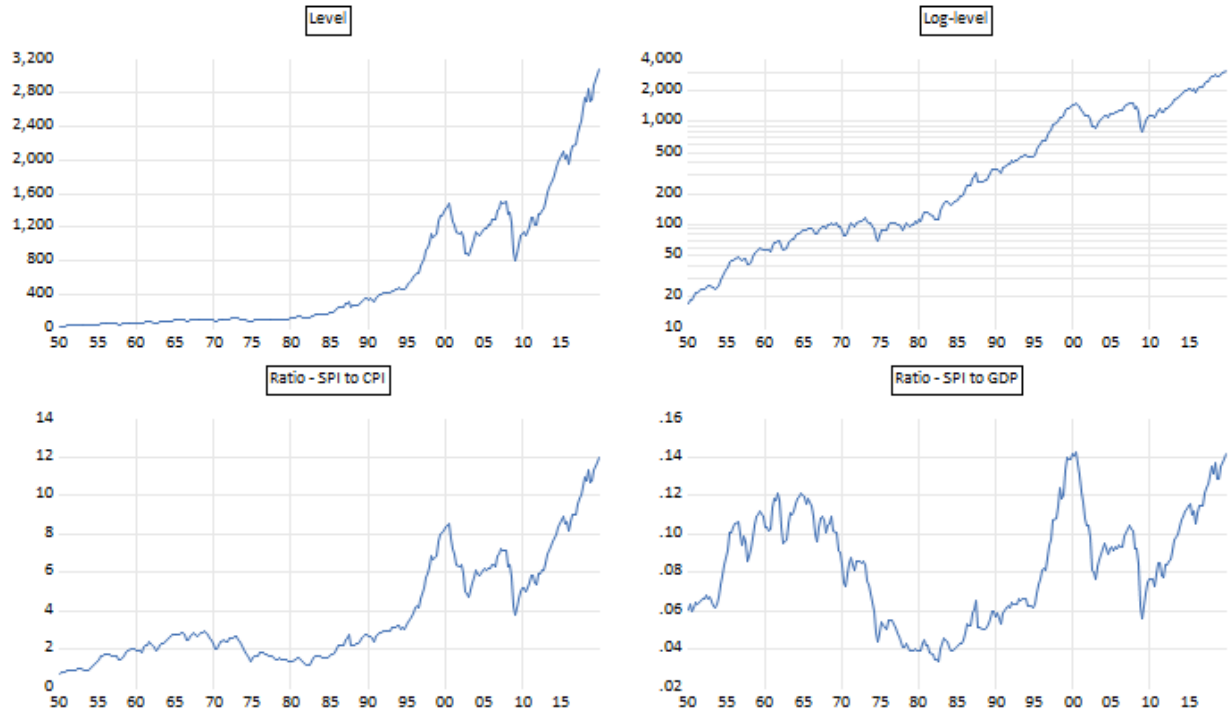
of CPI, in which case all the results including graphs would immediately be in terms of growth rate. However, in present case we saw that the transformation that was useful for us can be different from the actual transformation used as dependent variable, in which case redefining dependent variable would not be an option. Alternatively, some models combine different transformations on left-hand side and right-hand side of the equation. For example, one can sometmies encounter model that uses changes of spread of market interest rates on left hand side and level of market interest rates on the right hand side. Again, defining a new series and using it as dependent variable would not be an option here, since it would prevent us from using the (great) Eviews' functionality of handling forecasts from such models. Consequently, one needs functionality to control transformation used in graphs separately.

3.4 Application with log or ratio transformations

The application will use the US stock price index (SPI) as the variable of focus. Top lefty panel of Figure 30 displays the level of US SPI, this graphs makes it clear that the variable is exponentially growing variable, something that is confirmed by top-right panel, that uses logarithmic scaling for the vertical axis. The logarithmic scale helps a lot with visualizing the values of the series, since when pure level of the series is used then movements before 1990 are illegible. However, the logarithmic

scale does not change the fact that the series is trending, it only changes the nature of the trend from exponential to linear. This makes economical sense: stock prices reflect the profits of firms in the economy, and since the prices and output in economy are growing over time then stock prices as growing over time.

Figure 30: US SPI



If one would want to model the series in stationary form then one needs to eliminate the effect of growth in prices and output of the economy. Bottom-left panel of Figure 30 shows that taking ration of stock prices to overall price level does eliminate part of the trend, but not all. In contrast, taking ratio of stock prices to nominal GDP does change the series from trending to stationary.[32] Correspondingly, I will focus on model that uses ratio of SPI to nominal GDP as dependent variable:

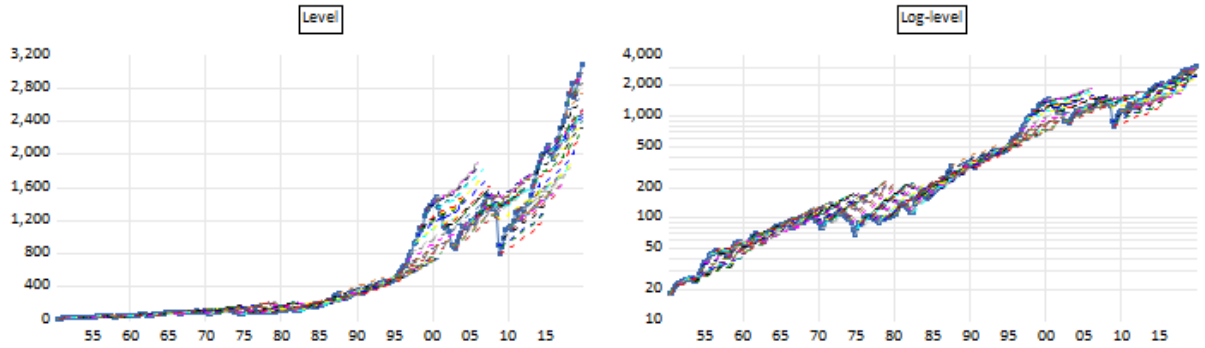
$$spi/gdp = \beta_0 + \beta_1 spi(-1)/gdp(-1) + \epsilon_t$$

Left panel of Figure 31 display the forecast summary graph.[33] The graph suffers from the same

[32] Here I ignore the considerations of alternative sources nonstationarity, e.g. breaks or changes in volatility.
[33] I focus on in-sample forecasting performance here for exposition purposes.

drawbacks as the graph in top-left panel of Figure 30: due to exponential growth of SPI it is impossible to evaluate the quality of forecasts in first half of the covered sample. `SpecEval` allows to address this problem in the same way as in Figure 30, i.e. by using logarithmic scale for the vertical axis. One just needs to specify the log as transformation argument. The resulting graph is in right panel of Figure 31. Once the logarithmic scale is used the forecast summary graph is much more valuable since it allows one to properly evaluate the forecast quality for the whole sample.[34]

Figure 31: Forecast summary graph - SPI




```
eq_spi_ar1.speceval(oos="f", trans="log")
```

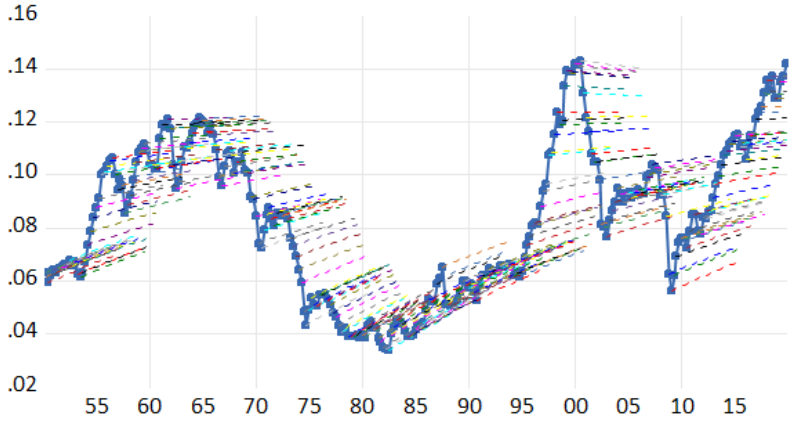
Nevertheless, the logarithmic scale provides interpretation solely in terms of forecast errors for SPI. Forecasting SPI is in some sense relatively easy: the model really just predicts that the SPI will grow at rate equal to growth of nominal GDP, with some allowance for auto correlation in the growth rate. Since the growth in nominal GDP is the main source of movements in SPI, as argued above, such prediction is relatively easy to make. Therefore, to properly evaluate the quality of forecasts it might be more useful to look at the prediction for SPI after accounting for the effect of development in nominal GDP. In other words, it might be more useful to look at prediction of the ratio of SPI to nominal GDP. `SpecEval` allows to do this if one specifies the ratio transformation and uses nominal GDP as benchmark variable:

[34] One could object that using logarithmic scale means that the same forecast error look different in beginning and in the end of the sample, so that it prevents proper evaluation of forecast quality. However, note that the economic interpretation of forecast errors in exponentially trending series should differ based on the value of the series since the variance of trending series increases. Making forecast error of 10 when the series has mean value of 100 and variance of 10 is qualitatively different from making the same forecast error when the mean value of 10 000 and variance is 1000. In other words, the proper evaluation of forecast errors for exponentially trending time series is in terms of percentage errors. Logarithmic scale is aligned with this view since the errors in terms of logarithmic scale are approximate to percentage errors.

```
eq_spi_ar1.speceval(oos="f", trans="ratio", graph_bench="gdp")
```

The resulting forecast summary graph is in Figure 32. The figure shows that the forecast in terms of ratio to nominal GDP are not bad, but that they are far from very good. This raises the issue whether including additional or alternative ARMA terms could help. I consider 4 alternative specifications: regression with second lagged dependent variable, and regressions with one moving average, one auto regressive, or both, errors in addition to single lagged dependent variable.[35] Table 11 displays the RMSE for all 5 specifications. It makes it clear that addion additional ARMA term helps, but that there are basically no differences in forecasting performance metrics between different types of additional ARM term.

