ProviewR

OPEN SOURCE PROCESS CONTROL

MPC controller

2020 02 27

Table of Contents

Introduction

This document describes the implementation of an MPC controller in ProviewR.

MPC controller

An MPC (Model Predictive Control) controller contains a model of the process and uses this model to calculate the future response for different controller outputs. The optimal output, ie the output that gives the smallest difference between the set value and process value, is then selected and used as the next output for the controller.

In the example below, the level control from the demo project, the MPC is compared with a traditional PID controller.

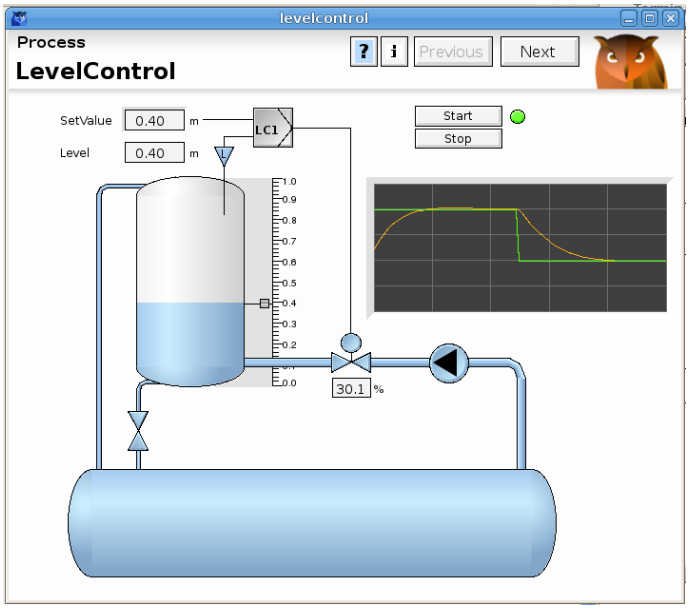

Fig 1 Level control

A step response from a PID is displayed in the figure below. The yellow output curve is mainly proportional to the error, ie the difference between the set value and process value. It reaches a high value immediately after the step and then gradual decreases as the error decreases.

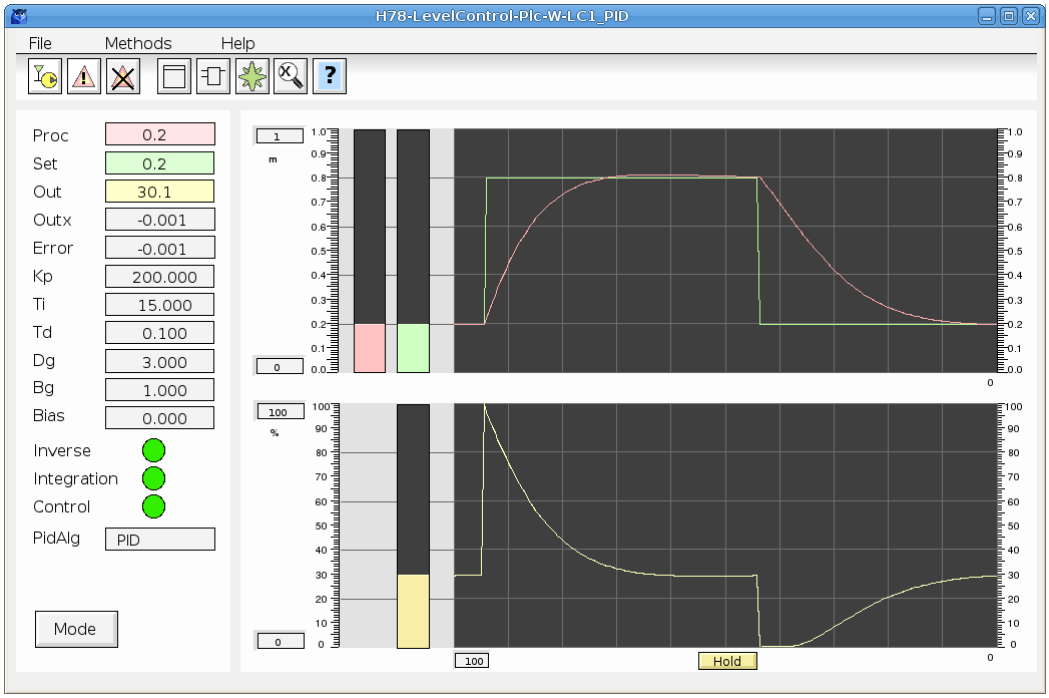

Fig 2 Step response for PID

The MPC controller below has an entirely different approach. The output reaches the maximum value right after the step, and stays there until shortly before the process value reaches the set value.

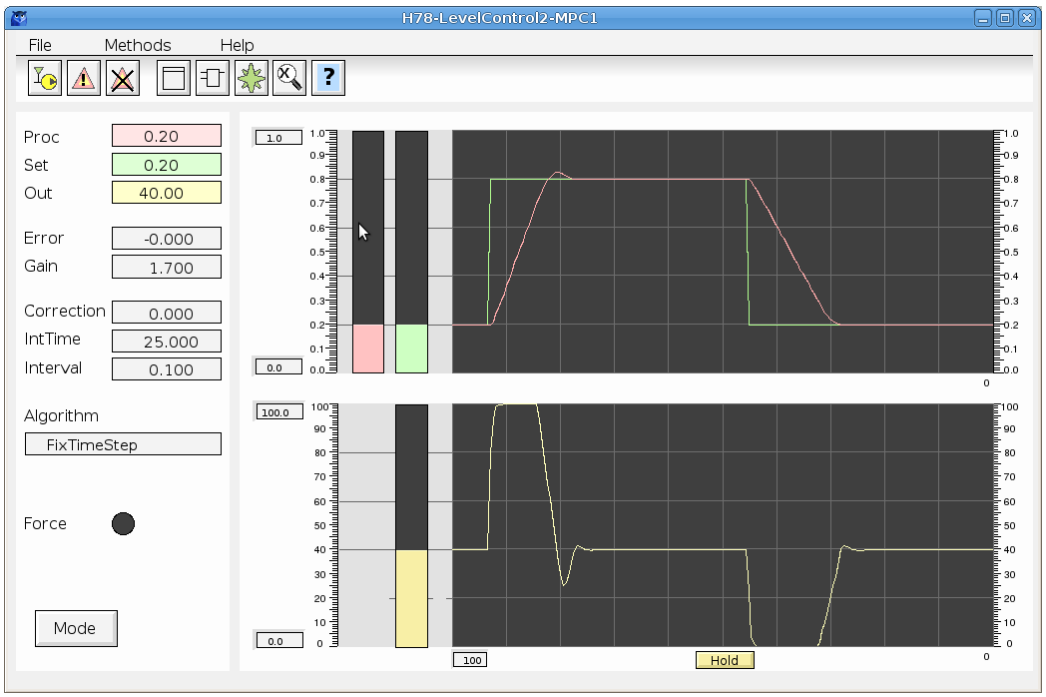

Fig 3 Step response for MPC

This is very close the the optimal behavior for a controller and shows the advantage of having the capability to foresee the future.

Prediction

The prediction is made iteratively by using the model to calculate the process value from the output value a number of time steps ahead. In the example below, 5 time steps is used, and in the initial step, 5 paths are calculated from the current output value. In the next and following time steps, each

path is split in another five paths. This makes a total sum of $5^5 = 3125$ paths, and for each path the corresponding process value path is calculated. In the lower graph the setvalue is drawn, which in this case is a step function, and a number of calculated process value paths. The most optimal path is regarded to be the one with the smallest area between the setvalue and processvalue (the gray area in the figure), and this path is chosen for the next iteration. In the next iteration (right graph below), the spread of the output paths is narrowed around the previously optimal path, and a new optimal path is calculated. Further iterations can give even more optimal paths.

In the end, it's actually only a small fraction of the first timestep that is used. The new output is set to the value at the next scantime of the controller, which is normally much shorter than a timestep. Then in the next scan, a new optimal path is calculated and the initial fraction of this path is used till the next scan.

