ISSN 0280-5316
ISRN LUTFD2/TFRT--5674--SE

An Adaptive PPI Controller

Per Johansson

Department of Automatic Control
Lund Institute of Technology
September 2001

Department of Automatic Control
Lund Institute of Technology
Box 118
SE-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
September 2001

Document Number
ISRN LUTFD2/TFRT—5674--SE

Author(s)
Per Johansson

Supervisor
Tore Hägglund, LTH
Lars Pernebo, ABB

Sponsoring organization

Title and subtitle
An Adaptive PPI Controller (En adaptiv PPI-regulator).

Abstract
Processes with long dead times are among the most difficult to control. This has urged the development of controllers that cope with this kind of delays. A possible approach is to use a **model predictive controll**er, i.e. a controller with an internal model of the process and its dead time. In ABB's new control system, **Control** IT, there exists such a controller, namely the **PPI controll**er. PPI stands for Predictive PI, and as the name indicates it shares some of its features with the common PI controller. The system contains an auto-tuner that is able to detect long dead times and to design a PPI controller. Another feature of Control IT is **adaptive contr**ol. An adaptive controller has the ability to change its parameters in order to optimize its performance. Today, there is no adaptation mechanism for the PPI controller.
This thesis investigates the possibilities of an adaptive version of the PPI controller. A test version has been implemented and simulation results show that there are processes that would benefit from an adaptive PPI. The big flaw is however that adaptation is possible only at set point changes, an event that might not be that common in industry. This implies that industrial studies must be performed before the solution is included in the final product. Other problems that have been encountered are sensitivity to noise and ramped set point changes. It has also been shown that the existing auto-tuner sometimes chooses a PPI design in cases were a PID design would be more useful. The thesis further discusses the subject of switching controller structure while in adaptation mode

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
43

Recipient's notes

Security classification

An Adaptive PPI Controller

Per Johansson

Table of contents

1 Introduction

This master thesis is made at the Department of Automatic Control at Lund Institute of Technology (LTH) and ABB Automation Products in Malmö.

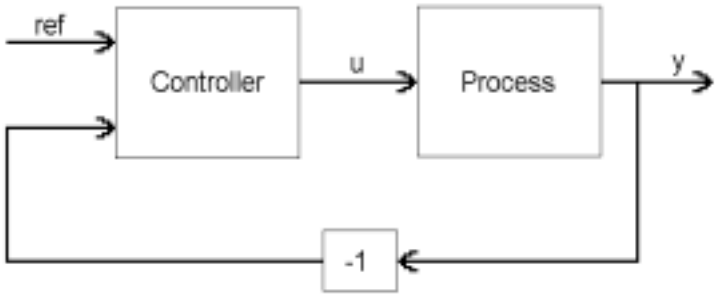

Figure 1.1 The standard feedback control loop.

When controlling something it is always good to be able to observe the thing that is controlled. In most cases it is in fact a need, try for instance to drive your car blind folded. In control theory, this observation is called feedback. Figure 1.1 shows a standard feedback control loop. If the dynamics of the process could be perfectly modelled, and one knew for sure that these dynamics were constant, there would not be necessary to use the feedback. This is of course not the case in reality, and therefore feedback control is used in order to make the control less sensitive to changing dynamics and poor models. The insensitivity to such changes is called the robustness of the control loop. Even better performance can be achieved with an *adaptive controller*, i.e. a controller that adapts to new dynamics. The parameters of an adaptive controller are updated by an adaptation mechanism in order to optimize the performance. Adaptive control is an active research area in control theory.

Many industrial processes have a delay between the control signal u and the process value y, i.e. the process does not react instantly on the control signal, see Figure 1.2[1]. This delay is called the dead time of the process. A long dead time makes the control loop less robust, and processes with long dead times are often difficult to control. A faster control can be achieved if one somehow can predict what will happen in the future (compare reducing speed due to fog that makes it hard to see the road). Prediction in control is often done by tracking of the derivative of the process value. When controlling processes with long dead times, however, this is not possible since the derivative is delayed as well. The solution to this is to use a model of the delay and the control signal to predict the future. These kinds of controllers are called *model-based predictive controllers*. In this thesis, the feasibility of an adaptive model-based predictive controller is studied.

[1] Notice the difference in notation between Figure 1.1 and Figure 1.2. The reference value (ref in Fig. 1.1) is called Sp (Set point) in Figure 1.2. Further is the control signal (u) named Out and the measured process value (y) named Pv. This notation will be used in graphs from Control IT throughout the thesis.

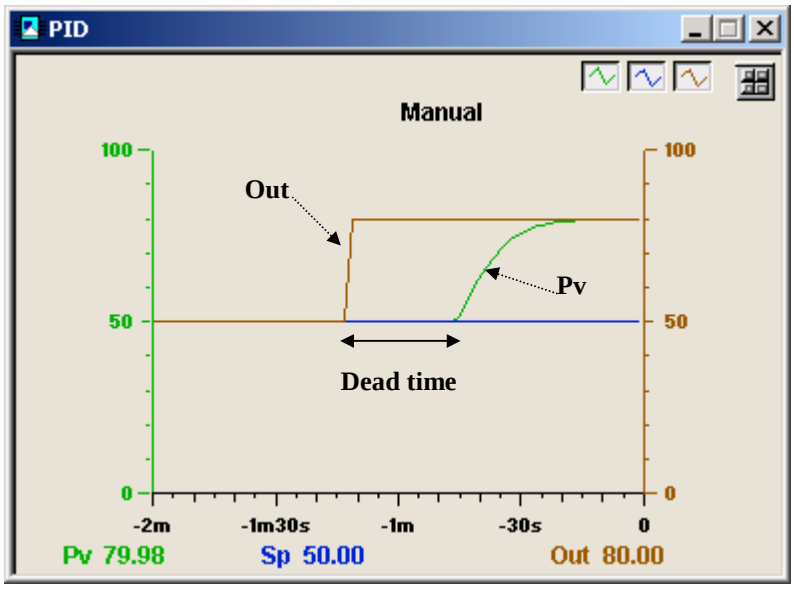

Figure 1.2 Open loop step response for a process with long dead time. The process value (Pv) does not react on the control signal (Out) until after about 25 seconds.