Figure 32: Forecast summary graph - SPI (ratio)




```
eq_spi_ar1.speceval(spec_list="eq_spi*", use_names="t", oos="f")
```

The logarithmic and ratio transformations are also useful for evaluating scenario forecast, especially if those are made for long horizon over which the exponential trend starts assert itself. Consider creating scenario forecasts from the first model discussed in this section:

```
eq_spi_ar1.speceval(scenarios="bl su sd", tfirst_sgraph="1970q1")
```

[35] This entails including two types of ARMA terms. For example, the model with single lagged dependent variable and autoregressive error has following form:

$$spi/gdp = \beta_0 + \beta_1 spi(-1)/gdp(-1) + a_t$$
$$a_t = a_{t-1} + \epsilon_t$$

Table 11: RMSE - ARMA models for SPI

Specification	Forecast horizons (# of steps ahead) 1	4	8	12	24	Avg.
EQ_SPI_AR1	45.0	111	172	213	292	167
EQ_SPI_AR2	43.7	110	170	210	276	162
EQ_SPI_REGARMA01	43.9	111	170	210	280	163
EQ_SPI_REGARMA10	43.6	110	170	209	274	161
EQ_SPI_REGARMA11	43.7	110	170	210	275	162

The resulting conditional scenario forecast graph is in top-left panel of Figure 33. Since the graph contains long sample spanning several decades it is hard to evaluate the magnitudes of movements properly and impossible to determine if the growth over such long period is consistent with the historical observations. Of course, one could look directly at the growth rate transformation, which is included in the spool by default, but given the volatility of the series this helps only partly, see top-right panel. Using the logarithmic transformation will lead to conditional scenario forecast level graph being replaced by log-level graph; while using the ratio transformation will mean that the conditional scenario forecast transformation graph will no longer be in terms of growth rate but in terms of ratio from the specified variable, see bottom panels of Figure 33.

```
eq_spi_ar1.speceval(scenarios="bl su sd", tfirst_sgraph="1970q1",
  trans="log")
```

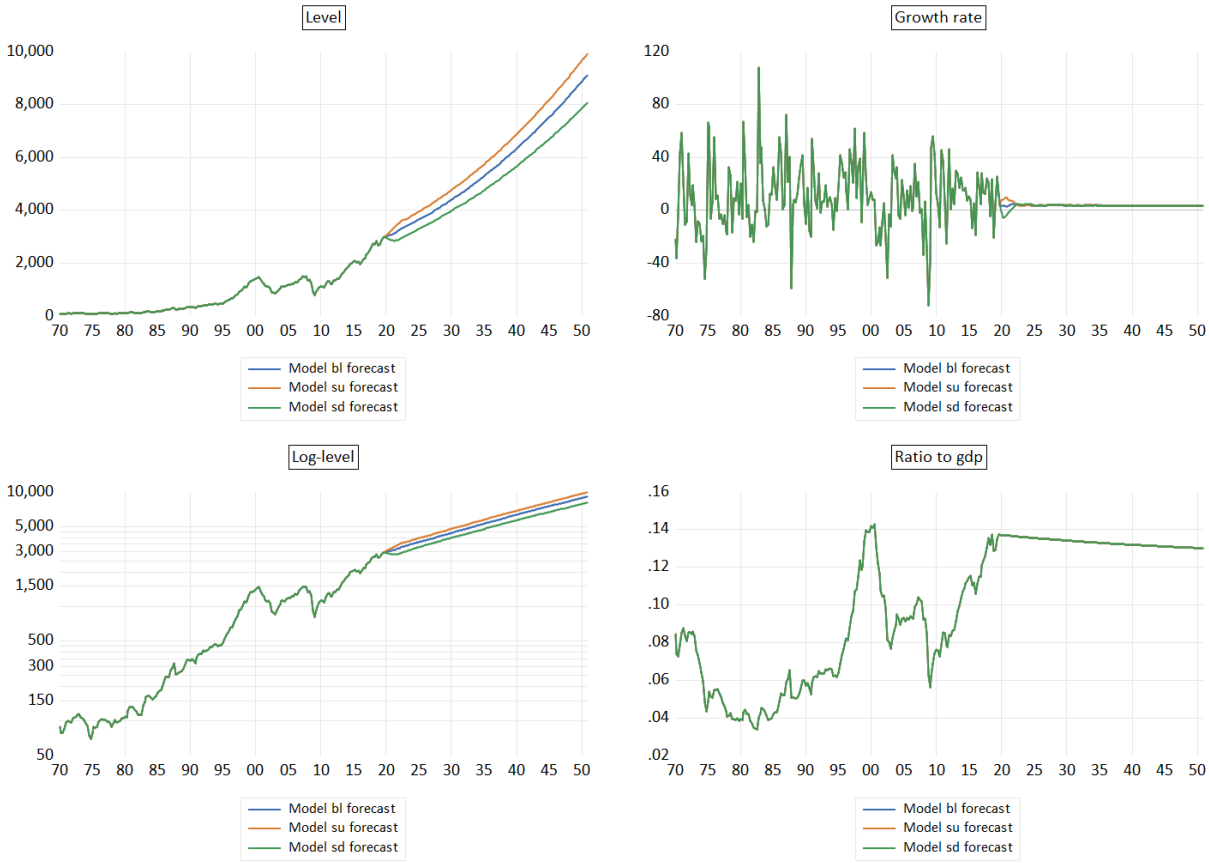
```
eq_spi_ar1.speceval(scenarios="bl su sd", tfirst_sgraph="1970q1",
  "trans="ratio", graph_bench="gdp")
```

3.5 Application with unconditional forecasts

In all previous applications we have evaluated the alternative specifications in terms of *conditional* backtest exercise: the historical forecasts were created based on the actual historical values of all independent variables. While this forecasting exercise is best suited for evaluating the alternative specifications since the only source of errors is the model error, often one is also interested in *unconditional* backtest exercise in which the forecasts are not based on any future information and hence are true forecasts.[36] `SpecEval` offers two ways how to produce such unconditional forecasts. In this application I will focus on approach when existing forecasts for independent variables are prepared in

[36] See discussion in for example `?` for more elaboration on value of conditional and unconditional backtest exercises as well as more complex example what unconditional backtest exercise entails.

Figure 33: Conditional scenario forecast graph - SPI



the workfile and then supplied as input to the forecasting process run by the add-in. Next application will then show how to make forecasts for

Consider again the Eonia rate, as in application 3.3. In that application I have studied the conditional forecasts, i.e. forecasts which were based on actual observed historical values for policy rates were used. Of course, these are not true forecasts, because at the point of forecast the forecaster would not know what will be the future values of policy rates. Therefore, it is interesting to look at how would the forecasts look if one would treat the future values of policy rates as unknown, i.e. if we replace the actual values of policy rates with the forecasts for policy rates. Of course, this requires us first to have such forecasts, what raises the questions of what model should be used to create those. A natural candidate a random walk model, which posits that the future values of policy rates will be equal to current values. While one could create such forecasts manually through simple loop, here I will use the `SpecEval` add-in functionality to create those, since it shows how the add-in can also be used for identities.[37] In addition, using the add-in will also mean that we can easily evaluate the resulting forecasts.

To create forecasts from random walk model for policy rates using `SpecEval` , we first need to create a string object that contains the identity:

```
string id_mrr_rw = "mrr = mrr(-1)"
```

The identity simply says that the current value of main refinancing rate should be equal to previous value, a random walk.[38] Once this string witht he identity is created, one can use `SpecEval` in exactly the same way as for equations.[39] This means that we can get the usual spool with results be issuing following command:

```
id_mrr_rw.speceval(nodialog)
```

Figure 34 shows the resulting forecast summary graph produced by the add-in. It is clear that the

[37] Note that random walk *without drift* is an identity in that it does not contain any estimated coefficient. This also means that forecasts from random walk cannot be created using an equation object in Eviews, since one cannot create such equation.