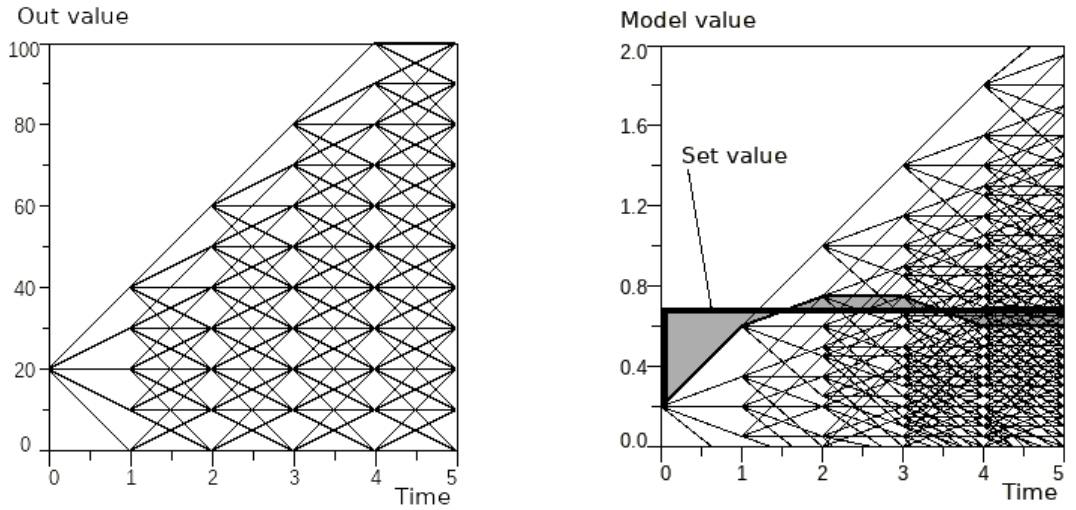

Fig 4 Outvalue and corresponding calculated model value

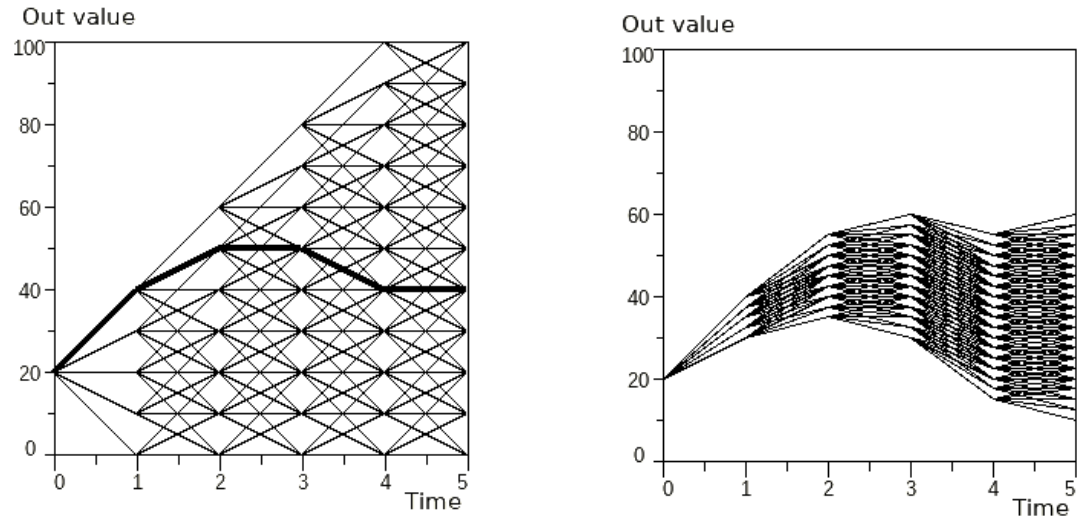

Fig 5 Optimal outvalue path and outvalue paths for next iterations

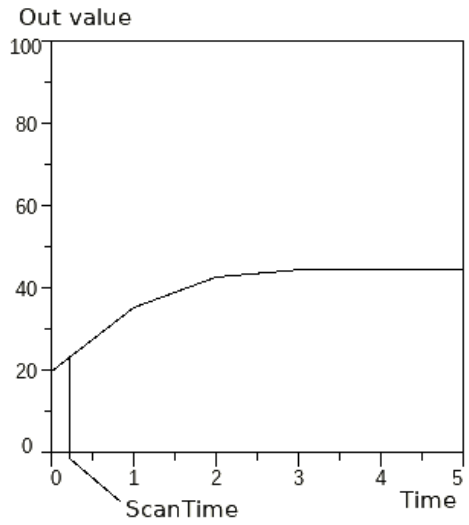

Fig 6 Final optimal output path

The figure below shows the prediction for the level control shortly after a change of the set value. The right diagrams shows the predicted process value (red) and out value (yellow). Shortly before the process value reaches the set value, the optimal out value path is to decrease the out value with the fastest possible ramp.

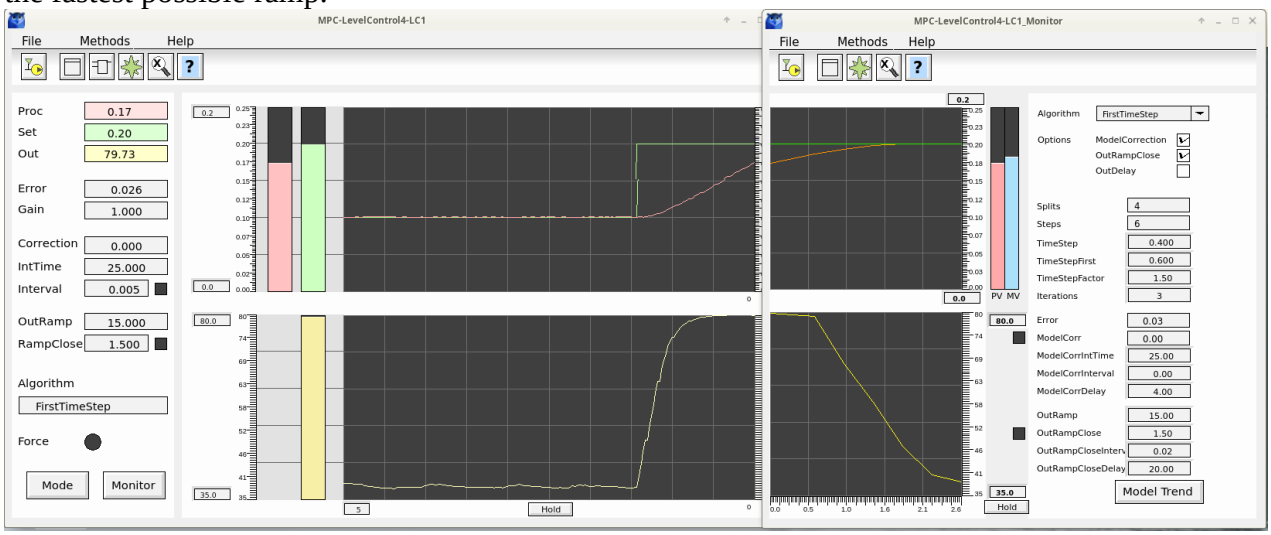

Fig 7 Level control prediction

A snapshot a few seconds later shows the real outcome. Note that the time scale for the predictive curve and the past curve differs.

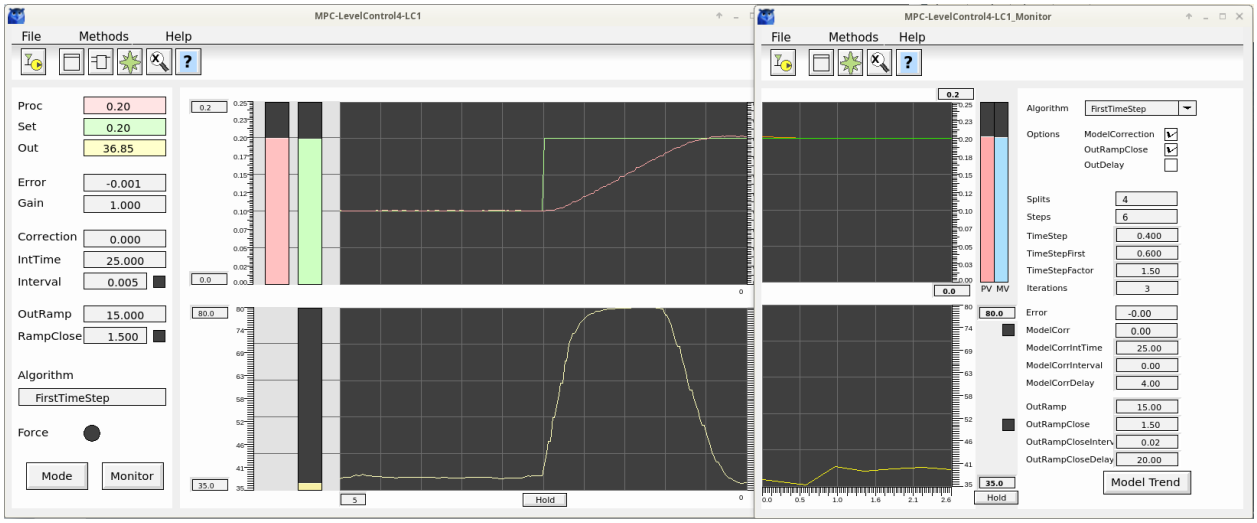

Fig 8 Level control outcome

Model

To calculate the process value for the different output paths, a model is used. The model takes the output value as input, but can also take a number of other parameters. In the level control example above, both the speed of the pump and the output flow are inputs to the model.

Two different types of models are implemented for the MPC, linear regression and a neural network with an MLP. These are described later in more detail.

Prediction configuration

Time steps and splits

The prediction is configured with the number of time steps, and the number of splits in each time step. This is done in the attributes *Steps* and *Splits*. Note that the number of paths are growing exponentially, both with the number of splits and the number of steps, and so is also the required CPU capacity. In the figure below there are 5 time steps of equal size, and 5 splits.

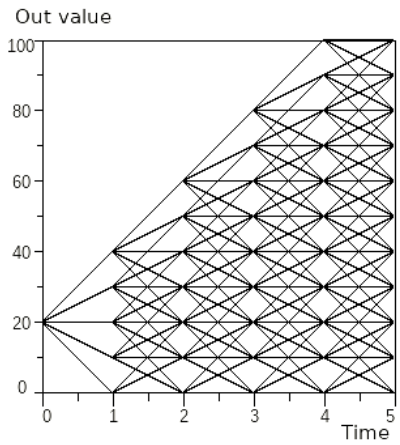

Fig 9 Time steps and splts

The sum of the time steps is called the horizon, and this is how far in the future the controller can see. There are three algorithms with different size of the time steps.