1.1 Problem formulation

In ABB's new control system, Control IT, there exist ready-made PID controllers. There is also a PPI controller, a controller that handles processes with long dead times. For the ordinary PID there already exists an adaptation mechanism, which adjusts the controller design if the process is changing. The PPI controller lacks such option, and the aim of this thesis is to investigate if it is possible to introduce an adaptive PPI controller.

1.2 Outline

This thesis is organized as follows. In chapter 2, control of processes with long dead times is discussed. Here is the PPI controller presented. Chapter 3 treats the main idea of how an adaptive version of the PPI can be implemented. The thesis continues with chapter 4, in which implementation aspects are considered. This chapter also includes a presentation of the Control Builder Professional, the programming tool in Control IT that has been used for implementation. Chapter 5 deals with the integration with the current system, whereas chapter 6 is dedicated to the simulations that have been carried out in order to verify the correctness of the algorithm. In chapter 7, finally, conclusions are drawn and future work suggested.

2 Control of processes with long dead times

Processes with long dead times are, as mentioned, among the most difficult to control. An increase of the dead time reduces the phase margin and might lead to unstable behaviour. This chapter presents different ways to control these processes.

2.1 PID control

Although control theory has come up with a variety of advanced controllers, the by far most common controller in industry still is the PID controller. The reason for the popularity of PID controllers is that this control structure is easily tuned, one does not need any advanced knowledge in control theory in order to tune the loop. The PID controller consists of three parts: the proportional or P-part that acts proportionally to the control error, the integrating or I-part, that eliminates static control errors and, finally, the derivative or D-part that acts as a predictor. In control of processes with long dead times the D-part does, as stated in the introduction, not improve the control. On the contrary, it rather gives a decrease in performance and therefore it is often omitted in these cases.

When the derivative part is switched off, all prediction is switched off. This is a drawback since in order to have a stable control it is important that the PI control design is rather slow. One sense that better performance can be achieved if prediction is used. This is discussed in the next section, which presents a nice solution. There are, however, some good things with using the PI controller when controlling processes with long dead times. The slow control makes the design quite robust. Furthermore, the PI controller is easy to tune since it only has two parameters, which are very well known among control engineers.

2.2 The Smith predictor

One way to achieve better control is to use a model-based predictive controller. The idea is to use a model, including the dead time, to predict what will happen in the future. The most common controller of this type is the Smith predictor. The structure of this controller is shown in Figure 2.1. The control signal is passed on not only to the process, but also to two models. These models are often first order approximations of the process. The first model, G(s), also includes an approximated dead time L.

$$G(s) = \frac{K_p}{1+sT} e^{-sL} \tag{2.1}$$

The second model, $G^*(s)$, is then the same system without dead time.

$$G^*(s) = \frac{K_p}{1+sT} \tag{2.2}$$

By subtracting the model output from the real process value, the controller is tricked to act as if there were no dead time. The models are of course not often that exact, but the first order model can capture most of the process dynamics. The main reason that one uses a first order system as model is that it is quite easy to find the parameters that approximate such a system, see further section 3.1. The controller is usually a PI controller.

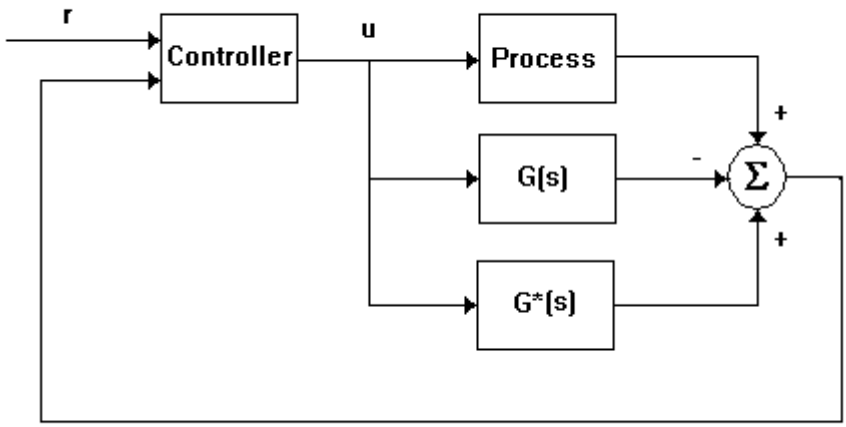

Figure 2.1 The Smith predictor. G(s) is a first order model of the process. $G^*(s)$ is the same model, but without dead time. The controller is often of PI structure.

2.3 The PPI controller

In section 2.1 it was mentioned that the PI controller is easy to tune. There are some simple rules that make it possible to tune the controller without having any knowledge of control theory. One example of such a rule is that higher gain K will give a faster but less damped control. The Smith predictor from section 2.2 does not share this easy-tuning feature. If the models are of order one, there are five parameters to be determined: K and T_i in the controller and K_p, T and L in the models. This section presents the PPI controller, which combines the prediction of the Smith predictor and the easy tuning of an ordinary PI controller.