[38] Note that the name of the identity object is irrelevant the same way the equation name is irrelevant. Similarly, the exact form of the identity does not matter either, and same identity can be written in different way. For example, an alternative way to write identity corresponding to random walk is using first difference transformation as follows:

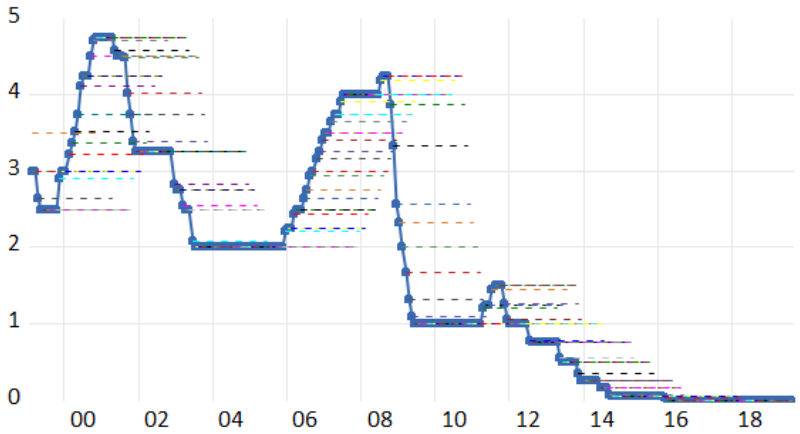
```
string id_mrr_rw = "d(mrr) = 0"
```

`SpecEval` will work in exactly the same way whether one uses the first or the second specification of the same identity.

[39] Of course, some of the functionality of the add-in is not applicable for identities.

forecasts are not great in terms of their precision, but it is hard to know if this could be improved upon. For example, random walk model was much better at forecasting MRR in the second half of the sample than financial markets, which were consistently predicting increase in MRR which never occurred.

Figure 34: Forecast summary graph - MRR



To use the forecast in further work, we first need to store them, something that is easily done by specifying the store option:

```
id_mrr_rw.speceval(nodialog,keep_forecasts="t")
```

After execution the workfile will include individual series for every individual forecast, using the naming convetion base_variable_fforecast_start, so that series 'mrr_f2008M01' contains forecast that start in 2008M01 (i.e. 2008M01 is the first month in which forecast is different from actuals). These forecasts can then be used in follow-up unconditional forecasting exercise for eonia rate.[40] Consider applying it to specification with ARMA errors we have used for Eonia rate in application 3.3. To specify that the add-in should use the forecasts for MRR rather than the actual historical values one needs to include it among the forecasted independent variables:

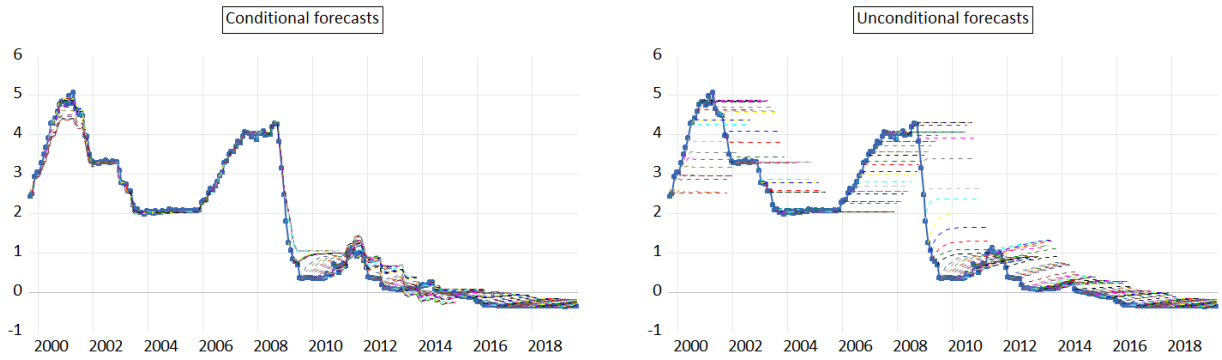
```
eq_eonia_arma.speceval(forecasted_ivariables="mrr")
```

Figure 35 includes the foreast summary graph from the conditional and unconditional backtest exercise, in left and right panel, respectively. Unsurprisingly, the figure shows that the unconditional

[40] Note that if the forecasts are created manually or just obtained elsewhere they need to conform to this naming convention.

forecasts are much worse than the conditional forecasts. This is especially true in periods when forecasts for MRR are far from the actuals.[41]

Figure 35: Forecast summary graphs - Eonia

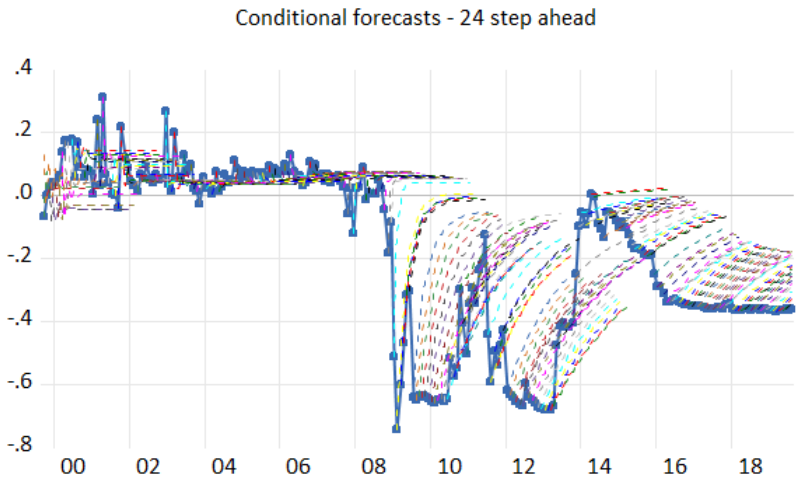


One could of course make this application more complicated. For example, **?** creates unconditional forecasts for the structural model in equation 5 in application 3.3. This requires forecasts for three independent variables, MRR as well as deposit rate and excess reserves. Moreover, it also considers two alternative models for excess reserves, a random walk model specified by identity and ARMA model estimated by equation. All of this can be easily achieved using the `SpecEval` add-in.

Rather than providing additional illustration of this functionality, I will illustrate one important feature of transformation graphing that was the topic of previous two applications. Specifically, the transformation graphs in terms of spreads or ratios are created with respect to the actual series used in forecasts. This means that if forecasted values were used for independent variable which is also the spread/ratio benchmark variable, then the forecast summary graph will display spread/ratio from the forecasted values, not the actual historical values. For illustration see Figure 36 which shows the forecasts from right panel of Figure 35, but this time displaying the spread from MRR rather than the level of Eonia rate. The figure demonstrates that the forecast summary graph in terms of spread provides a good instrument to evaluate the model for Eonia rate, in contrast to the forecast summary graph in levels. Of course, Figure 36 is the same as Figure 20.

[41] This of course illustrates the drawback of unconditional forecasts: vast majority of variation in Eonia rate comes from variation in policy rates, and hence vast majority of forecast errors in unconditional forecasting exercise comes from model for MRR, not from model for Eonia rate. This hampers the evaluation of the model for Eonia rate.

Figure 36: Unconditional forecast summary graph - Eonia rate spread



3.6 Application with multiple variables/equations

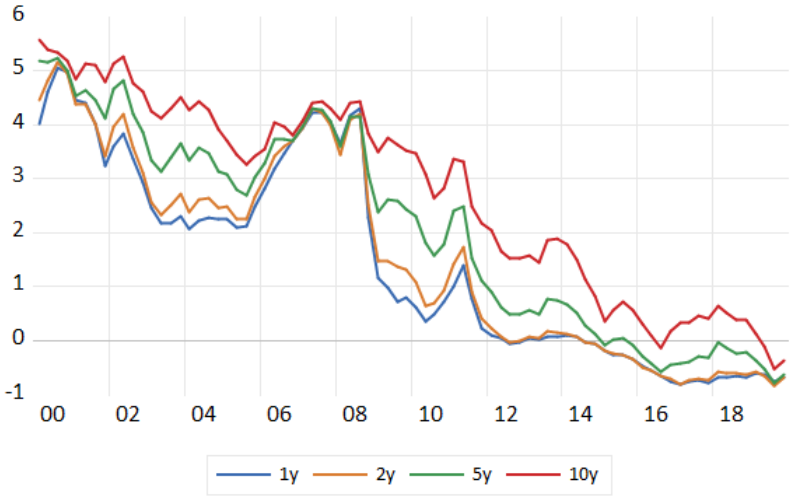
Previous application showed that the add-in can be used to perform unconditional forecasting for multivariate single-equation models by supplying the forecasts for the independent variables. However, this is only one approach to create unconditional forecasts, with another approach possible. Most importantly, the previous approach really makes sense only if the forecasts for independent variables are really independent of the forecasts for the main variable of interest. This was plausibly the case for Eonia rate and policy rates, since random shocks to Eonia rate do not influence the policy rate. However, this approach is no longer suitable if independent variables respond to variation in the variable of interest, what in previous application would require policy rates themselves to respond to variation in Eonia rate. In such situation the forecasting cannot be separated into two independent steps, one for independent variables and one for dependent variable, but rather the forecasts have to be created together.[42] `SpecEval` allows user to also perform such joint forecasting exercise, and this subsection shows how to do that.[43] In addition to showing how to perform such unconditional forecasting exercise, the section also highlights how the add-in allows user to include multiple *forecasts* in forecast graphs.

[42] Of course, for similar reasons it might be more proper to also estimate the models together, rather than separately. I ignore this consideration here, partly because at this point `SpecEval` is able to use only the most basic multiequation models, VARs.

[43] Note that the current approach is also applicable in situation when the forecasts are independent, so in some sense encompasses the first approach. That said the former approach is computationally more efficient when one is evaluating multiple specifications while using the same model for independent variables.