First there is *FixTimeStep* where all the steps are of equal size, and the size is set in the *TimeStep* attribute.

Then there is *FirstTimeStep* where the first step is different from the others. The first is set in *TimeStepFirst* and the others will have the size of *TimeStep*. A shorter first time step will speed up the controller, and a longer will slow it down and make it more stable.

Finally there is the *ProgressiveTimeStep*, where the size of the first step is *TimeStep* and the following time steps have the size of the previous multiplied with the factor in *TimeStepFactor.* If *TimeStep* is 1 second and *TimeStepFactor* is 1.5, the time steps will be 1, 1.5, 2.25, 3.37, 5.1 with a horizon of 13.2 seconds.

The algorithm is set in the *Algorithm* attribute and can be modified in runtime. Also the *TimeStep, TimeStepFirst* and *TimeStepFactor* can be modified in runtime. *Splits* and *Steps* though can not be modified in runtime.

Iterations

The number of iteration to perform are configured in the *Iterations* attribute. The default value is 3 and the CPU load will increase proportionally with the number of iterations.

Tuning

Model correction

It's inevitable that the model, at least for some combinations of parameters, will differ from the real process value. In the predictive algorithm there is nothing that compensates for this, the controller will strive for the somewhat erroneous model value and stay there. To remove this residual error, an integration term is added to the model value. The term integrates the error, ie the difference between the process value and the set value. If this term is active all the time, it will in high degree disturb the prediction result. There for a window can be specified where the integration is active, and also a delay time where the integration is activated a certain time after the process value has entered the window.

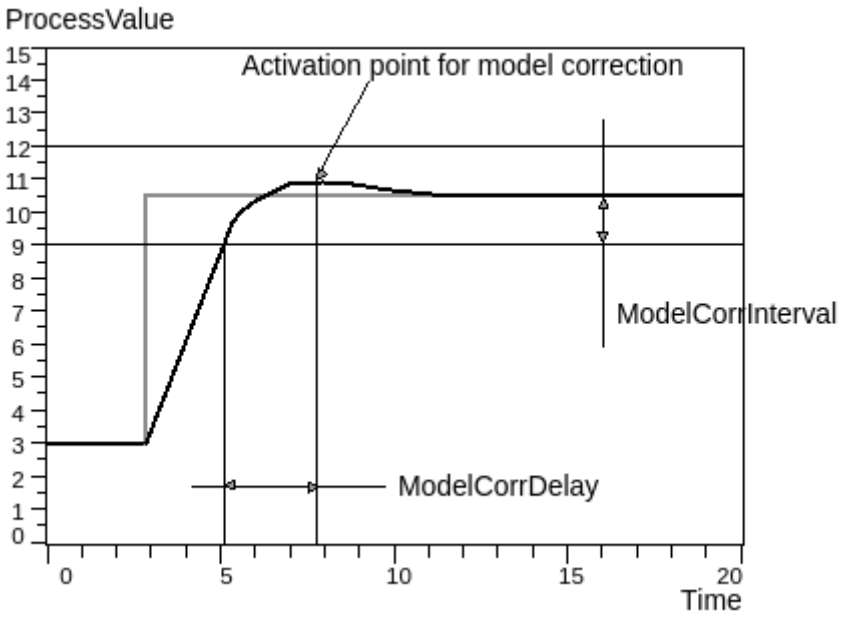

Fig 10 Model correction

The model correction is configured with the attributes

Out value ramp close

One way of tuning the controller is to adjust the maximum ramp of the out value. If the response from the controller is to fast, small oscillations can occur also in steady states. A lower value of the max ramp can remove these oscillations. The max ramp is set in the *OutRamp* attribute.

To keep the controller fast it is possible to reduce the ramp when the controller as reached a steady state. This is done with the *OutRampClose* function. When the process value is close to the set value, the out value ramp is reduced. Also a time delay can be configured, see figure below.

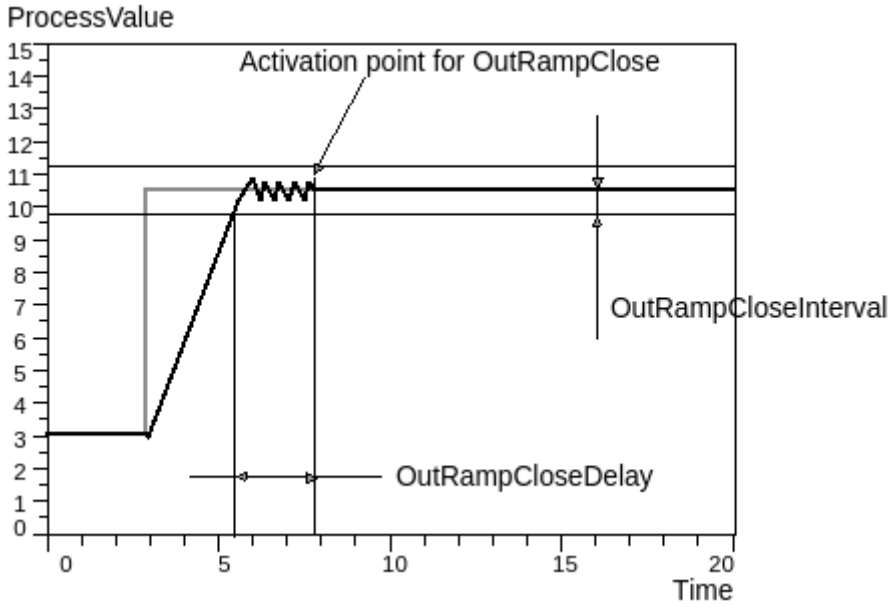

Fig 11 Out value ramp close

A range around the set value is configured with the *OutRampCloseInterval* attribute. When the process value is inside this range, and the *OutRampCloseDelay* time has elapsed, the maximum ramp is reduces to the value in the *OutRampClose* attribute.

Out value delay

If there is a delay between the time the out value is set, to the time it is affecting the process, this delay can be configured in the *OutDelay* attribute. This means that the out value from the controller, is fetched from an earlier point in the prediction.

The delay time should be less than the first time step.

Prediction attributes

Attribute	Description
Steps	Number of time steps in the prediction. The number of paths, and the CPU load will grow exponentially with Steps.
Splits	Number of new OutValue paths in each time step.
Algorithm	The algorithm used in the predictive calculation. - FixTimeSteps: all time steps are of equal length. The lenght is specified in attribute TimeStep. - ProgressiveTimeStep_ the first time step has the size specified in the attribute

	TimeStep, the next timesteps has the sizes of the previous multiplied by the value in TimeStepFactor. - FirstTimeStep: The first step has the length in FirstTimeStep, and the following the length in TimeStep.
TimeStep	Length of the time step in the predictive calculation in seconds. For algorithm FixTimeStep all the timesteps has this size. For algorithm FirstTimeStep all the timesteps has this size, except the first that has the size in the attribute FirstTimeStep. For algorithm ProgressiveTimeStep the first timestep has this size, and the following are increased by the TimeStepFactor value.
TimeStepFirst	The length of the first time step when algorithm FirstTimeStep is used.
TimeStepFactor	Factor for increment of the time step when algorithm ProgressiveTimeStep is used.
Iterations	Number of iterations.

Model correction attributes

Attribute	**Description**
ModelCorr	Model correction value.
ModelCorrIntTime	Model correction integration time.
ModelCorrInterval	Distance from the SetValue when the model correction should be activated.
ModelCorrDelay	Delay time for activation of the correction.
ModelCorrActive	Indicates that model correction is active.

Out value ramp close attributes

Attribute	Description
OutRampClose	Maximum ramp of out value when the error is less than OutRampCloseInterval. Minor fluctuations at steady state can in some cases be reduced by increasing the ramp. Note that the interval should be kept small as this restriction is not part of the prediction.
OutRampCloseInterval	Distance from the SetValue when the output ramp is restricted to OutRampClose.
OutRampCloseDelay	Delay of out value. Is used when there is a delay in the actuator for the control signal. The OutDelayTime should not exceed the size of the first time step in the prediction. It should also not exceed OutDelayMaxTime.
OutRampCloseActive	Indicates that out ramp close is active.

Linear regression model

In linear regression the model value y, is calculated from a number of input values x_1 - x_n . y is supposed to have linear dependencies of the input values with the formula

$y = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_3 + \ldots + a_n x_n$

where are a_0 - a_n constants that are calculated when the model is created. In our model, y is the process value, x_1 the out value from the controller, and x_2 - x_n other parameter that influences the process value.

Many processes are not linear, but by linearizing the the input parameters, linear regression can be used also for these processes.

Level control example

In this example we will create a model a level control. The difference from the level control example above is that we are now controlling the speed of the pump P1 instead of the control valve.