PPI stands for Predictive PI and the idea is presented in [Hägglund]. The structure is the same as in the Smith predictor with a PI controller. The novelty is the way the parameters are chosen. The controller gain, K, is set to $1/K_p$ and the integral time, T_i, to T. This leaves only three parameters, T, K_p and L to be determined. If one has an estimate of the dead time L, the other two parameters can be tuned manually as in an ordinary PI controller. The structure of the PI controller is:

$$u(t) = K(e(t) + \frac{1}{T_i}\int e(t)dt) \qquad (2.3)$$

Where e(t) is the control error signal,

$$e(t) = ref(t) - y(t) \tag{2.4}$$

Using this together with the models in eq.(2.1) and eq.(2.2) and the choice of K and T_i as above gives the following structure of the PPI controller:

$$u(t) = K(1 + \frac{1}{pT_i})e(t) - \frac{1}{pT_i}(u(t) - u(t - L)) \tag{2.5}$$

where *p* is the differential operator d/dt. Some performance is of course lost when this simple model is used, but the similarities to the PID controller and few tuning parameters are in many cases more useful than a slightly better model. The PPI controller is further not suited for all kind of processes. The auto-tuner in Control IT suggests a PPI design if the estimated dead time is about twice as big as the estimated time constant. A comparison between an ordinary PI and a PPI controller for a 'PPI suitable' process (G_4, see section 6.1) is made in Figure 2.2. Two controllers are auto-tuned. One of them uses the automatically chosen PPI controller with gain of 1.0, integral time T_i of 4.5 seconds and a dead time estimate of 10.5 seconds. The other one is forced to use the suggested PI controller. This controller has a gain equal to 0.2 and a T_i of 6.1 seconds. A set point step is made, and after that, a load disturbance is introduced. One can see that the PPI controller is faster both in the step response and to reduce the disturbance. The PPI controller has been implemented and tested in industrial applications with good result, see [Hägglund].

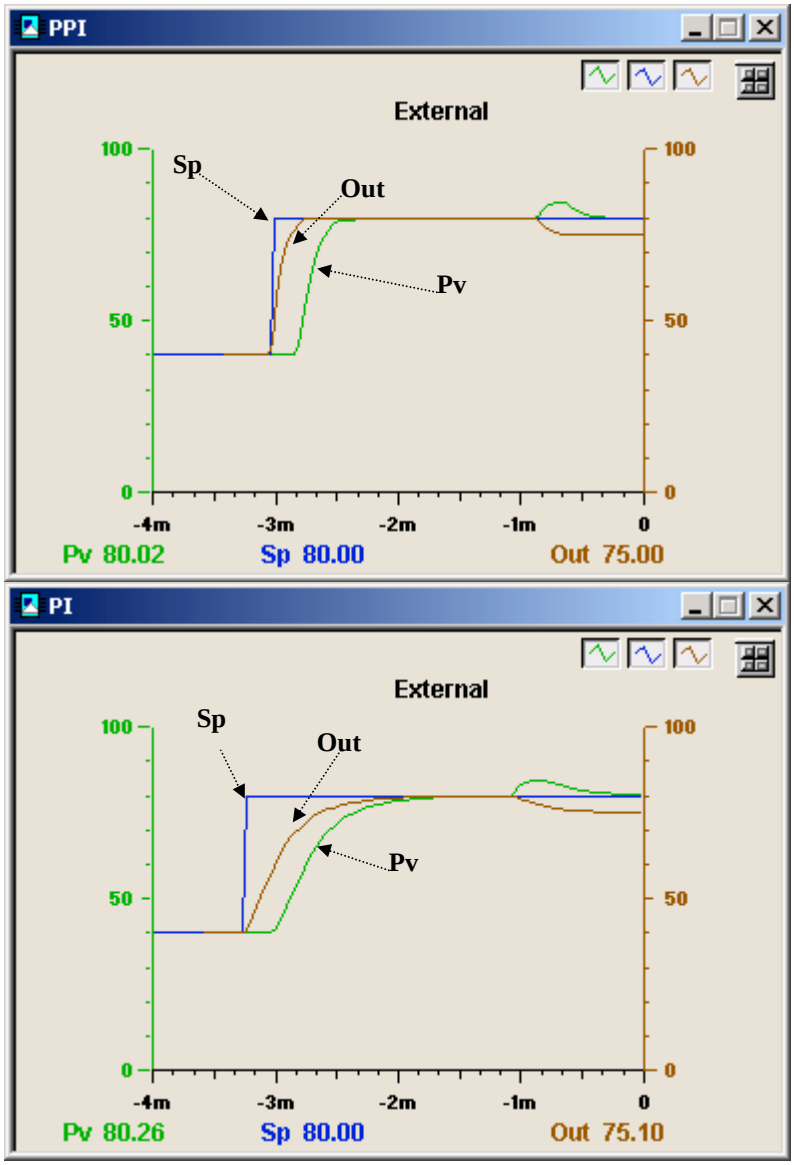

Figure 2.2 A comparison between a PPI (upper) and a PI controller during a set point step followed by a load disturbance. Notice that the controller output almost immediately reaches its new value in the PPI controller, whereas the output from the PI controller is more slowly raised.

3 How to make the PPI adaptive

The idea of adaptation is that the controller should be able to handle changes in the process dynamics. The adaptation mechanism detects changes in the process and then updates the controller design. A block diagram of the adaptive controller that is presented in this work can be seen in Figure 3.1. The updating algorithm tracks the set point, the controller output and the process value in order to detect changes in the process. There are many ways to do this, but one common need is that there must be an excitation of the system so that the dynamics become measurable. Note that a robust design can handle changes in the process, but with an adaptive controller the process can change more and a better performance can be attained. In Control IT today, there exists an adaptive PID-controller but there is no such function for the PPI controller and this is the motivation for this thesis.

