FcastDecomp	equation Proc

Decompose forecast into individual forecast drivers.

`fcastdecomp` creates graph that decomposes forecast for left-hand side variable into individual drivers, i.e. right-hand side variables multiplied by coefficient. If two scenarios are specified then graph will display difference in forecast drivers between first and second scenario.

Syntax

{%equation}.fcastdecomp(options)

Options

SCENARIOS=*arg(s)*	Scenario alias or space separated list of two scenario aliases. If two aliases are specified then graph will display forecast drivers of first scenario minus forecast drivers of second scenario. **Note**: If no alias is entered then forecast decomposition will use the actual (historical) series. **Note**: The program checks existence of the series with alias for each driver. If any of the underlying series with alias does not exist then it will use the series without alias. **Note**: User can specify secondary alias in square brackets (e.g. "_s1[_bl]" means that program will use '_s1' as alias, but if series with such alias is not found then the program will use '_bl' as alias). **Note**: Should include "_" if the scenario alias does.
INCLUDE_ADDF=*arg* *(default="f")*	Indicator whether add-factor series should be included in the graph. Arg may be "t" and "f".
INCLUDE_SUM=*arg* *(default="f")*	Indicator whether sum of all drivers should be included in the graph. Arg may be "t" and "f".
SAMPLE=*arg* *(default=current workfile sample)*	Sample of the forecast decomposition graph. **Note:** The graph refers to actual workfile series, so that the graph can be changed to manual or automatic update to vary the sample in GUI.
GRAPH_NAME=*arg* *(default="gp_fd")*	Name of graph in workfile.
KEEP_TABLE=*arg*	Indicator whether information table should be

(default="f")	stored. **Note**: The table is stored under name 'tb_forecast_decomposition'.
USE_TABLE=*arg* *(default="f")*	Indicator whether existing information table ('tb_forecast_decomposition') should be used.
PROMPT	Indicator that user dialog should be issued. Relevant when executed from program.

Examples

The commands

```
equation eq01.ls d(ip) c d(gdp) ir
eq01.fcastdecomp(nodialog)
```

will estimate equation object EQ01 linking changes in IP to changes in GDP and changes in IR, as well as constant. It will then create graph which will include 4 series – D(IP) as dependent variable series, and constant, D(GDP) and IR as driver. The graph will have sample of the current workfile page sample.

If instead the user calls following command:

```
eq01.fcastdecomp(sample="2020q1 2022q4",alias_list="_bl",
include_addf="t")
```

then the graph will have sample starting in 2020q1 and ending in 2022q4, and it will display following 5 series – D(IP_bl) as dependent variable, and constant, D(GDP_BL) and D(IR_BL) together with IP_A as drivers.

Finally, if following command is issued

```
eq01.fcastdecomp(sample="2020q1 2022q4",alias_list="_s1
_bl")
```

then the graph will include 3 series – D(IP_S1)-D(IP_BL) as dependent variable, D(GDP_S1)-D(GDP_BL) and D(IR_S1))-D(IR_BL) as drivers.

See next section for detailed explanation and section after that for illustrations.

Decomposition explanation

Consider your equation has following form:

$$d(y_t) = \beta_0 + \beta_1 x_t + \beta_2 d(z_t) + a_t$$

Then in each period the value of $d(y_t)$ is equal to constant β_0, effect of the two regressors x_t and z_t, and the value of add-factor a_t. The effect of each regressor is then equal to its value in given period multiplied by the associated regression coefficient. The language used here is that the forecast for $d(y_t)$ can be decomposed into 4 drivers of the forecast: $(1)\ \beta_0, (2)\beta_1 x_t, (3)\ \beta_2 d(z_t)$ and $(4)a_t$. The forecast decomposition graph then has 5 series: one capturing the forecast for $d(y_t)$ and 4 corresponding to the forecast drivers.

Note that in general one can do forecast decomposition only for the variables in their transformation (e.g. $d(y_t)$ rather than y_t), since otherwise the forecast might not be additively separable and/or forecast effects might accumulate. This also means that in presence of lagged

dependent variables the drivers show only the period effects, not the cumulative effects of individual regressors.