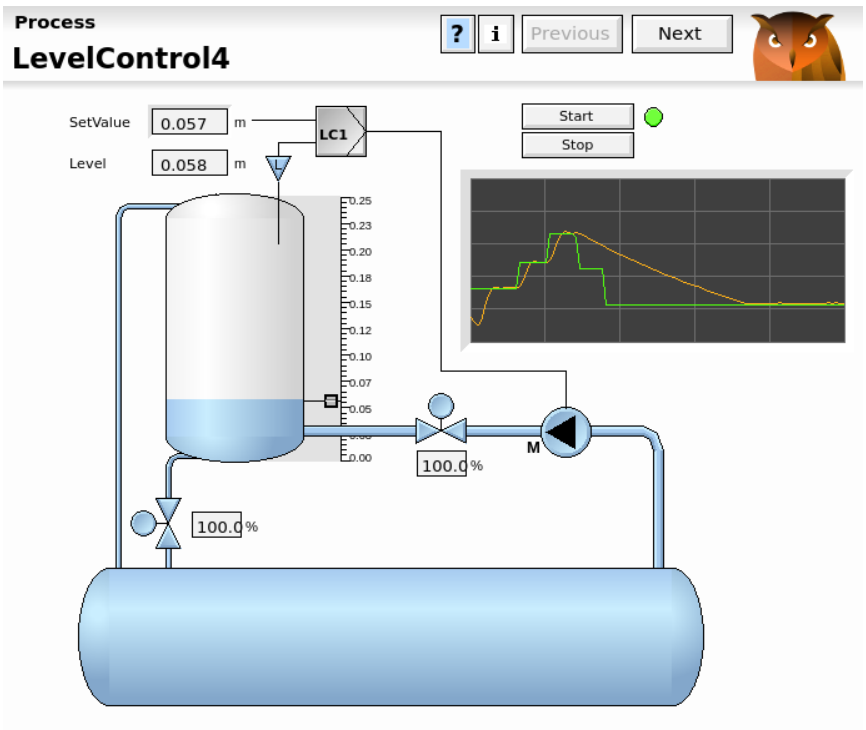

Fig 12 Level control

The level can be calculated from the integral for the input flow F_{in} minus the integral for the output flow F_{out}

$$L = a_1 \int_0^t F_{in}\,dt - a_2 \int_0^t F_{out}\,dt$$

We are setting the speed of the pump from the controller, and unfortunately it's not linear to the input flow, so we have to insert a conversion table to convert from speed to flow.

We assume that the out flow is proportional to output valve position.

The linearization we have to perform, is first to convert the pump speed to flow, then integrate the flow and also integrate the out valve position.

We start with making an Xtt log while driving the pump at different speeds. The tank level, the reference to the pump, and the order to the outflow control valve are logged. The log file is opened in the multivariate analyzer and the curves are shown in the figure below.

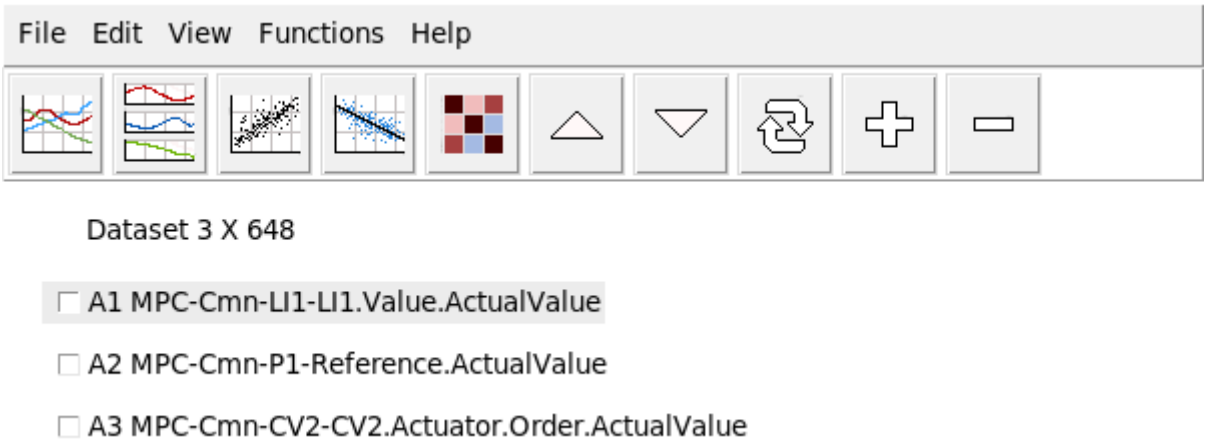

Fig 13 Logged attributes, tank level, pump reference and order to outflow control valve

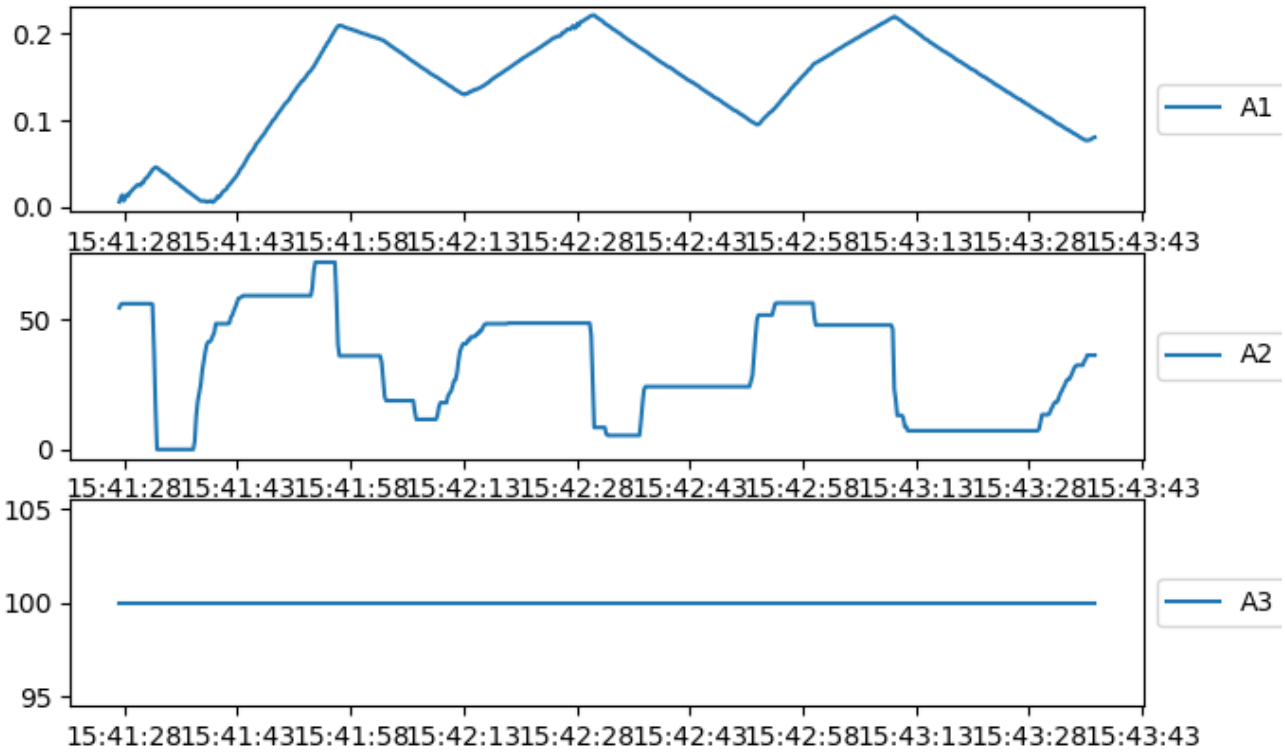

Fig 14 Curves for the logged values

Before the model is created with linear regression, the nonlinear attributes has to be linearized. The relation between reference and flow for the pump follows the curve below. From this an interpolation tables is created and written in a csv file, pump_curve.dat. From the *Convert item/Curve* function in the analyser, the curve is applied to the pump reference column.

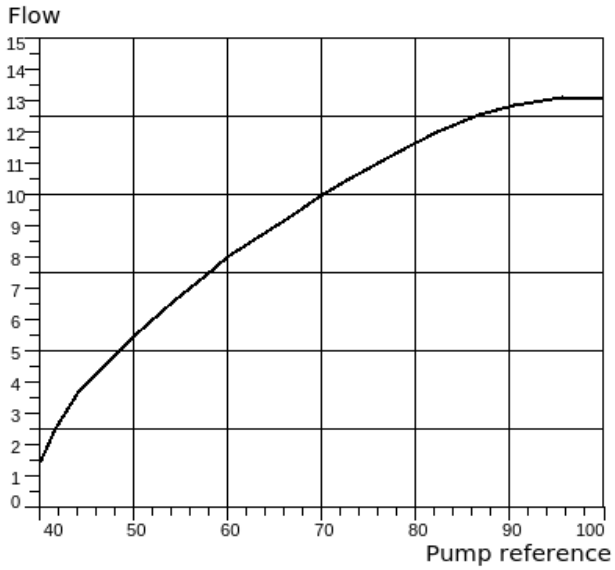

Fig 15 Speed to flow conversion curve

```
0, 0
38, 0
40, 2.57
42, 3.61
44, 4.08
46, 5.1
48, 5.96
50, 6.55
55, 7.3
60, 8.11
65, 9.1
70, 10.1
75, 11.1
80, 12.12
85, 12.9
90, 13.1
100, 13.1
```

Fig 16 Speed to pump conversion table, pump_curve.dat

Next step is to integrate both the input and output flow, thus column A2 and A3 are integrated with the *Convert item/Integral* function. The linearization is now complete and the dataset is displayed in the figure below.