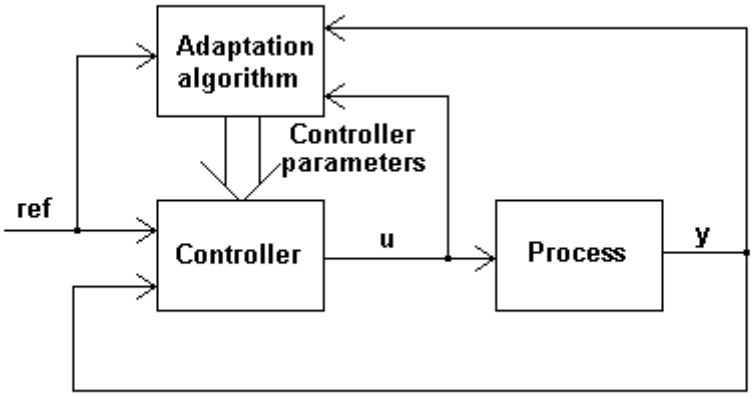

Figure 3.1 Block diagram for adaptation.

There are, however, also some problems introduced together with the adaptation. The idea is to make human supervision and retuning somehow unnecessary, but how can one guarantee the stability of the system? Adaptive controllers can also show strange behaviours like turning themselves off.

3.1 Estimating parameters in closed loop

As mentioned in section 2.3, one needs three parameters in order to tune the PPI. These are the process dead time, time constant and static gain. To make the PPI controller adaptive therefore requires that one can estimate these parameters in closed loop, and this is the topic of this section.

3.1.1 The method of moments

Given a step response, it does not seem that difficult to estimate the needed parameters. In an ideal world with no disturbances, there are several ways to estimate e.g. the dead time. But when noise and other disturbances are introduced, all methods depending on evaluation in a single point will give noise sensitive estimations. One solution to this is to, if possible, use integral calculations. If the noise has zero mean, it will not effect the estimation that much anymore. This section describes one such

method, the method of moments. The presentation of the method is brief though, since we are only interested in the result. For a more detailed description, see [Åström and Hägglund] and [Norberg]. The briefness might make this section quite hard to understand, but full understanding is not necessary. The introduction of the parameter T_{ar} is essential however, since the parameter will be used a lot throughout the thesis.

Consider the frequency domain description of the control loop

$$Y(s) = G(s) \cdot U(s) \tag{3.1}$$

With the three-parameter process model

$$G(s) = \frac{K_p}{1+sT} e^{-sL} \tag{3.2}$$

First, a new parameter is introduced

$$T_{ar} = T + L \tag{3.3}$$

This parameter is called the average residence time. The idea is now to estimate T_{ar} and either of T or L. Then, it is of course easy to calculate the last parameter from eq. (3.3). Let us start with the calculation of T_{ar}. Taking the logarithm of eq. (3.2) gives

$$\log G(s) = \log K_p - sL - \log (1+sT) \tag{3.4}$$

This expression is then differentiated

$$\frac{G'(s)}{G(s)} = -L - \frac{T}{1+sT} \tag{3.5}$$

and T_{ar} can now be derived from eq. (3.5) with s=0

$$T_{ar} = -\frac{G'(0)}{G(0)} \tag{3.6}$$

Differentiation of eq. (3.1) gives

$$Y'(s) = G'(s)U(s) + G(s)U'(s) \tag{3.7}$$

Furthermore, from the definition of the Laplace transform, the following holds

$$
\begin{aligned}
Y(0) &= \int_0^\infty y(t)dt \\
Y'(0) &= -\int_0^\infty t \cdot y(t)dt \\
U(0) &= \int_0^\infty u(t)dt \\
U'(0) &= -\int_0^\infty t \cdot u(t)dt
\end{aligned}
\tag{3.8}
$$

The restriction to eq. (3.8) is that the integrals must converge. Given a step response, how can one guarantee this? The solution to this (see [Norberg]) is to use the derivatives of the signals. If the process is in rest when the step is taken, the derivatives of y and u are zero. The same holds when the process has settled again after the step. Using the derivatives of the signals in Eq. (3.8) gives:

$$
\begin{aligned}
Y_d(0) &= \int_0^\infty y'(t)dt = \int_0^{t_f} y'(t)dt = y(t_f) - y(0) \\
Y_d'(0) &= -\int_0^\infty t \cdot y'(t)dt = -\int_0^{t_f} (y(t_f) - y(t))dt = -A_2 \\
U_d(0) &= \int_0^\infty u'(t)dt = \int_0^{t_f} u'(t)dt = u(t_f) - u(0) \\
U_d'(0) &= -\int_0^\infty t \cdot u'(t)dt = -\int_0^{t_f} (u(t_f) - u(t)dt) = -A_1
\end{aligned}
\tag{3.9}
$$

Where t_f is the time when steady state is reached again. The two remaining integrals can be interpreted as two areas, see Figure 3.2. Since the system is linear, eq. (3.1) can be rewritten as

$$Y_d(s) = G(s) \cdot U_d(s) \tag{3.10}$$

and eq. (3.7) as

$$Y_d'(s) = G'(s)U_d(s) + G(s)U_d'(s) \tag{3.11}$$

Now, by combining eq. (3.9) with eq. (3.6) and eq. (3.11), T_{ar} can be calculated as

$$T_{ar} = \frac{A_2(u(t_f) - u(0)) - A_1(y(t_f) - y(0))}{(y(t_f) - y(0)) \cdot (u(t_f) - u(0))} \tag{3.12}$$

The fact that the integrals in eq. (3.9) can be reduced to integrands of order one is a nice feature for numerical calculations. Both T and L can be estimated in the same way, but this requires second derivatives, and there is no way to reduce the integrals in this case. This leads to bad numerical calculations, and other solutions are needed.

In the auto-tuner in Control IT, the method of moments is used to calculate T. This is done with an open loop step though (see [Norberg]), and this method is therefore not suited for adaptation. The solution that has been chosen is to estimate T_{ar} according to this section, and the dead time in a manner that is presented in the next section. The last parameter, K_p, is calculated as the ratio between the change in process value and the change in control signal.