Output illustration

Consider following equation that will be used for the illustration:

```
equation eq.ls dlog(ip) c dlog(ip(-1)) dlog(gdp)
libor*dum_fincrisis
```

The equation includes 4 drivers: *constant*, *dlog(ip(-1))*, *dlog(gdp)*dum_recess* and *libor*dum_fincrisis*. The underlying variables are in order of appearance Czechia industrial production, Czechia real GDP, 3-month Libor and financial crisis dummy. Below is regression output for the equation:[1]

Dependent Variable: DLOG(IP)
Method: Least Squares
Date: 03/13/21 Time: 14:04
Sample (adjusted): 1999Q1 2019Q4
Included observations: 84 after adjustments

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	0.000716	0.002484	0.288121	0.7740
DLOG(IP(-1))	0.084046	0.097831	0.859089	0.3929
DLOG(GDP)	1.046453	0.270837	3.863774	0.0002
LIBOR*DUM_FINCRISIS	-0.017057	0.006197	-2.752421	0.0073

R-squared	0.450446	Mean dependent var	0.007765
Adjusted R-squared	0.429838	S.D. dependent var	0.020319
S.E. of regression	0.015343	Akaike info criterion	-5.469919
Sum squared resid	0.018832	Schwarz criterion	-5.354165
Log likelihood	233.7366	Hannan-Quinn criter.	-5.423387
F-statistic	21.85757	Durbin-Watson stat	1.763775
Prob(F-statistic)	0.000000		

The add-in can be used for three related but distinct types of forecast decomposition:

1. Decomposition of forecast in historical sample
2. Decomposition of scenario forecast
3. Decomposition of difference between two scenario forecasts

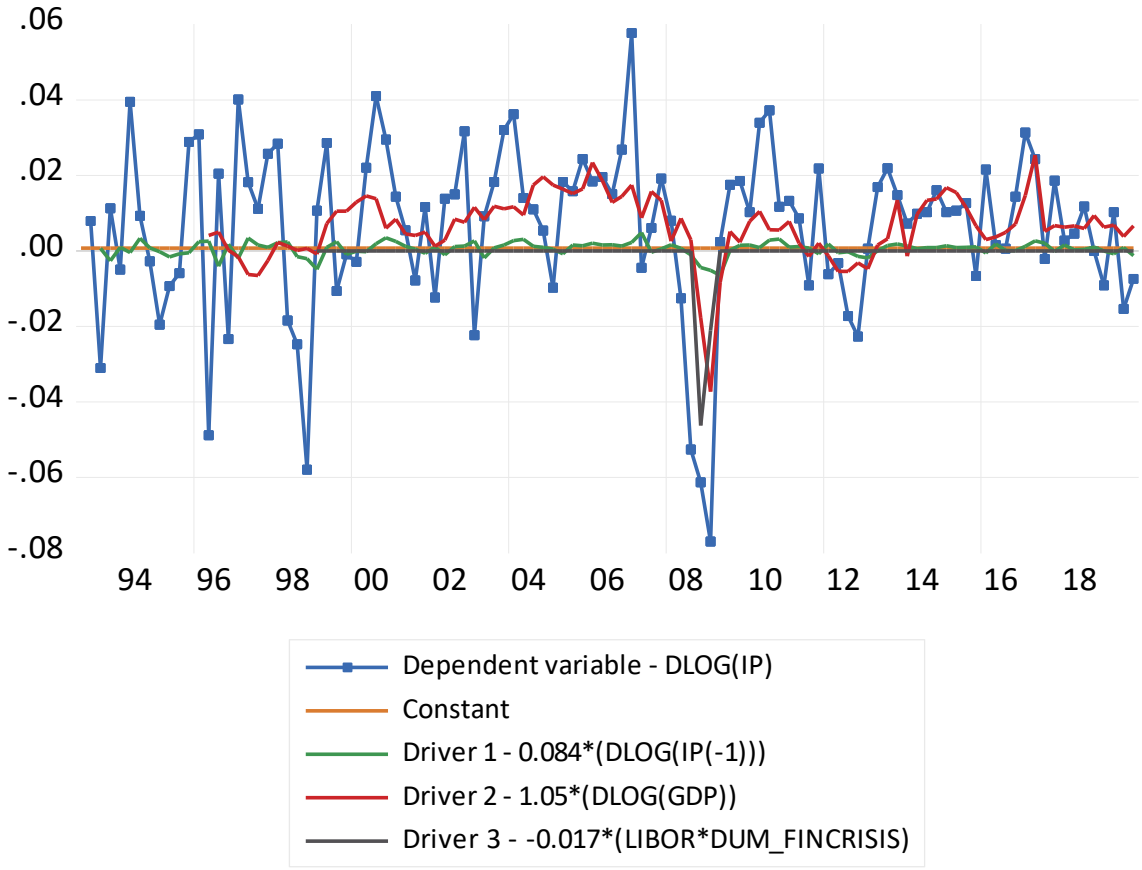
The three decompositions are achieved by different settings for the ‘scenarios’ option and are discussed in turn.