The functionality will be demonstrated on example of model for euro zone yield curve, i.e. model for multiple German yields. First, consider Figure 37 which shows the 1-year, 2-year, 5-year, and 10-year yields. The figure makes it clear that yields of different maturity move closely together, a economic theory suggests they should. This suggests that the different yields should be either modeled as function of each other, or as function of common driver. Here, I will explore the first approach which allows me to demonstrate the functionality of `SpecEval` add-in for forecasting based on models that include multiple equations. Specifically, I will treat 1-year and 10-year yields as truly exogenous to the remaining three yields, but the remaining two yields will be function of each other. The system of equation is therefore following:

Figure 37: German yields



$$2y_t - 1y_t = \beta_0 + \beta_1(5y_t - 1y_t)$$

$$5y_t - 2y_t = \beta_0 + \beta_1(10y_t - 2y_t)$$

This means that the spread between 2-year yield and 1-year yield is function of spread between 5-year yield and 1-year yield, while the spread between 5-year yield and 2-year yield is function of spread between 10-year yield and 2-year yield.[44] This makes the system of equations truly simultaneous: to

[44] The logic behind the model is simple: typically, the 2-year yield will be somewhere between 1-year yield and 5-year

know the 2-year yield one needs to know the 5-year yield, but to know the 5-year yield you need to know the 2-year yield. From perspective of evaluating forecasting performance this means two things. First, evaluating 2-year yield equation based based on *conditional* forecasting performance is unlikely to be very valuable: if one knows the 1-year and 5-year yields then there is likely little scope for forecast errors. Second, to construct unconditional forecasts one cannot perform the same exercise as in previous application, since 5-year yield cannot be foretasted independently of forecast for 2-year yield. The only solution is for the two yields to be forecasted together. `SpecEval` add-in provides such possibility.

To forecast multiple variables together one needs to specify that the model used for creating forecast should contain multiple equations using the settings options 'eq_list_add'. In present case the add-in would be executed as follows:

```
eq_2y.speceval(eq_list_add="eq_5y", horizons_graph="4",
   trans=spread,graph_bench="yield1y")
```

The resulting forecasts (in terms of spread from 1-year yield) are displayed in left panel of Figure 38. Meanwhile, right panel shows the analogical forecasts when the 5-year yield is treated as exogenous, i.e. when forecasting model does not include equation for 5-year yield. The key point of the figure is that if one would treat the 5-year as exogenous variable then the forecast performance would seem much better than it would in reality be.[45]

Of course, the functionality can be used in context of multiple alternative specifications. For example, consider situation when one wants to compare the performance of two alternative specifications for 2-year yield, one proposed above, and one which in addition includes AR(1) error term. The two specifications can be evaluated in the semi-unconditional setting as follows:

```
eq_2y.speceval(spec_list="eq_2y_ar1", alias_list="simple ar1",
   eq_list_add="eq_5y", horizons_graph="4", trans=spread,graph_bench="yield1y",
   oos="t")
```

yield. This model structure aims to approximate that, while ensuring that if forecaster adjust either of the 2 endogenous yields then the other yields respond.

[45] Of course, this is true for all comparisons of conditional and unconditional forecasts. However, as argued above, in many situations this does not mean that one should evaluate models in terms of unconditional forecasts, since the errors for the forecast errors for independent variables are truly independent of the forecast errors for the dependent variable, so they only serve as noise making the forecast performance evaluation harder. This is different from the situation here where the forecast errors for 5-year yield are function of forecast errors for 2-year yield. Note that if one would use the approach outline in previous application then the same problem would still apply, but to a lesser degree: while one would still use forecasts for 5-year yield, these forecast would be conditional on perfect knowledge of 2-year yield, and hence one would be including only the forecast error that is independent of forecast error for 2-year yield. In some applications this might be of interest.

Figure 38: Forecasts for 2-year yield

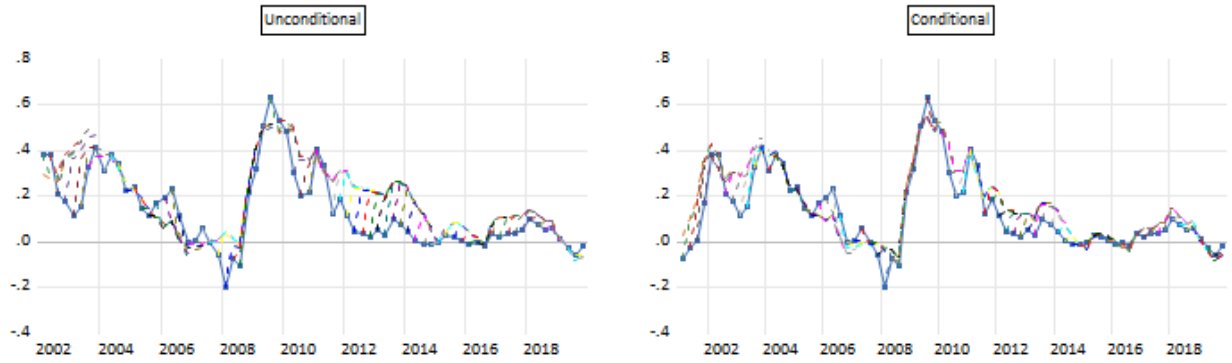


Table 12 shows the resulting RMSE for both specifications, highlighting the fact that including the AR(1) error term helps with the short horizon, but makes forecasts less precise in medium and long horizons.

Table 12: RMSE - models for 2-year yield

	Forecast horizons (# of steps ahead)					
Specification	1	4	8	12	24	Avg.
simple	0.11	0.12	0.11	0.11	0.13	0.12
ar1	0.081	0.11	0.14	0.14	0.16	0.13

The above example showed that one can evaluate multiple alternatives for single equation while including additional equations in the forecasting model. However, in many situations user might want to evaluate alternative specifications for system of equations. For example, one might want to evaluate the forecasting performance for 2-year yield corresponding to two different set of equation: one outlined above, and one in which both equations include AR(1) error term. What is effectively needed is ability to change specifications for additional equations as well as the main equations. `SpecEval` add-in allows user to do that. One only needs to add "[ALIAS]" at the end of the equation name and ensure that the original equations and the additional equation(s) have the same alias. For example, consider estimating equation for 5-year yield with AR(1) error and naming it 'eq_5y_ar1', so that equations for both yields use alias '_ar1'. Then one can evaluate the forecasting performance of system of equations without and with AR(1) error as follows:[46]

[46] The above execution also illustrates two specific aspects of behavior of the add-in with respect to aliasing of additional equations:

- Since the full alias for equation for 2-year yield '_ar1', rather than 'ar1', the add-in will not be able to locate the additional equation. However, in such situation the add-in tries to locate the equation with alias that includes

```
eq_2y.speceval(spec_list="eq_2y_ar1", alias_list="simple
  ar1", eq_list_add="eq_5y[ALIAS]", horizons_graph="4",
  trans=spread,graph_bench="yield1y")
```

Table 13 shows the resulting RMSEs. As can be seen including AR(1) error also in the equation for 5-year yield lowers the 1-period ahead RMSE.

Table 13: RMSE - models for 2-year and 5-year yield

	Forecast horizons (# of steps ahead)					
Specification	1	4	8	12	24	Avg.
simple	0.11	0.12	0.11	0.11	0.13	0.12
ar1	0.065	0.11	0.14	0.14	0.17	0.12

The multiple forecasted variable functionality is not limited to equations. User can also include identities. Consider an example in which one wants to also include 1-year yield, but which will be separated into equation for spread between 1-year yield and eonia rate, and identity for 1-year yield as follows:

$$1y_t^s = \beta_0 + \beta_1 1y_{t-1}^s + \epsilon_t 1y_t = 1y_t^s + eonia_t$$

where $1y_t^s$ is the spread between 1-year yield and eonia rate.[47] To include both the equation and the identity in the forecasting model one simply needs to create the equation, a string with the identity and then specify them as additional equations as follows:

```
eq_2y.speceval(eq_list_add="eq_spread1y id_yield1y eq_5y",
  horizons_graph="4", trans=spread,graph_bench="yield1y")
```

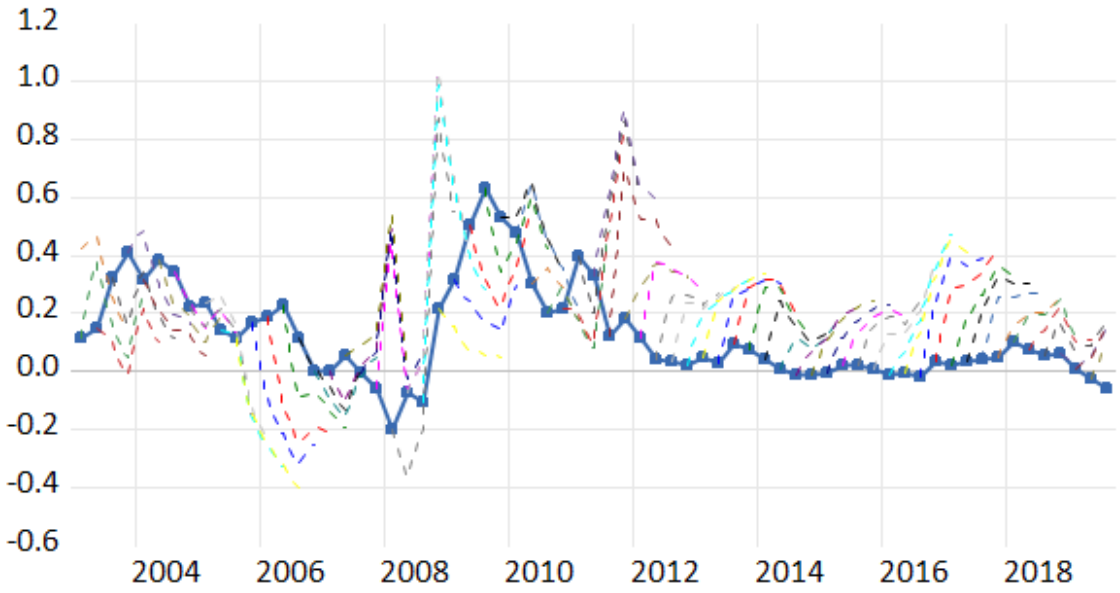
where 'eq_spread1y' is the equation for 1-year spread and 'id_yield1y' contains the identity for 1-year yield. Figure 39 shows the resulting forecasts, which of course show much lower precision given that 1-year yield is now treated as forecasted variable.