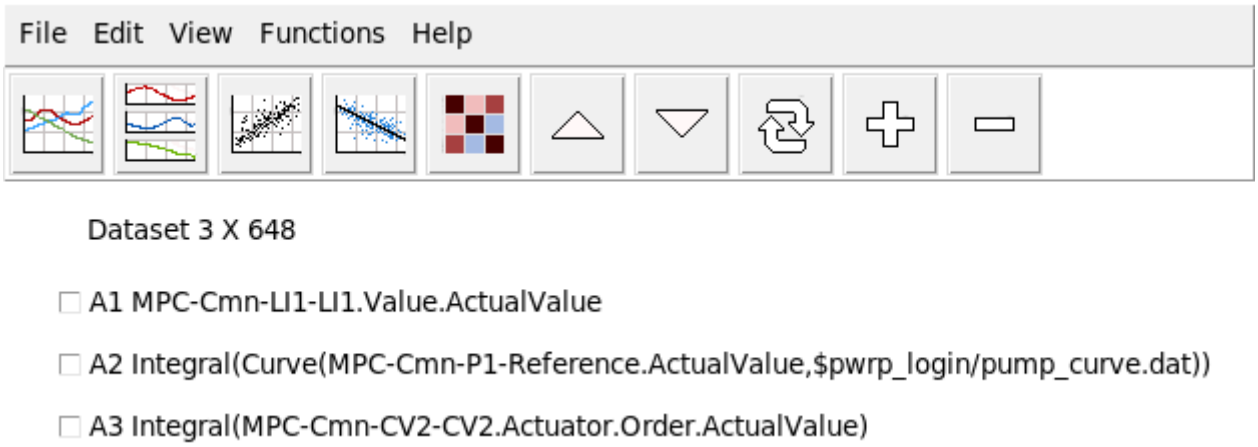

Fig 17 Linearized dataset

The model is now created with the *Linear regression* function and the result is displayed below. The score 0.986 is acceptable, and the coefficients a_0 is 0.004324, a_1 is 0.002375 and a_2 is -0.00006. These values will be inserted into the CompMPC object.

Score	0.9865
Intercept	0.004323
Integral(Curve(MPC-Cmn-P1-Reference.ActualValue,$pwrp_login/pump_curve.dat))	0.002375
Integral(MPC-Cmn-CV2-CV2.Actuator.Order.ActualValue)	-0.000060

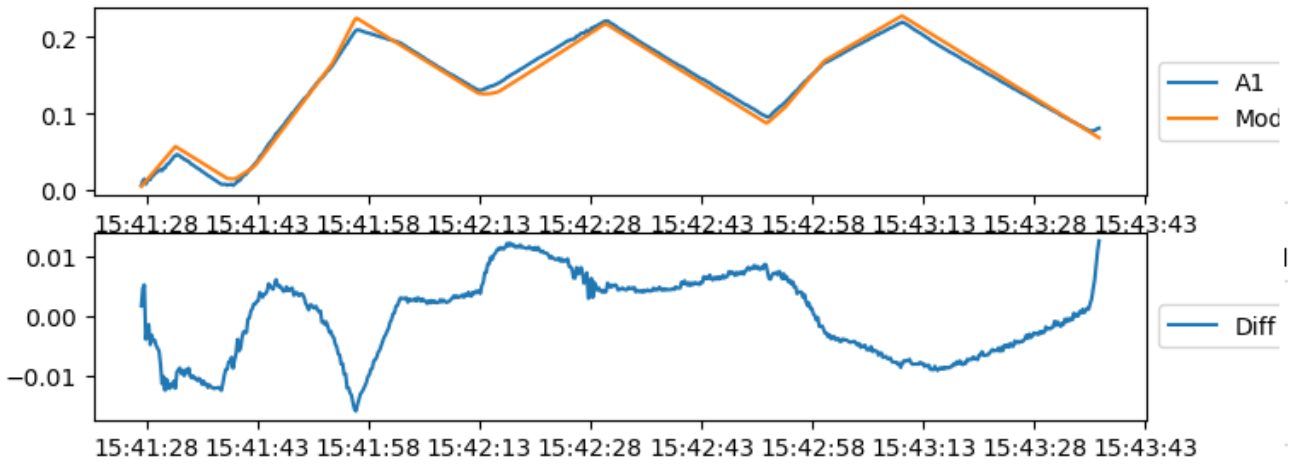

Fig 18 The linear regression model

When using the model in the MPC controller, the input data for the controller have to be linearized in the same way as when the model is created. The controller is configured in the ProviewR database with a CompMPC object in the plant hierarchy, and a CompMPC_Fo in the plc program.

The coefficients for the linear regression are inserted into the CompMPC object, a_0 into the attribute *Coeff0*, a_1 into *AttrCoeff[0]* and a_2 into *AttrCoeff[1]*.

The pump reference to flow conversion is configured in *TightAttrRelation[0]* which is set to *Curve*. This means a conversion will be made with interpolation in from a Table object connected to the C1 input of the function object. Thus the points of the conversion table in inserted in to the *Pump2* table object that is connected to the C1 input.

The integration of the in and out flow are configured in the *BaseAttrRelation[0]* and *BaseAttrRelation[1]* which are both set to *Integral*.

Setting *NoOfAttr* to 2 completes the model configuration of the controller.