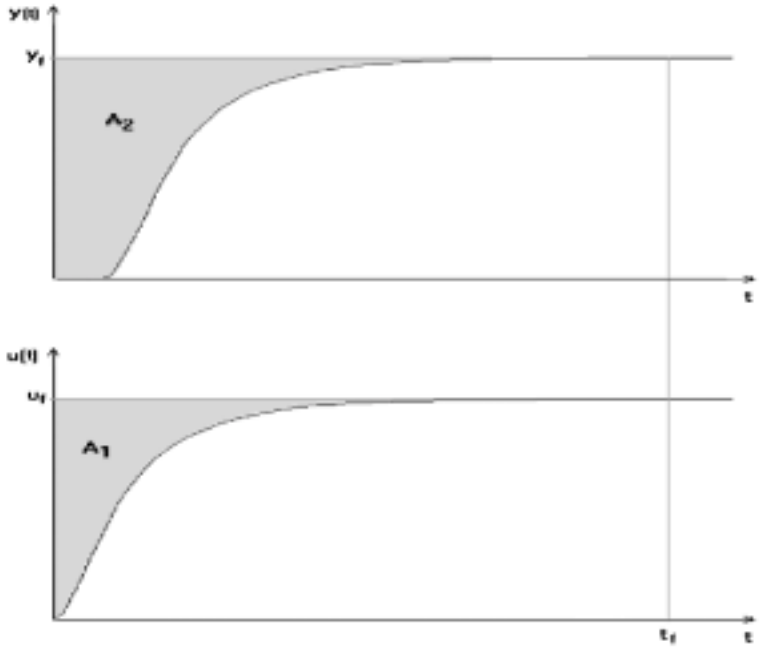

Figure 3.2 Areas for T_{ar} calculations.

3.1.2 Estimating the dead time

The problem with dead time estimating in closed loop is that in order to estimate the process dead time, the only event that excites the system enough is a change in the reference value, see [Isaksson *et al.*]. This is a disappointment because in process industry, control is often focused on suppressing disturbances to keep the process on a certain level, and set point changes are rare events. This section presents two different methods to estimate the dead time from a set point step response. Both methods have their benefits and disadvantages, and in the final solution both are used.

The first approach is to simply check the process value and set the dead time equal to the time from the step is taken until the process starts to react. This method is in the rest of the thesis called the *level method*. The problem when using this technique is to distinguish the step response from noise or other disturbances. The solution is to set the dead time equal to the last time the process value is below some level above the original value. The method can be further improved by taking the time in between the first time above the level and the last time below. The level method is, as have been shown in simulations, very noise sensitive. When ramped set point changes are used, however, this is the best way to estimate the dead time.

The second estimating technique is the one that has been proven to be the most useful. The idea is to take the maximum derivative during the step response and then

extrapolate a straight line with this slope. The dead time is taken as the time when the straight line intersects with the old set point level (see Figure 3.3). This method is called the *derivative method*. The largest problem with this approach is numerical problems when calculating the derivative. These problems will be carefully discussed in section 4.3.1.

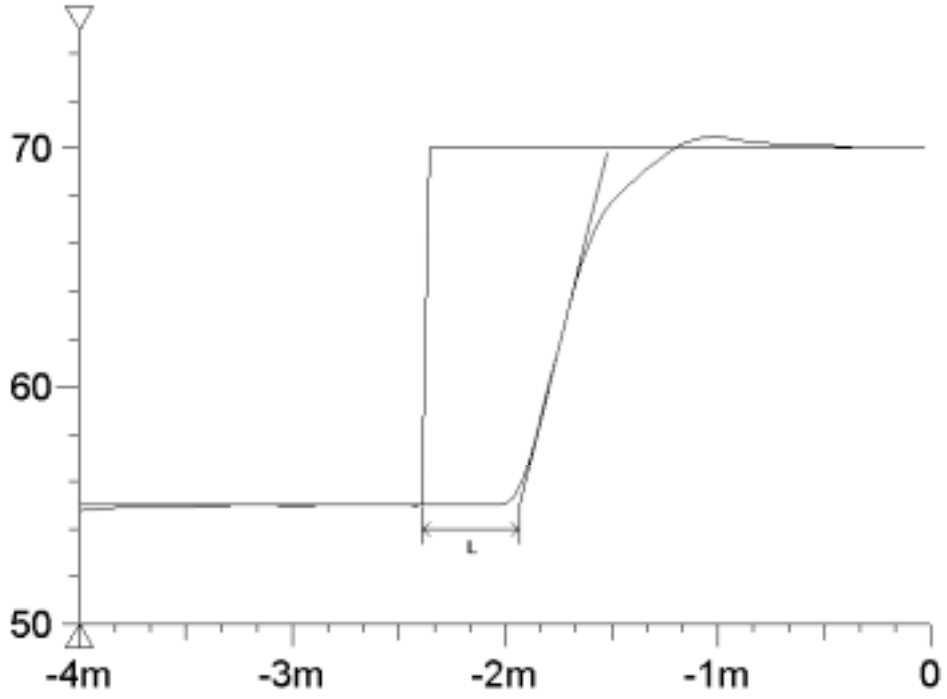

Figure 3.3 Calculation of the dead time L, using the maximum derivative of the process value during a step response.

It has been shown, that in order to get usable estimates from the derivative method, it is needed that the change in control signal is somewhat close to a step change. This will give an 'open loop-like' behaviour of the step response. Remember that it is the open loop time constant and dead time that is wanted. If the step response is slow, the dead time will be estimated too long. It is still possible to use the derivative method in the adaptation as long as the controller is of PPI structure, since the control signal in that case almost directly is set to the assumed new level. It will be shown later that the adaptation can be run also when a PID controller is used. In this case, and when ramped set point changes are used, the dead time must however be estimated with the level method.