"_" in addition to specified alias.

- Since the original equation for 5-year yield does not include alias '_simple', then the add-in will fail to locate such equation. In such situation it will attempt to locate equation without any alias, in current case 'eq_5y', and use that one. This means that user can analyze mix of specifications, some that have alternative specification for the additional equations and some that don't.

[47] Of course this separation is economically irrelevant: one could simply replace $1y_t^s$ in the first equation with $1y_t - eonia_t$ and get exactly the same equation. however, sometimes it is more convenient from forecasting perspective to have the 1-year spread as explicitly defined variable.

Figure 39: Forecasts for 2-year yield



Finally, there is also alternative, possibly more straightforward way of including additional equations. Rather than specifying list of equations to be added, user can specify list of variables for which equations/identities should be included together with specifying name of model object that includes equations/identities for given variables. This is especially useful in context of using possibly complex identities. Consider re-doing previous analysis, but this time in situation when the workfile contains model object 'm_yields' that contains the equations for 1-year spread and 5-year yield and identity for 1-year yield. In such situation one could perform the same analysis by using following execution command:

```
eq_2y.speceval(eq_list_add="spread1y yield1y yield5y",
  model_name_add="m_yields", horizons_graph="4", trans=spread,graph_bench="yield1y")
```

where crucially I have included the information about the model objects that contains the additional equations/identities. In such situation the add-in will first try to search for equation/string objects of given name, and it if fails to locate them, it will search for model objects for these variables specified.

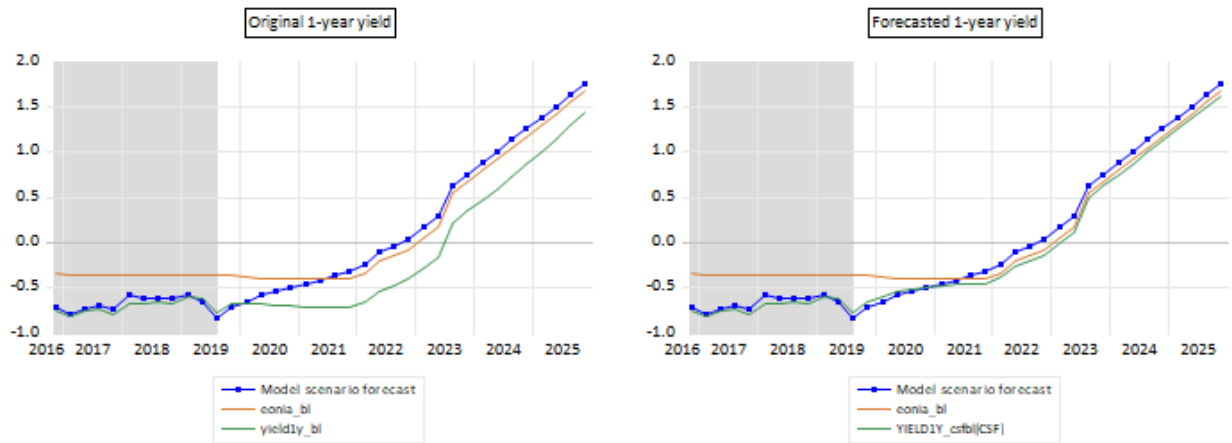
The inclusion of additional equations in the forecasting model is not limited to backtesting forecasting exercises. If additional equations are included in the model, then they will also be used in creating conditional scenario forecasts. Like this one can make the conditional scenario forecasts arbi-

trarily less conditional by including equations for additional independent variables, effectively making these variables endogenous in terms of the scenario forecasts. Continuing with the example above, but this time we want to analyze the conditional scenario forecasts while treating 1-year, 2-year and 5-year yields as endogenous variables. To do that one can just do the same things as always, i.e. specify list of scenarios. Imagine though that you want to also include the forecasts for eonia rate and 1-year yield in the conditional scenario graphs for 2-year yield, so that you include settings option 'graph_add_scenarios':

```
eq_2y.speceval(eq_list_add="spread1y yield1y yield5y",
  model_name_add="m_yields", scenarios="bl", graph_add_scenarios="eonia
  yield1y")
```

The resulting conditional scenario graph is in left panel of Figure 40. However, it should be clear from from the figure that the graph displays something potentially different from what the user had in mind. Specifically, it displays the forecasts for eonia rate and 1-year yield that were originally in the workfile. While for Eonia rate this is as intended, since there is no other forecast, for 1-year yield it is likely that the user would want to see the forecast actually created in the forecasting process, given that is the forecast used for forecasting 2-year yield. One can easily rectify this by indicating that it should be the conditional scenario forecast, not original forecast, that should be used in the graph. This is done by appending "[CSF]" at the end of the 1-year yield variable name in the 'graph_add_scenarios' option:

Figure 40: Forecasts for 2-year yield



```
eq_2y.speceval(eq_list_add="spread1y yield1y yield5y",
  model_name_add="m_yields", scenarios="bl", graph_add_scenarios="eonia
  yield1y[CSF]")
```

The result is in right panel of Figure 40: the 1-year yield forecast displayed in the graph now clearly corresponds to the 1-year yield forecast used in forecasting 2-year yield.

3.7 Application with custom re-estimation

Previous applications have focused on the functionality inbuilt in the add-in. While this functionality includes some customization of the equation re-estimation process, there are many applications which require more complex re-estimation that cannot be performed using the in-build process. For these reasons the program allows user to specify that custom re-estimation should be performed by setting add-in argument option 'custom_reest="t"'. If this argument is supplied then the program will perform commands contained in separate program named 'custom_reestimation.prg'. It will first search in current Eviews run path (i.e. if speceval is executed from program then the location of that program). If such program is not located then it will use default location in the `SpecEval` add-in folder, subfolder 'subroutines'.[48] The exact contents of the custom re-estimation program are fully up to the user. The only thing that is required is that at the end of the re-estimation procedure there is model object named 'm_speceval' which will be able to produce forecasts for given base variable. The rest of application provides illustration of such custom re-estimation program.

The application will again focus on Eonia rate, as in applications 3.3 and 3.5. This time around I will focus on the situation when one wants to produce out-of-sample forecasts for model with break.[49] Specifically this application will consider the situation when one wants to estimate following equation:

$$Eonia_t = \begin{cases} \beta_{10} + \beta_{11} MRR_t + \beta_{12} DR_t + a_{1t} & \text{if } D_t^{Regime} = 0 \\ \beta_{20} + \beta_{21} MRR_t + \beta_{22} DR_t + a_{2t} & \text{if } D_t^{Regime} = 1 \end{cases}$$

where D_t^{Regime} is regime dummy indicator and a_1, a_2 follow ARMA(1,1) process. The idea to allow relationship between Eonia rate and the two policy rates, as well as the dependency on past shocks, to

[48] This default program includes a simple example of such custom re-estimation which will lead to random-walk model forecasts.

[49] While Eviews provides infrastructure for models with break under its 'breakls' estimation command - and hence basic out-of-sample forecasting for break models can be handled by the existing `SpecEval` infrastructure - the current application will consider case in which the existing Eviews infrastructure cannot be used, as explained below.

be different before and after 2008M10, when Eonia rate has undergone change in regime, see discussion in **??**.

If the equation could be recursively estimated via the Eviews `breakls` estimation method then one could simply rely on the `SpecEval` in-built functionality to produce the out-of-sample forecasts. However, there are three reasons why one cannot use the existing Eviews estimation method for models with breaks, *recursively* or otherwise. First, the break should be included only if the date is above certain threshold. Second, there is potential perfect multicolinearity between MRR and DR, which before 2009 moved in perfect lockstep, and what prevents use of in-built Eviews break tests. Third, Eviews breakls estimation method does not allow for ARMA terms. Therefore, rather than estimating the equation in Eviews and the executing the `SpecEval` add-in as is, one needs to estimate single regime equation and write custom re-estimation code that will produce recursive estimates of the two regime equation.