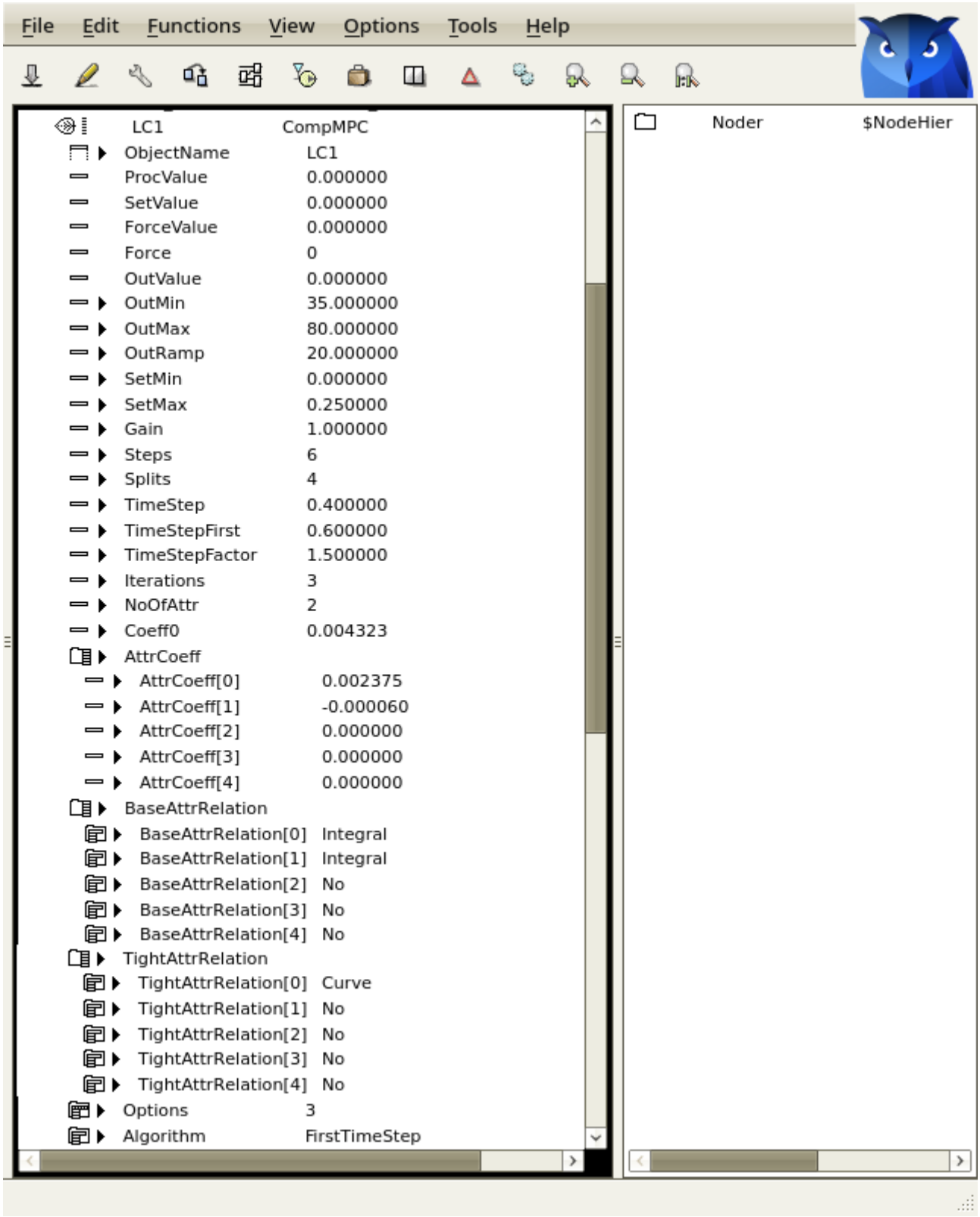

Fig 19 Configuration of CompMPC object

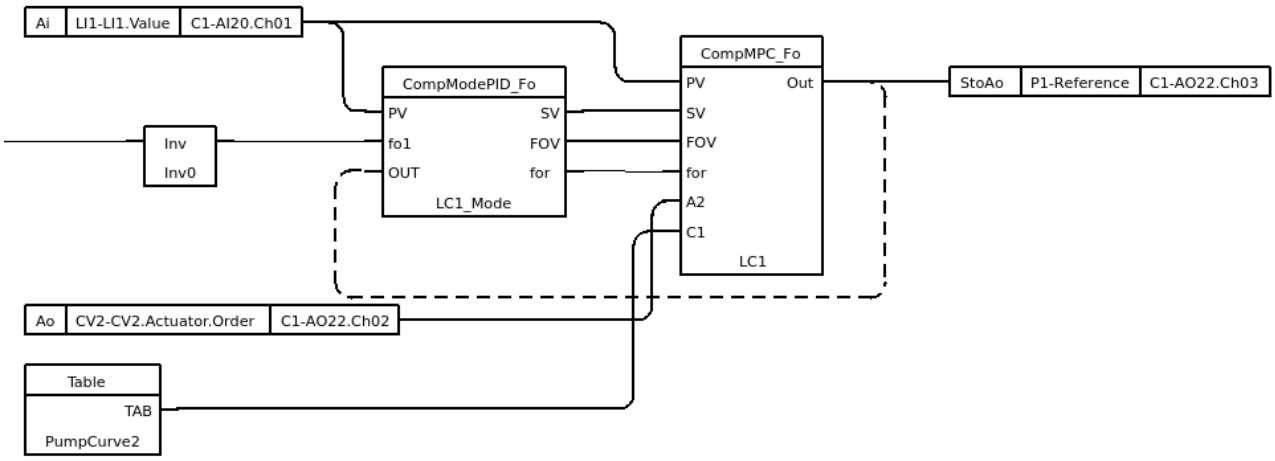

Fig 20 Plc program for CompMPC object

A step response from the controller is shown in Fig 8.

Linear regression attributes

Attribute	**Description**
NoOfAttr	Number of attributes used by the model. The attributes are normally connected to the Attr1, Attr2, … input pins of the function object.
Coeff0	Intercept coefficient in the linear regression model.
AttrCoeff	Coefficients for the attributes in the linear regression model.
BaseAttrRelation	Base type of relation for the attribute in the linear regression model.
TightAttrRelation	Tight relation for an attribute.

MLP model

MLP (multilayer perceptron) is a neural network with an input layer, a number of hidden layers and an out layer.

The MLP is implemented in the classes *CompMPC_MLP* and *CompMPC_MLP_Fo*.

The MLP can handle nonlinear relations, thus no conversion of the input parameters are required. One problem though is that the MLP can't handle accumulative processes as there is no time relation involved in the MLP training. A solution to this is implemented where inputs that have integral relation are trained also with time shifted values. The time shifts should equal the time steps in the prediction.

Flow control example

In this example we will study a flow control, where the flow is measured with the flow sensor FI1, and controlled by the speed of the pump P1. As we have seen in the Level control example above, the pump characteristics are nonlinear, but the MLP will take care of this and no linearization is needed.

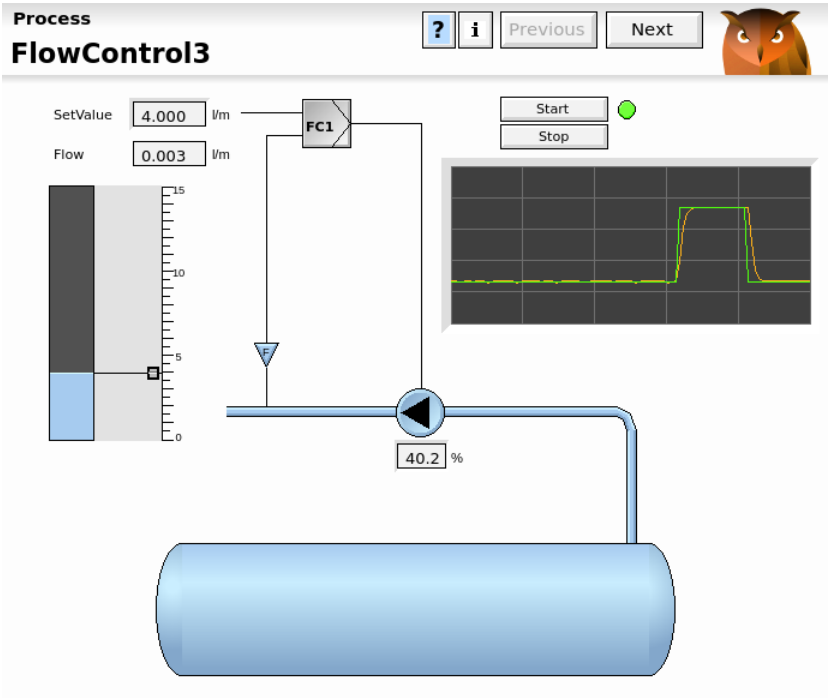

Fig 21 Flow control

In this example we will only regard the flow and the pump speed. We start by making an Xtt log with the signal from the flow sensor and the pump reference.

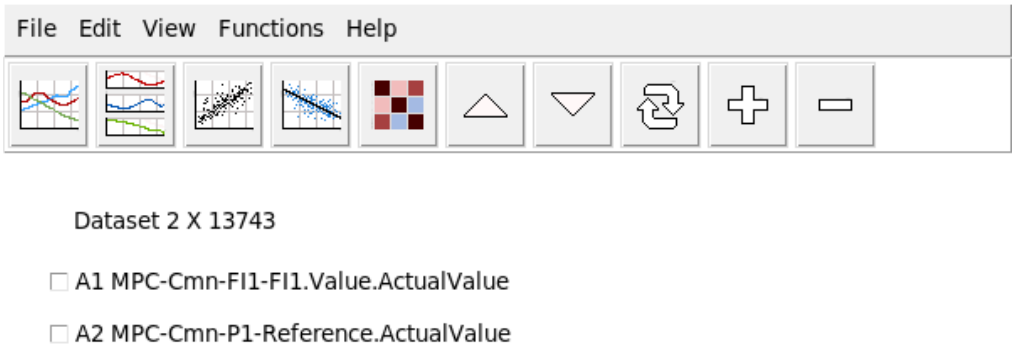

Fig 22 Dataset with flow and pump reference

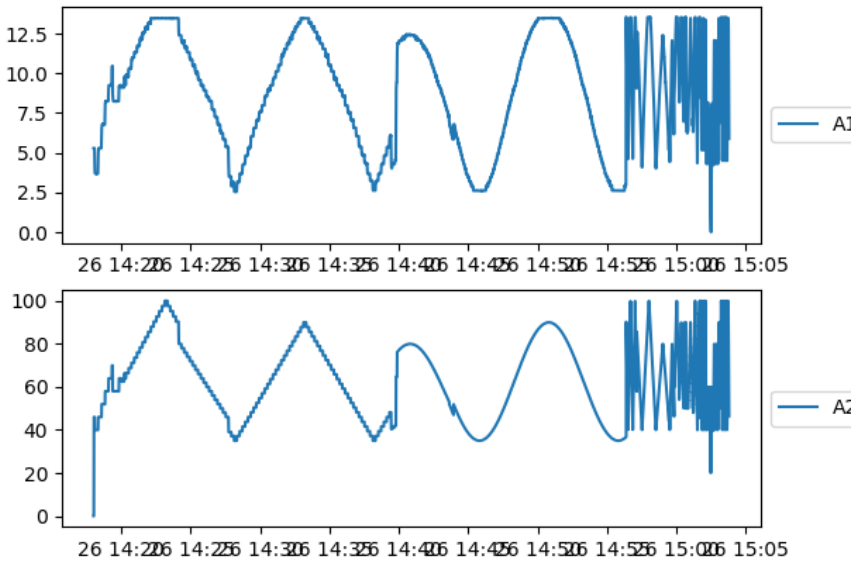

Fig 23 Curves for dataset

The MLP regressor is opened and scaler is set to *Standard*, solver to *adam* and Activation to *relu*. The default layer sizes are kept with three hidden layer and 20 nodes in each layer.