When the method of maximum derivative is used, the dead time is estimated a little longer than the true dead time. This is not a problem though. On the contrary, simulations shows that higher performance is achieved if the estimated dead time is set a little longer than the real dead time, see [Hägglund]. This is due to the loss of dynamics when one uses a first order model to describe a system of higher order. To compensate for this, the model has a longer dead time. Figure 3.4 shows the difference between a fourth order system and a predicted first order model of that system.

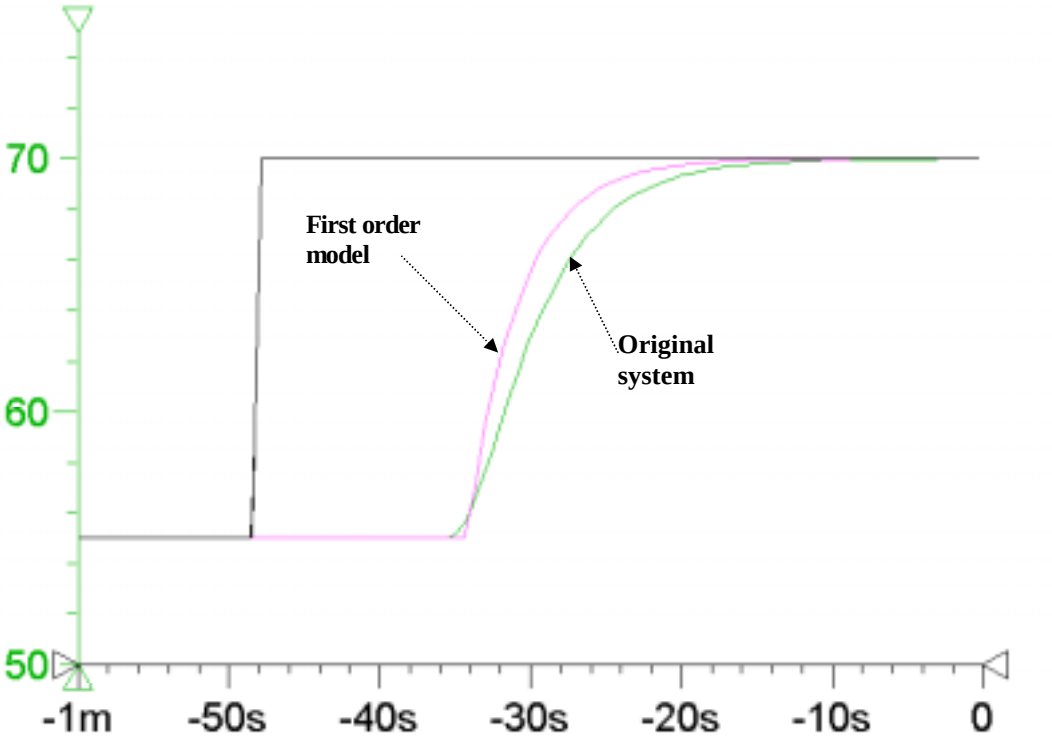

Figure 3.4 Comparison between a fourth order system and a first order model.

3.2 Supervision

One of the reasons that adaptive controllers are not that popular in industry is the fact that one really could not guarantee a safe performance. Even if most algorithms can handle a majority of cases, there are always unpredictable disturbances that can affect the performance. This is also the case for the method presented in the previous sections. One cannot be sure that the estimates are good, and therefore some kind of supervision that detects bad estimates is needed. This is of course quite tricky and there is no way to be totally secure.

The bad estimates that are the simplest to detect are the ones that are totally unrealistic, e.g. negative results. The most common such error that has been encountered is a negative estimate of the time constant T. This might depend on two different things. The dead time might have been estimated too long, in fact longer then T_{ar}, and the result is a negative T. The second possibility is that the T_{ar} estimation is too small. Given this, there are three possible ways to handle the situation. The first, and the safest, is to simply discard the result and use the old values. If the reason for the bad estimates is a change in the process, this is not optimal though. The alternative is to use either the new T_{ar} and old L or the new L and old T. Since the L estimation is more error prone, one might think that the first approach is the best. On the other hand, the reason for bad T_{ar} estimates is often changes in the process, and therefore is the second approach used. This is by no means verified as the best solution.

Above, only treatment of estimates when T is negative was handled. There is another check that could be performed in order to verify good estimates. A change in dead time will, if there is no change in other dynamics, result in an equally large change in T_{ar}. If the dead time estimate is changed considerably, one can check if T_{ar} has changed equally and if so is not the case the dead time estimate is considered bad. This test has not been implemented since there are many design parameters to decide upon, e.g. what size of dead time change is considered large? Further is it a weak assumption that only one of L and T changes between every step.

If gain scheduling (see further section 3.3.2) is used in combination with the adaptive PPI, only set point steps within a gain scheduling range are accepted as valid. This supervision is made outside the PPI adaptation function block. The adaptive PPI is notified from the gain-scheduling algorithm when a change in range is made. The PPI adaptation is reset upon such change.

3.3 Is the adaptation useful?

Before rushing into the implementation, one should maybe pose the question: "Is there really a need for an adaptive PPI controller?" In order to answer this question properly, a good thing would be to ask control engineers who work with processes with long dead times the following questions:

- Is it common with changing processes that necessitate the use of adaptation?
- How often are there changes in the set point?
- Are the PPI controller used at all?

Unfortunately, such an investigation has not been performed. In this section, the issues mentioned above are discussed though.