If the options ‘scenarios’ is left empty then add-in will use actual historical series (i.e. series without alias) for the decomposition. Consider following add-in execution from the equation:

```
eq.fcastdecomp(noprompt)
```

As a result the add-in will create following graph:

[1] Data for this illustration and program executing it is part of the add-in package.

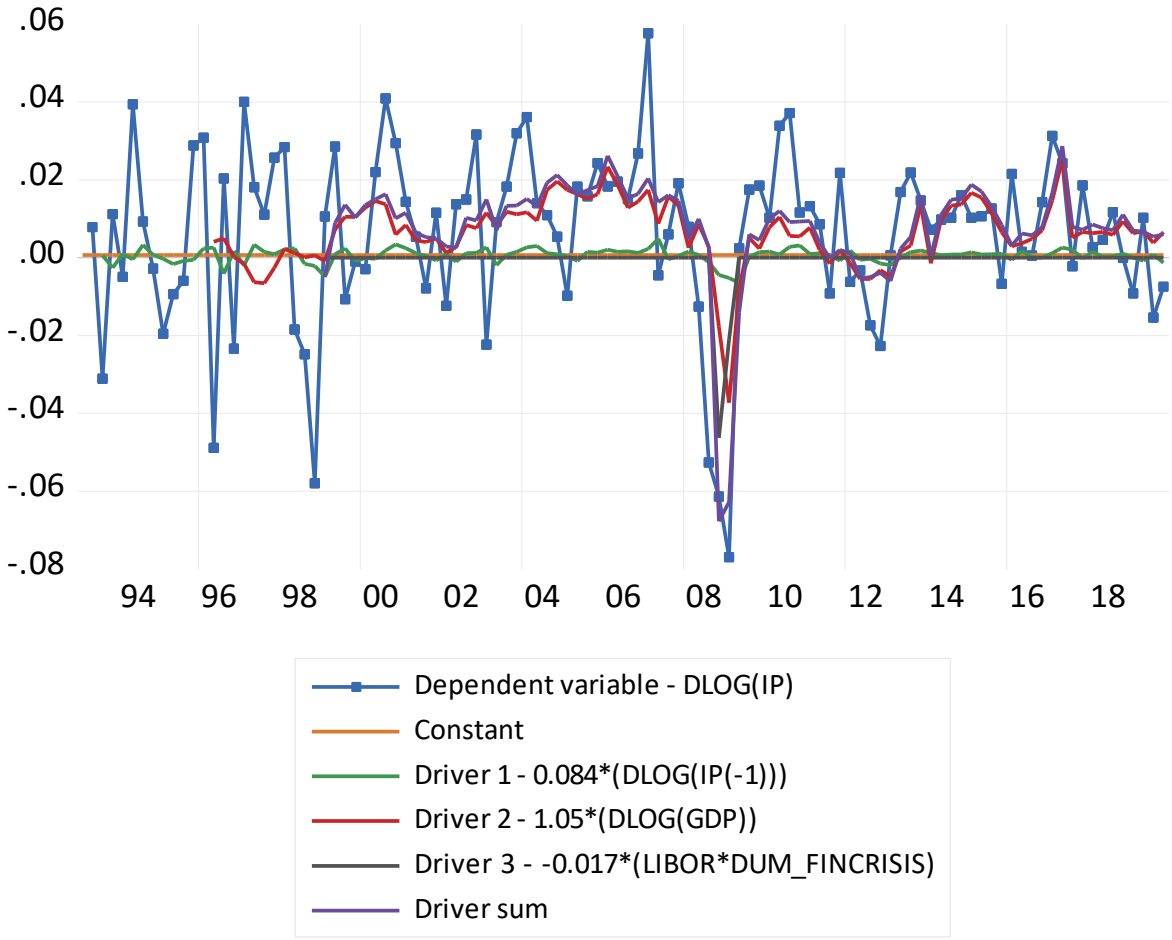


The graph contains 5 series: one showing the dependent variable, and 4 showing the forecast drivers. The orange line captures the effect of constant and of course does not feature any variability, and it acts only as level shift for the forecast. Meanwhile, the remaining drivers vary over time, even though the last driver shows variance only in one period (2008-2009). Importantly, the individual repressors are multiplied by the associated coefficients so that all series have the same scale irrespective of the units of the underlying variable.

In terms of interpretation, the forecast driver graph shows that the driver corresponding to lagged dependent variable copies the dependent variable with one period delay, as expected, but has much lower volatility so that its influence is small. This of course reflects the relative small coefficient for the regressor. Meanwhile, the driver for GDP is clearly closely correlated with the dependent variable and important in terms of magnitude. Finally, while the driver for the 3-month Libor has only single period of non-zero values, it has large effect in this period that corresponds well to behavior of the dependent varaible.

The graph above has one important drawback: the series with dependent variable captures the actual historical values of the variable, i.e. it does not represent actual forecast. One can address that by including the sum of drivers as additional driver as follows:

```
eq.fcastdecomp(include_sum="t")
```