First, consider estimating the single equation model as follows:

```
equation eq_eonia.ls eonia c mrr dr ar(1) ma(1)
```

Next, we need to create the custom re-estimation program that will be used by the add-in later on. The first step of the custom re-estimation is re-estimating the single-regime equation. To do that, one needs to first check whether the two policy rates are perfectly correlated in the given recursive estimation sample, or not. Since `SpecEval` performs this automatically it does include functionality for this that can be leveraged here. Specifically, there is a subroutine 'perfect_corr_identification' that check whether perfect correlation is present. The full code that results in recursive estimation of single regime equation is below:

```
' 1. Estimating original equation

%est_command_reest = {st_spec_name}.@command

if @isobject("gr_"+ st_spec_name + "_regs")=0 then
  {st_spec_name}.makeregs gr_{st_spec_name}_regs
endif

!reg_n = gr_{st_spec_name}_regs.@count
call perfect_corr_identification(gr_{st_spec_name}_regs.
    @members, %tfirst_reestimation,  %tlast_reestimation, "
    perfectcor")

if @instr("  "  + @upper(%est_command_reest) + " ", " C ")=0
     and !reg_n=2  then
  %perfectcor = "f"
endif

if %perfectcor = "t" then
  for !reg = 2 to  !reg_n
    if !perfectcor{!reg} = 1 then
      %reg_string = gr_{st_spec_name}_regs.@seriesname(!reg)
      %est_command_reest = @trim(@replace(" " + @upper(%
          est_command_reest) + " ", " "+ @upper(%reg_string)
          + " ", " "))
    endif
  next
endif

smpl {%sub_tfirst_reestimation} {%sub_tlast_reestimation}
equation eq_nobreak.{%est_command_reest}
```

Next, if the recursive estimation sample ends later than 2009M01, then we want to perform standard Chow test on the structural part of the equation to determine whether use of two regimes is supported by formal statistical analysis. This is performed by following section of the code:

```
' 2. Testing

if @dtoo(%sub_tlast_reestimation)>@dtoo("2009M01") and @dtoo
   (%sub_tlast_reestimation)>(@dtoo("2008M10")+{st_spec_name
   }.@ncoef*2) then

  %eq_command_structural = {st_spec_name}.@command

  for %arma_term ar(1) ma(1)
     %eq_command_structural = @replace(@upper(%
        eq_command_structural),@upper(%arma_term),"")
  next

  equation eq_structural.{%eq_command_structural}

  freeze(tb_test) eq_structural.chow 2008m10 @ c

  scalar sc_pval = @val(tb_test(6,5))

  if sc_pval<0.05 then
    !two_regimes = 1
  endif

' rename tb_test  tb_test_{!fp}
' rename sc_pval sc_pval_{!fp}

  delete(noerr) tb_test sc_pval
endif
```

Finally, we need to estimate the final equation that will be used for forecasting. If presence of two regimes has been supported by the statistical analysis then estimation sample will start in 2008M10l; if not then the estimation sample will cover full available sample.[50] This is done in next section of the code:

[50] Note that the errors are assumed to be different between the two regimes so that peice-wise estimation is appropriate.

```
' 3. Estimating final equation

if !two_regimes = 0 then
  smpl {%sub_tfirst_reestimation} {%sub_tlast_reestimation}
  equation {st_spec_name}_reest.{%est_command_reest}
else
  smpl 2008M10 {%sub_tlast_reestimation}
  equation {st_spec_name}_reest.{%est_command_reest}
endif

Finally, we need to create the model object that will be
    used in further procedures by \speceval add-in:

if !fp=1 then
  model m_speceval
  m_speceval.merge {st_spec_name}_reest
else
  m_speceval.update {st_spec_name}_reest
endif

copy {st_spec_name}_reest {st_spec_name}_reest{!fp}
```

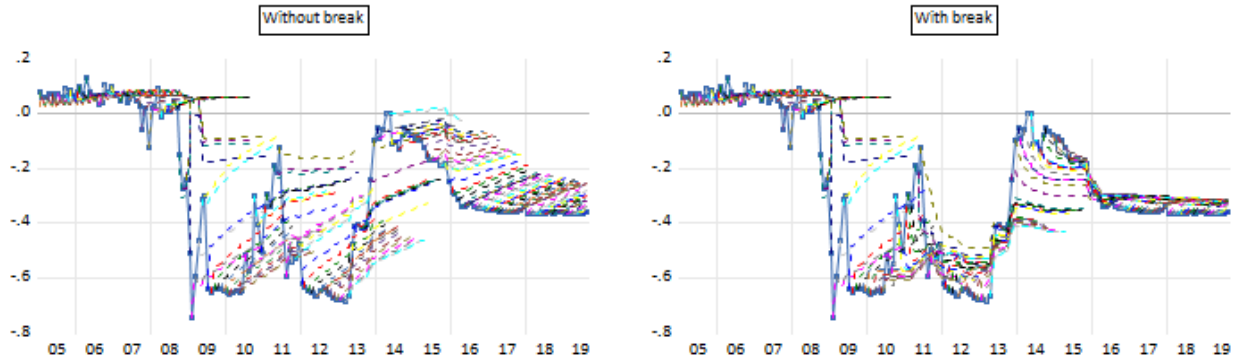
There are two things worth mentioning. First, in the present case one simply needs to put the resulting equation in the model object. However, in general the model object can be arbitrarily complicated; for it can include additional equations, identities or other estimation objects. Second, the model object is created only once and the equation is then being updated in the model object to preserve the continuity between different recursive forecasts. Here this is achieved by leveraging the program string '!fp' which indicated current recursive forecast period order. Of course, this is strictly speaking not necessary and one could create the model object from scratch in each recursion.

Once the program with custom re-estiamtion subroutine is finished one can execute the `SpecEval` from the original program as follows:

```
eq_eonia.speceval(custom_reest="t")
```

To complete this application Figure 41 illustrates the value allowing for break in the equation. Left panel shows the forecast summary graph for equation without break, while right-panel shows the same graph when one allows for break in the relationship. The figure makes it clear that allowing for break is important for the forecast quality.

Figure 41: Forecast summary graphs - Eonia rate with break



3.8 Application with using intermediate objects

Apart from using the existing functionality and outputs in-built in the add-in, `SpecEval` also allows user to easily store process objects which can then be used for analysis. The full list of process objects that can be stored is in `SpecEval` documentation in document '*Outputs for specification evaluation add-in.docx*'. Here I illustrate the storing and use of these process objects on simple application that considers creating graphs of recursive impulse responses.

The application will again focus on Eonia rate, as in applications 3.3, 3.5 and 3.7. Since there was a break in relationship between Eonia rate and the policy rates, as discussed in **??**, it is interesting to see how does this break influence the recursive ARMA model structure if it is not accounted for. Consider using the `SpecEval` add-in to create recursive equation estimates for following equation:

$$Eonia_t = \beta_0 + \beta_1 DR_t + a_t$$

where a_t follows ARMA(p,q) model, as determined according to SIC. To obtain and store the recursive equations one needs to execute `SpecEval` as follows[51]

```
eq_eonia.speceval(auto_select="t", exec_list="forecasts stability",
   keep_eqs="t")
```

The key settings argument is 'keep_eqs' that instructs the add-in to store the recursive equations.[52]

[51] Note that I included only 'forecasts' and 'stability' arguments in the execution list, since at this point I am interested only in the recursive equations that are produced in the forecasting process and in the model stability graphs, and not in any other other parts of the add-in.

[52] In addition I have also instructed the add-in to store the information objects, since I will rely on those in later analysis.

As a result after execution the workfile will include recursive equation objects with following naming convention: 'eq_eonia_reesttstart' for equation 'eq_eonia' estimated on sample from first estimation date until last date before 'tstart'; as such, equation object 'eq_eonia_reest2008m09' contains equation 'eq_eonia' estimated on sample from start of the estimation until 2008M08. Crucially, I have specified that automatic selection should be performed (and included the necessary information among the equation object attributes), so that recursive equations do not differ only in the estimated coefficients but also in AMRA terms that are included.

With these equations available one can proceed with analysis of recursive impulse responses. Specifically, consider the situation that we want to know how does the impulse response to unit shock after 6, 12 and 24 months evolve with the estimation sample. To answer this question one can use the in-built Eviews functionality to study the impulse responses. For single equation, the following command saves vector 'v_irf' with impulse response for up to 24 periods:

```
eq_eonia.arma(type=imp,imp=1,hrz=24,save=v_irf)
```