3.3.1 Possible usage

It is a well-known fact that with a PPI controller one gains faster control but loses robustness. This means that the use of PPI controllers is restricted to fairly constant processes. There might be processes that could benefit the faster control of the PPI if there were a good adaptation. One should remember though, that since the estimation of the dead time is possible only at set point changes, which might not occur that often, it is important that the adaptive PPI controller only is used in cases when there are small and slow changes in the process parameters. These aspects have been considered when the simulation examples have been chosen. If set point changes are very rare, maybe only at production changes, the usefulness of the adaptation is very limited.

The conclusion is that in cases where PPI controllers have been used with good result, an adaptive version might improve the performance if only set point changes are made reasonable frequent. This means that there are good reasons for implementing the adaptive algorithm for the PPI, at least in order to perform simulations.

3.3.2 Adaptation vs. gain scheduling

If the changes in the process parameters are measurable, one can tune the controller parameters for several ranges of process dynamics and then choose the proper controller when working in a certain range. This is called gain scheduling, and is often a good way to handle non-linear processes with partly linear behaviour.

Many times, the reason for the dead time is a transportation of material. If the delay is proportional to some measurable state, e.g. the speed of a conveyer or the flow of a liquid, it is possible to use gain scheduling to handle the changes in dead time. The advice is to use gain scheduling when possible. One could of course also combine the two methods in order to improve the control in each range.

4 Implementation

This chapter treats implementation aspects. First, Control Builder Professional is introduced. Then is the algorithm presented and problems discussed.

4.1 Control Builder Professional

Control Builder Professional (CBP) is an engineering tool that supports the IEC 61131-3 standard. IEC 61131-3 (or 1131) is a standard that specifies methods and languages to program/configure process control. The standard contains five programming languages that are all included in CBP. The five standard programming languages are: Structured Text (ST), Instruction List (IL), Function Block Diagram (FBD), Ladder Diagram (LD) and Sequential Function Chart (SFC). With these languages it is possible to develop programs that then can be downloaded and executed in a controller. The controller can be an independent hardware unit or a so-called soft controller, which runs on an ordinary PC.

An application in 1131 is built up by Program Organization Units (POU) that makes it easer to structure the work and re-use code. A POU can contain other POU:s, i.e. the idea is similar to object oriented programming. The POU:s that are specified in 1131 are: programs, function blocks and functions. Even though programs cannot be included in other POU:s, it is defined as a POU. Since 1131 do not fully support object oriented programming, there are an additional POU added in CBP, namely the control module type. The control module type also contains a graphical editor, which makes it easy to design GUI:s for soft controllers. All implementation in this thesis has been carried out in CBP using ST. The adaptation algorithm is implemented as a single function block that then has been added to the already existing PID control module.

4.2 The algorithm

This section contains a brief description of the PPI adaptation algorithms that have been implemented. The structure can be viewed in Figure 4.1, which presents the solution as a state chart.

The algorithm starts in ‘wait for steady’ when adaptation is turned on. When the process is steady near the set point, the steady mode is entered. While in this mode, the algorithm estimates noise and waits for a set point step to occur. If the process is disturbed from its steady state, the algorithm re-enters the wait for steady mode. If a set point change is made while in steady state, the estimation of the three parameters is started. The algorithm is now in the ramp mode and stays there as long as the set point is changed every sample. The ramp is aborted if a set point change is made in opposite direction than the first change, and in that case, the algorithm is put in wait for steady mode again. When the ramped set point change is over, the step state is started and steady state is awaited. The step is aborted if the step amplitude is found to be too small in comparison with the noise level or if there is a new set point change before steady state is reached. When the step is ready and steady state is achieved, new parameters are calculated and if they are good the controller is redesigned. The algorithm is once again in steady state mode and a new step can be performed.