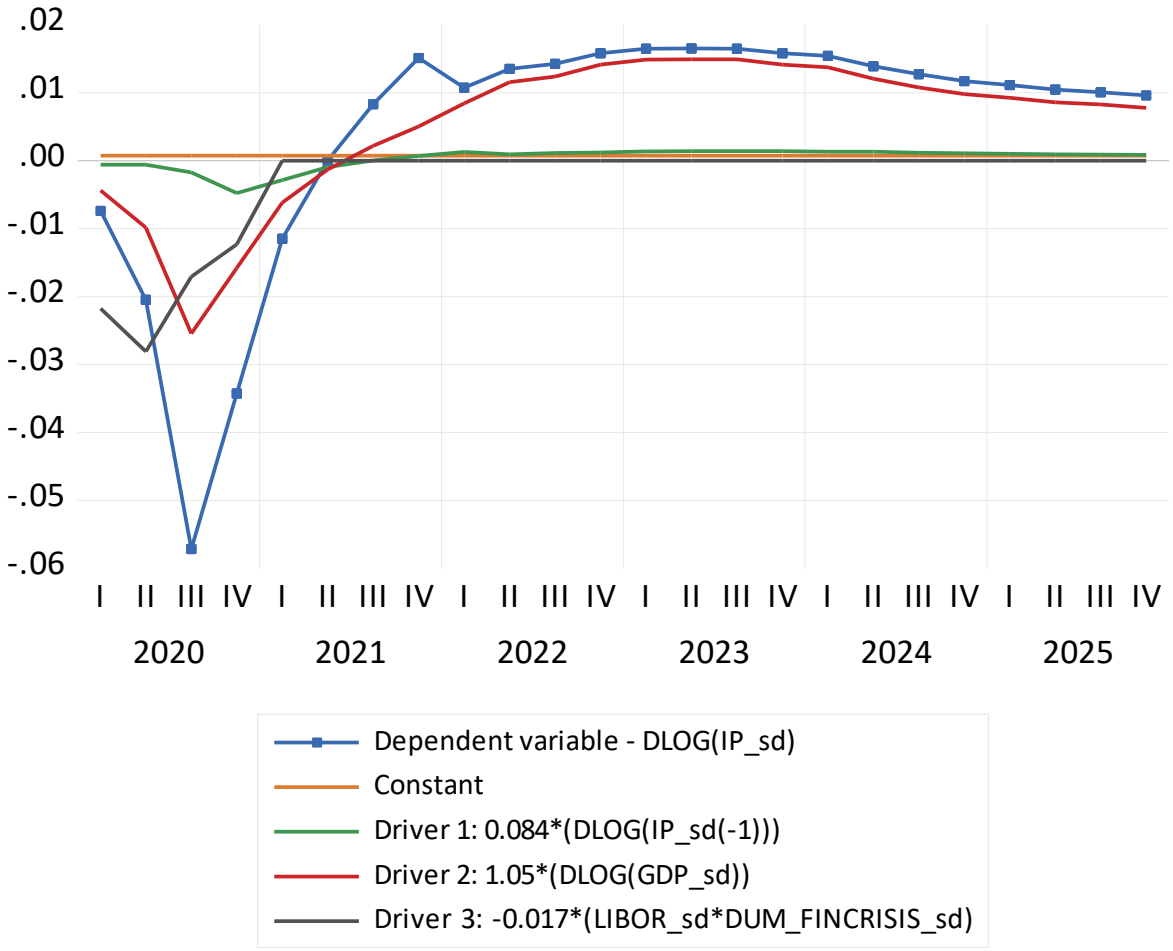


The graph is identical as before with exception of the last series, which shows the sum of all the 4 drivers. This allows user to compare the actual dependent variable with what the equation would forecast for it, as well as decompose that forecast into individual drivers.

If single scenario is specified then the add-in creates graph decomposing that scenario forecast for the dependent variable into individual forecast drivers. Consider following execution:

```
eq.fcastdecomp(scenarios="_sd",sample="2020q1 2025q4")
```

The resulting forecast decomposition graph is below. The graph again includes 5 series – one for dependent variable and 4 forecast drivers – but this time around the values corresponding to values under scenarios '_sd'. The graph shows that the log-difference of industrial production is in beginning very negative, but then it turns positive before settling at normal values. This of course corresponds to deep recession and following recovery, reflecting the fact that the 'SD' scenario is downside scenario. The constant again does not add any variation to the forecast, while the drier corresponding to lagged dependent variable again copies the dependent variable and again features relatively little variation. The two major sources of variation in the forecast are the drivers 2 and 3. The dependent variable most closely follows the second driver, but the correspondence is far from exact. In first few quarters the third driver exerts large influence on the forecast. Meanwhile, throughout the forecast the period effects of the second and third drivers is propagated into future periods though the effect on lagged dependent variable.