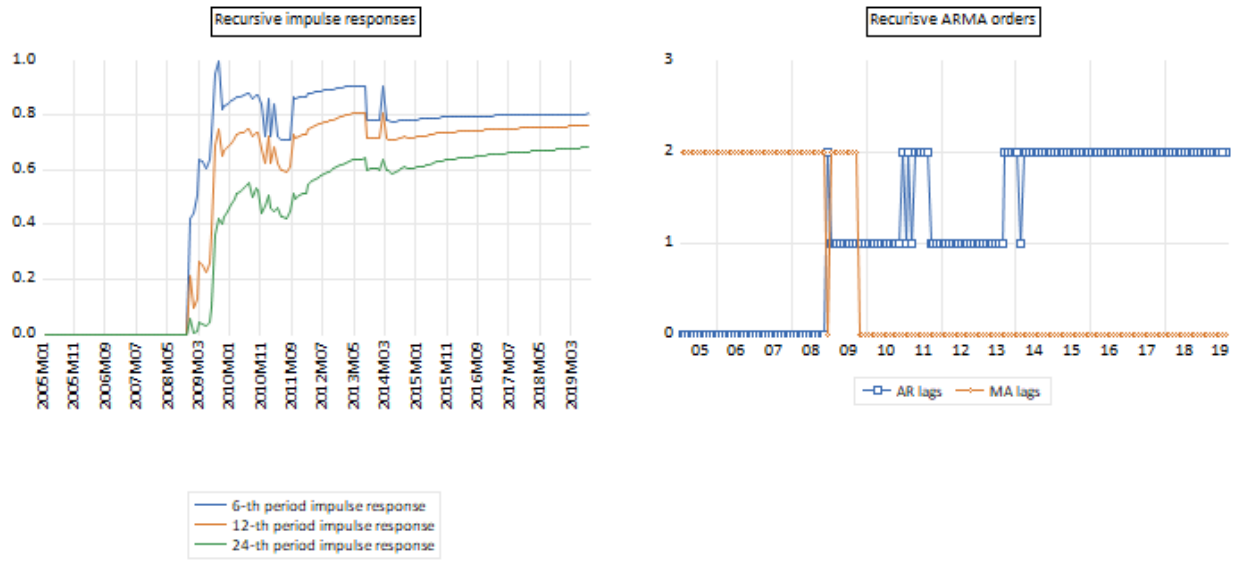
Leveraging this command, one can obtain the impulse responses based on the stored recursively estimated equations, store them in matrix object and create a graph displaying those impulse responses corresponding to the end of estimation sample. This is performed by following code:

```
%sub_horizons = "6 12 24"
!max_horizon = @val(@word(%sub_horizons,@wcount(%
   sub_horizons)))

matrix(sc_forecastp_n,@wcount(%sub_horizons)) m_irf

%fqs = ""

for !fp = 1  to sc_forecastp_n

  %fq = tb_forecast_numbers(1+!fp,2)

  eq_eonia_reest{%fq}.arma(type=imp,imp=1,hrz={!max_horizon
     },save=v_irf_{!fp})
  close eq_eonia_reest{%fq}

  for !h = 1 to @wcount(%sub_horizons)
    %h = @word(%sub_horizons,!h)
    m_irf(!fp,!h) = v_irf_{!fp}({%h})
  next

  %fqs  = %fqs +%fq+ " "
next


smpl {st_tfirst_backtest} {st_tlast_backtest}
freeze(gp_irf) m_irf.line
gp_irf.setobslabels {%fqs}

for !h = 1 to @wcount(%sub_horizons)
  %h = @word(%sub_horizons,!h)
  gp_irf.setelem(!h) legend({%h}-th period impulse response)
next
```

Figure 42 displays the graphs on which resulting analysis can be made. The left panel shows the recursive impulse responses at the three selected horizons. The graph makes it clear that the impulse responses increase over time: initially they are zero at the selected horizons, but as soon as the estimation sample includes observations after 2008M10 the impulse responses become positive. Eventually, even the response after 24 months becomes large. Meanwhile, right panel provides explanation for this: the selected model change from moving average model, which features only very short propagation of shocks, to model with one, and later two autoregressive orders, which can feature long propagation of shocks. Overall, the figure makes it clear that ignoring the break in the relationship between Eonia rate and DR means that the selected model resembles more and more a random walk

model as more observations are added; this is understandable since the break means that the Eonia rate never returns to levels which were normal before the break.

Figure 42: Forecast summary graphs - Eonia rate with break



4 Conclusion

The applications above provided illustration to using the `SpecEval` add-in in general as well as illustration of the interactive and iterative model development process the ass-in is meant to facilitate. However, the applications are not exhaustive, either in terms of the add-in functionality or in terms of the types of situations to which it can be applied to. Therefore, the list of applications might in future be enlarged, or existing applications might be provided with more detail.

A Complete programs for applications

A.1 Application 1

```
wfclose(noerr) .\..\data\data_for_application1.wf1
wfopen .\..\data\data_for_application1.wf1

%ip_first = ip.@first
%ip_last = ip.@last

smpl {%ip_first} {%ip_last}
graph fg_ip.line ip
fg_ip.addtext(t)

smpl @all
freeze(sp_ip_arma) ip.autoarma(tform=log, diff=1, select=sic
   ,agraph, atable, etable, eqname=eq_ip_arma) ip_f c
equation  eq_ip_arma.LS(ARMA=ML, ARMAOPT=KA) DLOG(IP) C MA
   (1) MA(2)

eq_ip_arma.speceval(noprompt)

smpl @all
equation eq_ip_arma2.LS(ARMA=ML, ARMAOPT=KA) DLOG(IP) C AR
   (1)

smpl @all
freeze(sp_ip_arma3) ip.autoarma(tform=log, diff=1, select=
   aic,agraph, atable, etable, eqname=eq_ip_arma3) ip_f c
equation eq_ip_arma3.LS(ARMA=ML, ARMAOPT=KA) DLOG(IP) C AR
   (1) AR(2) AR(3) AR(4) MA(1) MA(2) MA(3) MA(4)

eq_ip_arma.speceval(spec_list="eq_ip_arma*")

smpl @all
equation eq_ip_static.ls dlog(ip) c dlog(gdp)

eq_ip_static.speceval(spec_list="eq_ip_arma",use_names="t",
   graph_add_backtest="gdp[r]")

eq_ip_static.speceval(exec_list="medium stability")
eq_ip_static.speceval(spec_list="eq_ip_arma",use_names="t",
   graph_add_backtest="gdp[r]",keep_info="t",tfirst_test="
   2000q1")
eq_ip_static.speceval(spec_list="eq_ip_arma",use_names="t",
   graph_add_backtest="gdp[r]",keep_info="t",oos="f")
```

```
smpl @all
equation eq_ip_static2.ls dlog(ip) dlog(gdp)
eq_ip_static.speceval(spec_list="eq_ip_static*",
   horizons_forecast="1 2 4 8 16 40 80",horizons_graph="4 8
   40",alias="with without")

smpl @all
equation eq_ip_static_dummy.ls dlog(ip) c dlog(gdp)
   dum_recess*dlog(gdp)

eq_ip_static.speceval(spec_list="eq_ip_static_dummy",
   horizons_forecast="1 2 4 8",subsamples="2008q3-2009q4
   ,2011q3-2013q2",oos="t",alias="normal dummy")

smpl @all
equation eq_ip_static_exports.ls dlog(ip) c dlog(gdp)
   dum_recess*dlog(gdp) dlog(ex)

eq_ip_static.speceval(spec_list="eq_ip_static_dummy
   eq_ip_static_exports",horizons_forecast="1 2 4 8",
   subsamples="2008q3-2009q4,2011q3-2013q2",oos="f",alias="
   normal dummy exports",keep_forecasts="t")

eq_ip_static_exports.speceval(exec_list="medium
   decomposition",subsamples="2008q3-2009q4,2011q3-2013q2",
   oos="f")

eq_ip_static_dummy.speceval(scenarios="bl su sd")

eq_ip_static_dummy.speceval(exec_list="medium
   scenarios_individual",scenarios="bl su sd",tfirst_sgraph=
   "2006q1", tlast_scenarios="2025q4",graph_add_scenarios="
   gdp[r]",trans="deviation")

smpl @all
equation eq_ip_static_dummy2.ls dlog(ip) c dlog(gdp) (@movav
   (dum_recess,4)>0)*dlog(gdp)

eq_ip_static_dummy.speceval(spec_list="eq_ip_static_dummy2",
   exec_list="medium scenarios_individual",oos="f",scenarios
   ="bl su sd",tfirst_sgraph="2006q1", tlast_scenarios="2025
   q4",graph_add_scenarios="gdp[r]",trans="deviation",alias=
   "short_dummy long_dummy")
```

A.2 Application 2

```
wfclose(noerr) .\..\data\data_for_application2.wf1
wfopen .\..\data\data_for_application2.wf1

%cpi_first = cpi.@first
%cpi_last = cpi.@last

smpl {%cpi_first} {%cpi_last}
graph fg_cpi_level.line cpi
fg_cpi_level.addtext(t,sb) Level
graph fg_cpi_growth.line @pca(cpi) @pcy(cpi)
fg_cpi_growth.addtext(t,sb) Growth
fg_cpi_growth.setelem(1) legend(Month-on-month)
fg_cpi_growth.setelem(2) legend(Year-on-year)
graph fg_cpi.merge fg_cpi_level fg_cpi_growth
fg_cpi.align(2,1,1)

smpl @all
series s_cpi_growth = @pca(cpi)
freeze(sp_cpi_arma) s_cpi_growth.autoarma(tform=none, diff
   =0, maxar=4,maxma=4,select=aic,agraph, atable, etable,
   eqname=eq_cpi_arma) s_cpi_growth_f c
equation  eq_cpi_arma.LS @pca(CPI) C AR(1 to 3) MA(1 to 4)

eq_cpi_arma.setattr("selection_info") maxar=4,maxma=4,info=
   aic

eq_cpi_arma.speceval(auto_select="t",exec_list="medium
   stability")
copy sp_spec_evaluation sp_spec_evaluation_stability

eq_cpi_arma.speceval(auto_select="t",trans="growth",
   tfirst_graph="2009Q1")
```