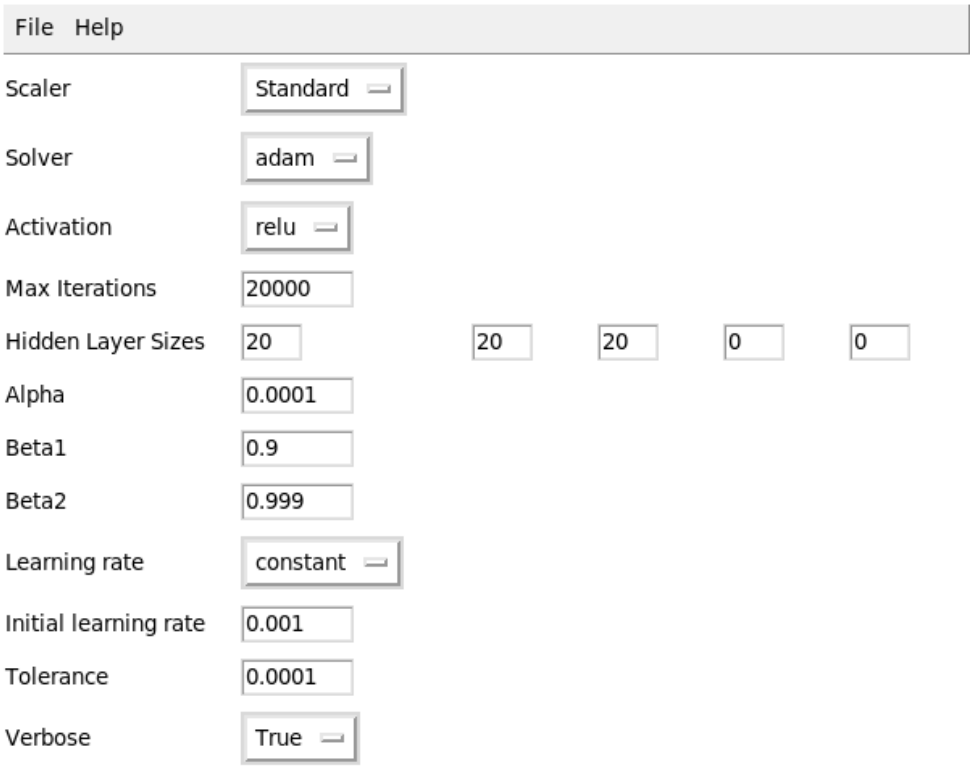

Fig 24 MLP model creation

By activating *File/Create model* in the menu the learning starts and after a while the result is displayed.

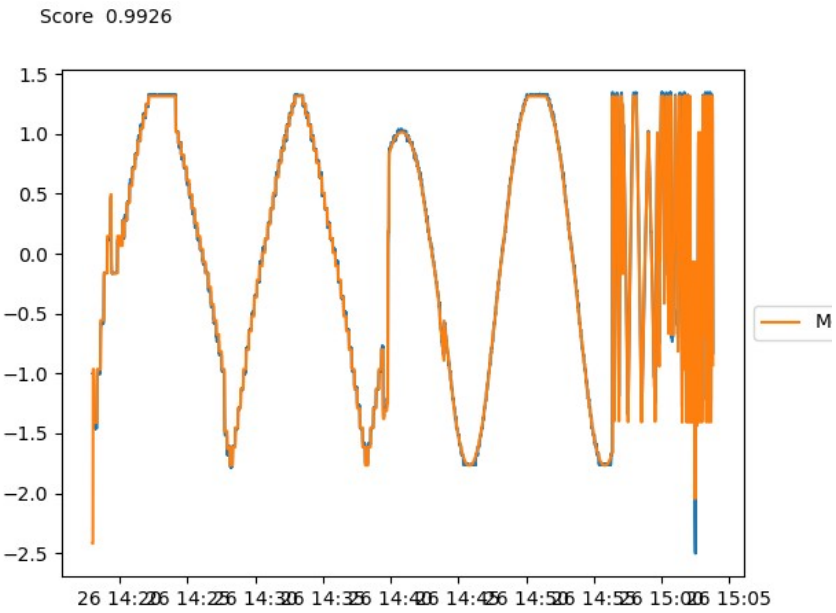

Fig 25 Model result

The model is saved and exported from *File/Save model* and *File/Export model.* The saved model will be applied to some test curves and the exported model will exported to the process node and read by the CompMPC_MLP controller.

We also made another Xtt logging of the same parameters that now will be used as a test set. The dataset is read into the analyzer, and by opening the MLP regressor and activating *File/Apply model* in the menu, the previously created model can be applied on our test data. The result is displayed

below, and as you can see the result for low flows are quite poor. The reason is that there are hardly any samples in this area in the training set, and this shows the importance of covering the whole parameter range in the training set. Unlike the linear regressor, the MLP is quite bad at extrapolating and interpolating. Adding some samples for low flows to the training set finally give us an acceptable model.

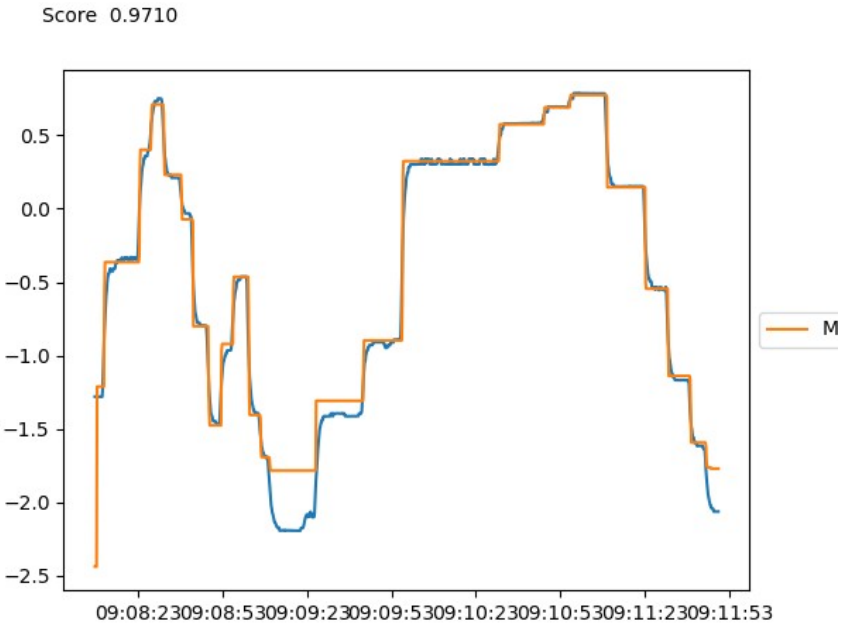

Fig 26 The model applied to the test dataset

The controller is configured with an *CompMPC_MLP* in the plant hierarchy, and the function object *CompMPL_MLP_Fo* in the plc code.

The exported model file, *$pwrp_load/flow_model2.dat*, is inserted into the attribute *ModelFile* of the CompMPC_MLP object. The file also has to be distributed to the process node. This is actually the only model specific configurations are required. The the predictive part is configured and the model correction.

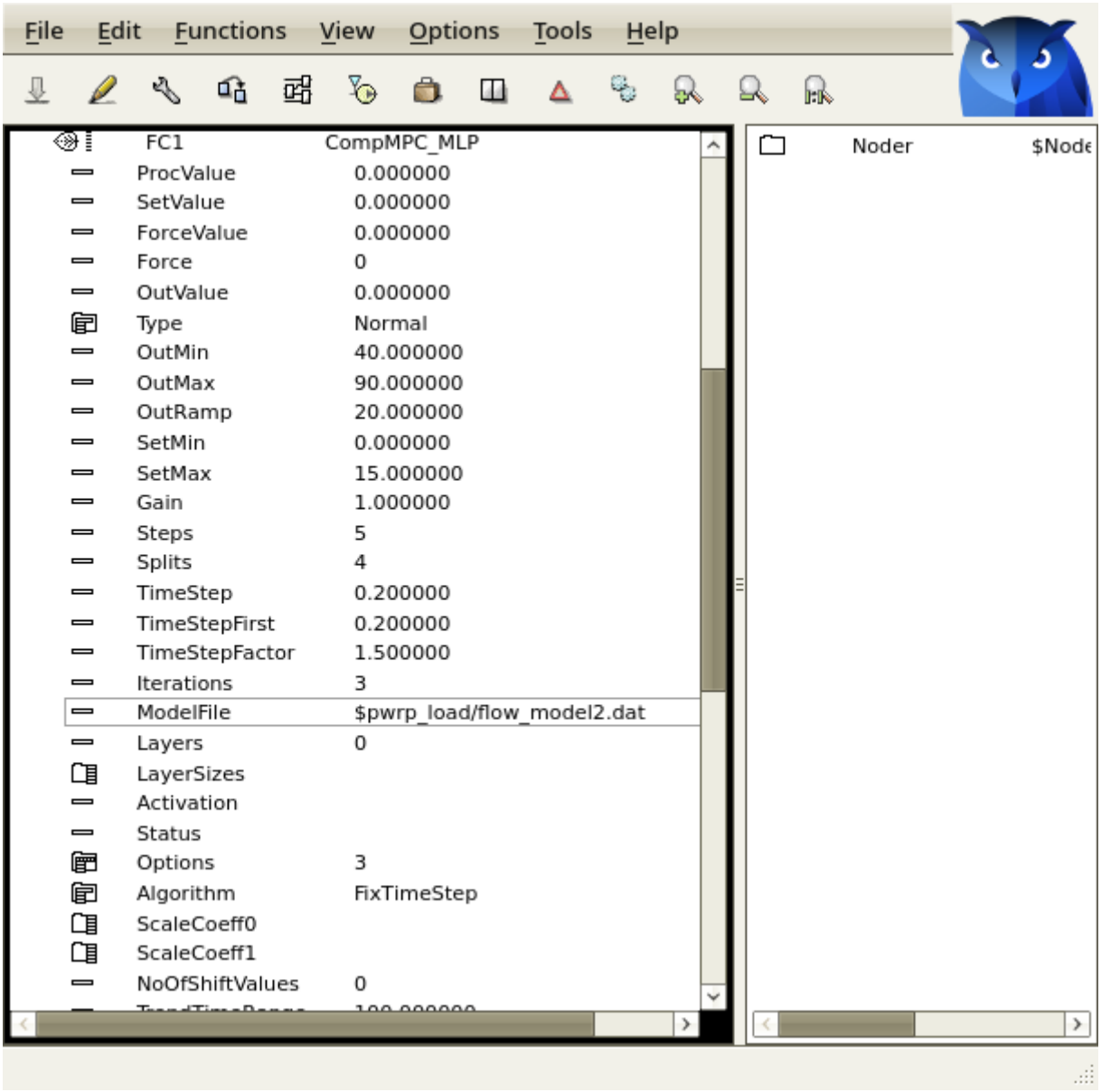

Fig 27 Plant configuration for CompMPC_MLP

The configuration is completed with a mode object of class CompModePID, and the plc programming.