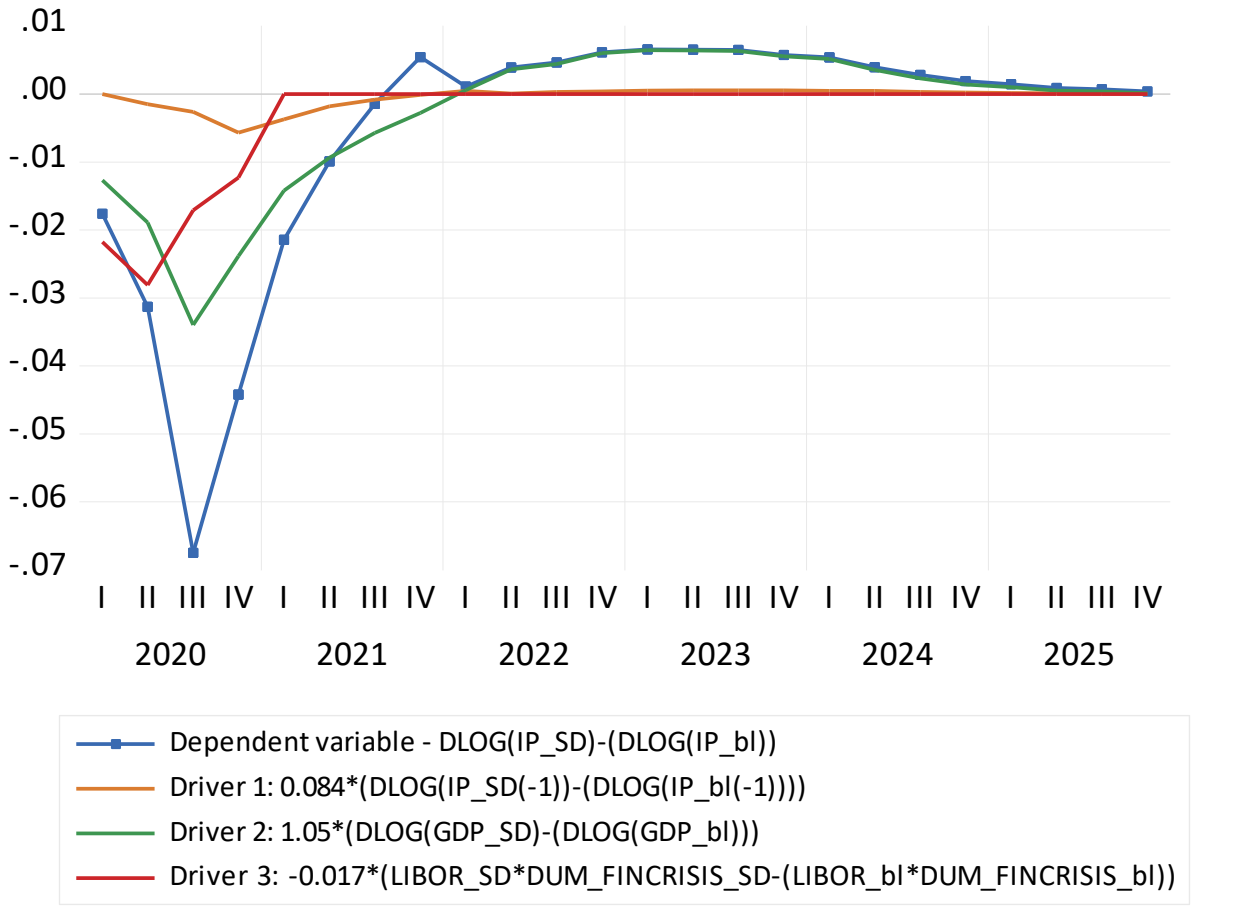


Finally, if user specifies 2 scenarios the add-in will create forecast decomposition difference graph, subtracting the forecast drivers values based on of second scenario from values based on first scenario. Consider following execution:

```
eq.fcastdecomp(scenarios="_sd _bl",sample="2020q1 2025q4")
```

The resulting forecast difference decomposition graph is below. The As before, the first series corresponds to dependent variable. But this time around the series displays difference between values of dependent variable under the '_sd' scenario and the '_bl' scenario. While the profile is fairly similar – reflecting the fact the under the '_bl' scenario the forecast does not feature much variation – the level is shifted downwards – reflecting the fact that the log-difference is positive under '_bl' scenario. Therefore, the series displays the differences for the dependent variable under the two scenarios. The same applies to the three drivers.[2] Hence the forecast difference decomposition graph decomposes the difference in values of dependent variable in two scenarios into differences between values of the individual drivers/regressors. This is especially useful in situation when the base scenario features lot of variation in either the dependent variable or the regressors which complicate the analysis of single scenario forecast.