A.3 Application 3

```
1
2 wfclose(noerr) .\..\data\data_for_application3.wf1
3 wfopen .\..\data\data_for_application3.wf1
4
5 pageselect monthly
6
7 %eonia_first = eonia.@first
8 %eonia_last = eonia.@last
9
10 smpl {%eonia_first} {%eonia_last}
11 graph fg_eonia_level.line eonia mrr
12 fg_eonia_level.addtext(t,sb) Level
13 graph fg_eonia_spread.line eonia-mrr
14 fg_eonia_spread.addtext(t,sb) Spread
15 graph fg_eonia.merge fg_eonia_level fg_eonia_spread
16 fg_eonia.align(2,1,1)
17
18 smpl @all
19 equation eq_eonia_static.ls eonia c mrr
20
21 eq_eonia_static.speceval(nodialog,exec_list="medium")
22 eq_eonia_static.speceval(nodialog,trans="spread",graph_bench
    ="mrr")
23
24 smpl @all
25 equation eq_eonia_arma.ls eonia c mrr ar(1)
26
27 eq_eonia_arma.speceval(nodialog)
28 eq_eonia_arma.speceval(trans="spread",graph_bench="mrr")
29
30 eq_eonia_static.speceval(trans="spread",graph_bench="mrr",
    graph_add_backtest="mrr-dr[R]")
31 eq_eonia_static.speceval(trans="spread",graph_bench="mrr",
    graph_add_backtest="log(er)[R]")
32
33 smpl @all
34 equation eq_eonia_structural.ls eonia-(mrr*(1-dum_er)) c dr*
    dum_er log(er)*dum_er
35
36 eq_eonia_structural.speceval(trans="spread",graph_bench="mrr
    ")
37 eq_eonia_structural.speceval(trans="spread",graph_bench="mrr
    ",oos="f")
```

```
%euribor_first = euribor.@first
%euribor_last = euribor.@last

smpl {%euribor_first} {%euribor_last}
graph fg_euribor_level.line euribor eonia mrr
fg_euribor_level.addtext(t,sb) Level
graph fg_euribor_spread.line euribor-mrr euribor-eonia eonia-mrr
fg_euribor_spread.addtext(t,sb) Spread
graph fg_euribor.merge fg_euribor_level fg_euribor_spread
fg_euribor.align(2,1,1)

smpl @all
equation eq_euribor_static.ls euribor c eonia

eq_euribor_static.speceval(trans="spread",graph_bench="eonia")

smpl {%euribor_first} {%euribor_last}
graph fg_spreads.line euribor-eonia libor-ffr


smpl @all
equation eq_spread_level.ls euribor-eonia c libor-ffr
equation eq_spread_diff.ls d(euribor-eonia) d(libor-ffr)

 eq_spread_level.speceval(trans="spread",graph_bench="eonia")
 eq_spread_diff.speceval(trans="spread",graph_bench="eonia")

 eq_spread_level.speceval(trans="spread",graph_bench="eonia",subsamples="2008M09-2009M09,2011M10-2012M06")
 eq_spread_diff.speceval(trans="spread",graph_bench="eonia",subsamples="2008M09-2009M09,2011M10-2012M06")

pageselect quarterly

smpl @all
equation eq_spread_ecm.ls  d(euribor-eonia) = c(1)*(euribor(-1)-eonia(-1)-c(2)) + c(3)*d(libor-ffr)

eq_spread_ecm.speceval(trans="spread",graph_bench="eonia",scenarios="bl s0 s4")


series libor_spread_bl = libor_bl-ffr_bl
series libor_spread_s4 = libor_s4-ffr_s4

eq_spread_ecm.speceval(exec_list="medium scenarios_individual",trans="spread",graph_bench="eonia",scenarios="bl s0 s4",graph_add_scenarios="libor_spread",baseline_forecast="t",original_forecast="f")
```

A.4 Application 4

```
wfclose(noerr) .\..\data\data_for_application4.wf1
wfopen .\..\data\data_for_application4.wf1

pageselect quarterly

%spi_first = spi.@first
%spi_last = spi.@last

smpl {%spi_first} {%spi_last}
graph fg_spi_level.line spi
fg_spi_level.addtext(t,sb) Level

smpl {%spi_first} {%spi_last}
graph fg_spi_loglevel.line spi
fg_spi_loglevel.axis(l) log
fg_spi_loglevel.addtext(t,sb) Log-level

smpl {%spi_first} {%spi_last}
graph fg_spi_ratio1.line spi/cpi
fg_spi_ratio1.addtext(t,sb) Ratio - SPI to CPI

smpl {%spi_first} {%spi_last}
graph fg_spi_ratio2.line spi/gdp
fg_spi_ratio2.addtext(t,sb) Ratio - SPI to GDP

graph fg_spi.merge fg_spi_level fg_spi_loglevel
   fg_spi_ratio1 fg_spi_ratio2
fg_spi.align(2,1,1)
fg_spi.display
```

```
smpl @all
equation eq_spi_ar1.ls spi/gdp c spi(-1)/gdp(-1)

eq_spi_ar1.speceval(oos="f")

eq_spi_ar1.speceval(oos="f",trans="log")

eq_spi_ar1.speceval(oos="f",trans="ratio",graph_bench="gdp")

equation eq_spi_ar2.ls spi/gdp c spi(-1)/gdp(-1) spi(-2)/gdp(-2)
equation eq_spi_regarma10.ls spi/gdp c spi(-1)/gdp(-1) ar(1)
equation eq_spi_regarma01.ls spi/gdp c spi(-1)/gdp(-1) ma(1)
equation eq_spi_regarma11.ls spi/gdp c spi(-1)/gdp(-1) ar(1) ma(1)

eq_spi_ar1.speceval(spec_list="eq_spi*",use_names="t",oos="f")


eq_spi_ar1.speceval(scenarios="bl su sd",tfirst_sgraph="1970q1")
eq_spi_ar1.speceval(scenarios="bl su sd",tfirst_sgraph="1970q1",trans="log")
eq_spi_ar1.speceval(scenarios="bl su sd",tfirst_sgraph="1970q1",trans="ratio",graph_bench="gdp")
```

A.5 Application 5

```
wfclose(noerr) .\..\data\data_for_application5.wf1
wfopen .\..\data\data_for_application5.wf1

pageselect monthly

string id_mrr_rw = "mrr = mrr(-1)"

id_mrr_rw.speceval(nodialog)

id_mrr_rw.speceval(nodialog,keep_forecasts="t")

smpl @all
equation eq_eonia_arma.ls eonia c mrr ar(1)

eq_eonia_arma.speceval(FORECASTED_IVARIABLES="mrr",trans="spread",graph_bench="mrr")
```

A.6 Application 6

```
wfclose(noerr) .\..\data\data_for_application5.wf1
wfopen .\..\data\data_for_application5.wf1

pageselect monthly

string id_mrr_rw = "mrr = mrr(-1)"

id_mrr_rw.speceval(nodialog)

id_mrr_rw.speceval(nodialog,keep_forecasts="t")

smpl @all
equation eq_eonia_arma.ls eonia c mrr ar(1)

eq_eonia_arma.speceval(FORECASTED_IVARIABLES="mrr",trans="
   spread",graph_bench="mrr")
```

A.7 Application 7

```
wfclose(noerr) .\..\data\data_for_application7.wf1
wfopen .\..\data\data_for_application7.wf1

pageselect monthly

smpl @all
equation eq_eonia.ls eonia c mrr dr ar(1) ma(1)

eq_eonia.speceval(custom_reest="t",trans="spread",
   graph_bench="mrr",tfirst_test="2005M01")

eq_eonia.speceval(custom_reest="f",trans="spread",
   graph_bench="mrr",tfirst_test="2005M01")
```

A.8 Application 8

```
wfclose(noerr) .\..\data\data_for_application8.wf1
wfopen .\..\data\data_for_application8.wf1

pageselect monthly

smpl @all
equation eq_eonia.ls eonia c dr ar(1)

eq_eonia.setattr("selection_info") maxar=4,maxma=4,info=SIC
eq_eonia.speceval(auto_select="t",tfirst_test="2005m01",
   exec_list="forecasts stability",keep_eqs="t",keep_info="t
   ")

%sub_horizons = "6 12 24"
!max_horizon = @val(@word(%sub_horizons,@wcount(%
   sub_horizons)))

matrix(sc_forecastp_n,@wcount(%sub_horizons)) m_irf

%fqs = ""

for !fp = 1  to sc_forecastp_n

  %fq = tb_forecast_numbers(1+!fp,2)

  eq_eonia_reest{%fq}.arma(type=imp,imp=1,hrz={!max_horizon
     },save=v_irf_{!fp})
  close eq_eonia_reest{%fq}

  for !h = 1 to @wcount(%sub_horizons)
    %h = @word(%sub_horizons,!h)
    m_irf(!fp,!h) = v_irf_{!fp}({%h})
  next

  %fqs  = %fqs +%fq+ " "
next


smpl {st_tfirst_backtest} {st_tlast_backtest}
freeze(gp_irf) m_irf.line
gp_irf.setobslabels {%fqs}

for !h = 1 to @wcount(%sub_horizons)
  %h = @word(%sub_horizons,!h)
  gp_irf.setelem(!h) legend({%h}-th period impulse response)
next
```