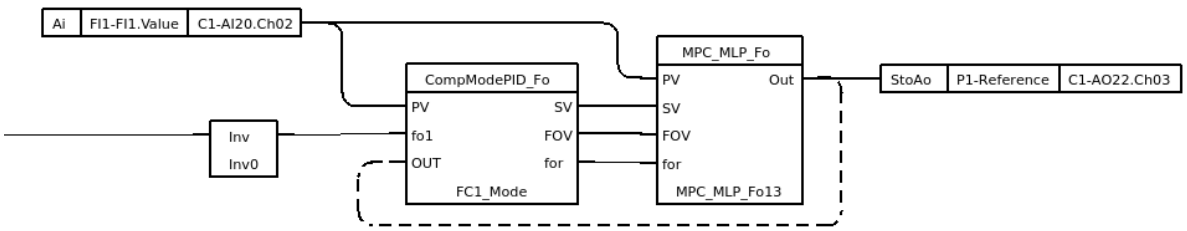

Fig 28 Plc program with CompMPC_MLP_Fo

After some tuning of the outvalue ramp and integration time, the resulting step response is displayed below.

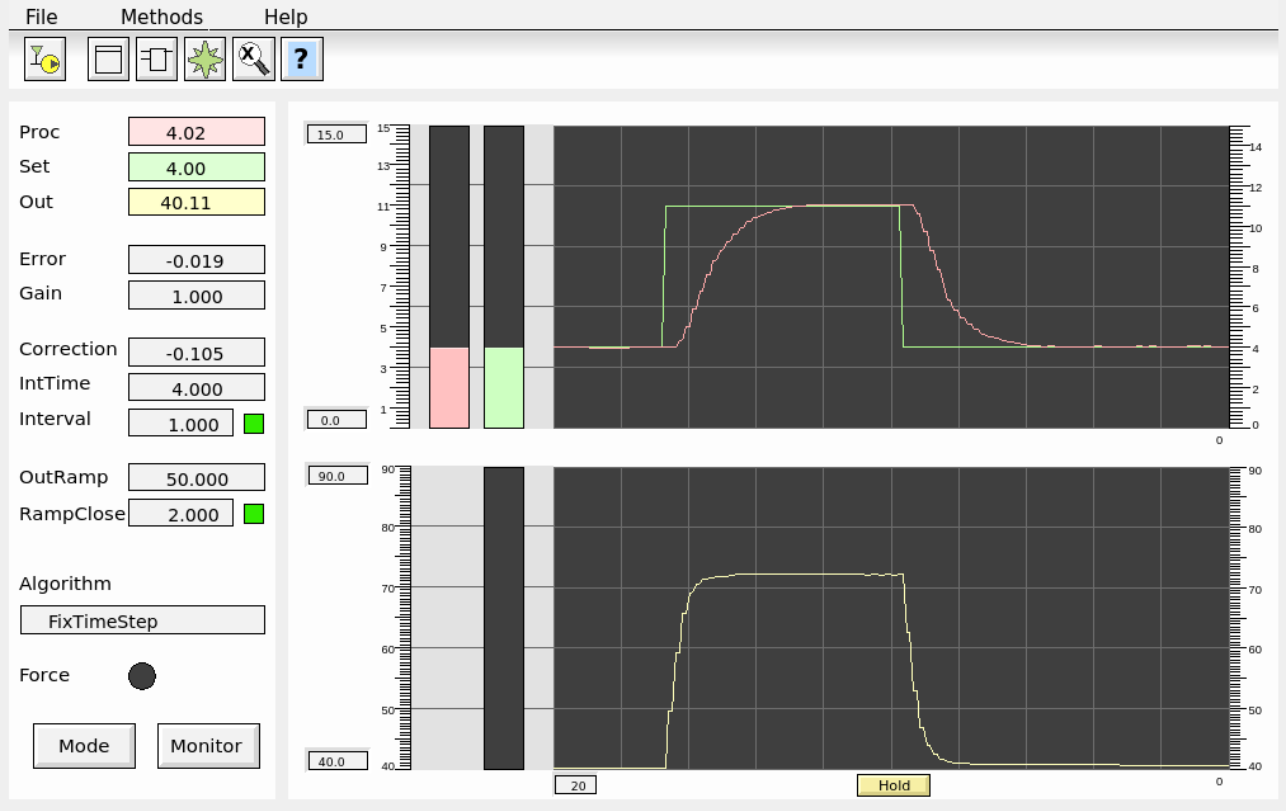

Fig 29 Step response

Monitor object

A monitor object, CompMPC_Monitor, that displays curves for the prediction and the model can be connected to the CompMPC and CompMPC_MLP objects.

In the upper graph, the set value (green) and the optimal predicted process value (orange) are displayed. In the lower graph the optimal out value (yellow) is displayed.

The current model value compared with the process value are displayed with two bars, MV (model value) and PV (process value)

The object also contains a trend curve for the model and the process value. The trend is opened from the *Model Trend* button.

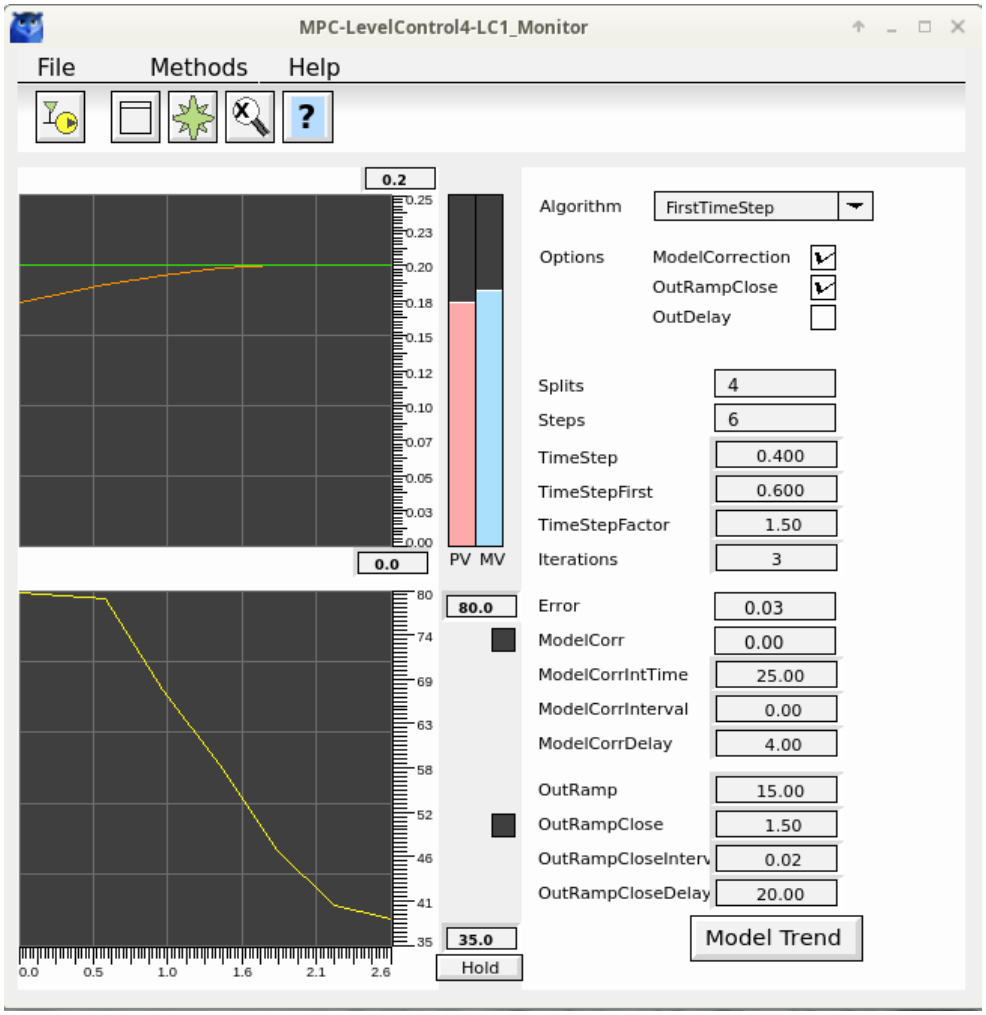

Fig 30 Object graph for monitor object

MLP model attributes

Attribute	Description
Type	Normal or accumulating.
ModelFile	Model file containing the parameter for the MLP. This file is created by the export function of the MLP Regressor.
ScaleCoeff0	Coefficient for scaling of attribute values. If scaling is selected when the model is created, these values are fetched from the model file. If other scaling should be performed, the scaling coefficients should be inserted manually.
ScaleCoeff1	Coefficient for scaling of attribute values.
NoOfShiftValues	Used for type accumulating. Number of shifted columns.