[2] Note that constant is not included, since difference of two identical constants is zero.



Apart from the forecast decomposition graph the add-in can also store the table that contains list of the included drivers, both in standalone form (second column) and in difference form (third column) – see table below. This information can be further leveraged in custom decompositions.

Driver #	Stand-alone driver	Difference driver
DV	DLOG(IP_salias)	DLOG(IP_salias1)-(DLOG(IP_salias2))
0	0.00072	
1	0.084*(DLOG(IP_salias(-1)))	0.084*(DLOG(IP_salias1(-1))-(DLOG(IP_salias2(-1))))
2	1.05*(DLOG(GDP_salias))	1.05*(DLOG(GDP_salias1)-(DLOG(GDP_salias2)))
3	-0.017*(LIBOR_salias*DUM_FINCRISIS_salias)	-0.017*(LIBOR_salias1*DUM_FINCRISIS_salias1-(LIBOR_salias2*DUM_FINCRISIS_salias2))

Contact Information

Please send any questions, comments, criticisms, or complaints to the dedicated Eviews forum or privately to kamil.kovar@cerge-ei.cz or. If you'd like to contribute to the project, please feel free to send a pull request to https://github.com/CrisisStudent/FcastDecomp.

Additional notes

The add-in is separated into settings&execution program file and subroutines program file. This allows developers to repurpose the subroutine(s) outside of the add-in.