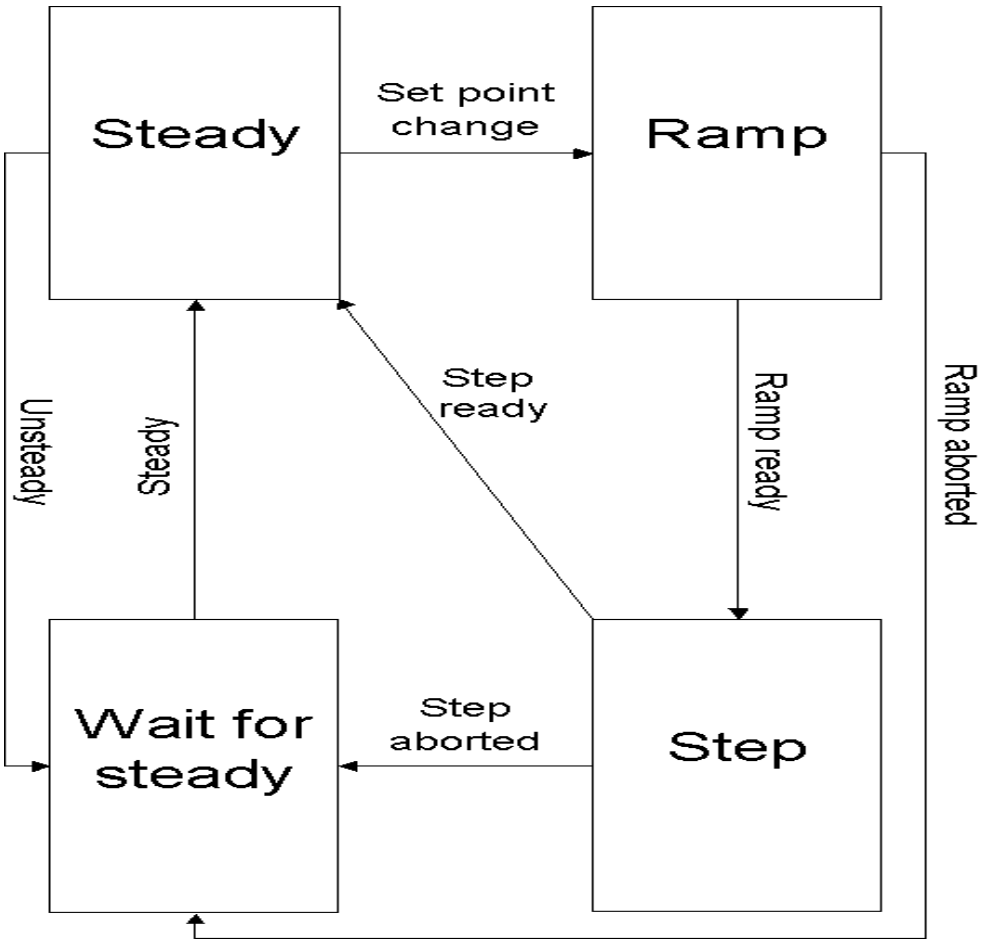

Figure 4.1 State chart for the PPI adaptation algorithm

4.3 Problems

Not surprisingly did some problems occur during implementation and simulations. This section discusses these and gives the solutions that are used.

4.3.1 Derivatives and noise

As mentioned before, the derivative sensitivity to noise is a most severe problem. Filtering with a low pass filter can reduce this sensitivity, but if the filtering is too 'hard', the signal will be so distorted that there is no use of the derivative. There is thus a trade off between good derivatives and noise insensitivity. One way to handle this trade off situation is to use some kind of adaptive filter. If the noise level is high there is need for harder filtering and the filter constants can be changed. Similarly, with less noise, the filter can be adapted to the noise level. This approach has been tested with fairly good result, but there are some questions to answer. How often should one update the filter? How should the maximum/minimum filtering be chosen?

Another approach is to sample the derivative with a longer sampling time. This can be seen as filtering, but with only one parameter to be chosen, namely the derivative

sampling time. This method has been tested, and the result is hopeful. The strategy for when and how to update this sampling time is the following. The most important thing is that the sampling must be sufficiently fast in relation to the process dynamics. The parameter T is a measure of how long time the step response takes if the dead time is neglected. It is therefore logical to use this parameter when to chose the derivative sampling time. It has proven that it is sufficient to use a derivative sampling time of 0.5*T seconds. The time constant is updated after every step so it is natural to also update the derivative sampling time at these occasions.

Even better results are achieved when a combination of the two methods are used. First a low pass filter is used and then the sparse derivative method improves the result. The filter must not filter too hard, and its design is fixed. The adaptive PPI can cope with fairly large noise levels when using this solution. One restriction still holds though: the amplitude of the step must of course be larger than the noise. In order to update the controller at as small set point steps as possible, the noise level is detected continuously.

4.3.2 When to stop T_{ar} calculations

There are as mentioned three parameters to be estimated. Problems with the estimation of L were discussed in the previous section. The K_p parameter is the most simple to estimate, and noise does not disturb the estimation if only averaging is used. The remaining parameter, T, is not estimated directly but calculated from the estimates of T_{ar} and L (see section 3.1). This section treats problems with the T_{ar} estimation.

Since T_{ar} is estimated by calculating areas, the method is not that sensitive to noise. There are another problem though. How does one know when to stop the calculations of the areas? According to the equations (3.9) the calculations should stop when the process is at rest after the step. Since a controller with integral action is used, one can be sure that the control error will be zero. One way to check that the process is at rest is then to stop when the derivative is approximately zero and the new set point is reached. This will introduce the problem with derivatives and noise again, and therefore this criterion is skipped. Instead just the process mean value is measured and the calculations stops when it is sufficiently close to the set point. This does of course give rise to the question: “When is the mean process value close enough?” One could of course set a fixed percentage of the process value range as a limit, but simulations shows that smaller steps will then give poorer results. This is because the rise time in a linear system is independent of the size of the step, and accordingly it takes longer time for the process value to reach the limit if the step is of smaller amplitude. Therefore, the limit is decided as a percentage of the step size. Furthermore, it is necessary for the averaging to be done over some time to handle overshoots and large noise levels and therefore another criterion is added. The averaging is done for at least the length of three estimated time constants after the process has reached the new level.

4.3.3 Ramped set point changes

It is not necessary that the change in the set point is a distinct step in order to get a correct T_{ar} estimate. Theoretically, any change that is finite in time will give a useful result. In reality, there must be some restrictions to the shape of the set point change in order to implement the integral calculations. The final version of the adaptive algorithm handles two kinds of set point changes: a pure step and a ramped step. Ramped set point changes are popular in industry, since they give a safer behaviour of the step response. The problem that arose when implementing the ramped step was the definition of a ramp. The solution requires a change in set point in same direction every sample of the step. When the set point is constant for two consecutive samples, the step amplitude is fixed, and if the set point now changes before steady state around the new level is reached, the step are considered aborted and no new estimates are calculated.

4.4 Design parameters

During implementation, several decisions were made concerning numerical levels and constants. It is not sure that these decisions are the best. On the contrary, it is likely that they need to be adjusted if the algorithm is tested in an industrial environment. The parameters and the chosen values are listed in the table below. The names refer to the actual variables in the function block.

Name	Description	Chosen value
DerLimit	Sampling time for derivative.	MAX(0.5*T , 2*system sampling time)
Tf	Time constant for low-pass filter.	Tc/30 (Tc from relay tuning).
NoisePeriod	Period for updating noise level.	30 seconds.
SteadyLimit	Limit for mean value to be considered at steady state.	MAX(0.005*step amplitude , noise level)
----------	Time that mean-values must be steady before a step is considered ready.	3*T
NoiseLim	Fraction above old set point that indicates step for level estimates.	MAX(NoiseLevel/2 , 0.5% of operating range)

Table 4.1 Design parameters.

5 System integration

This chapter handles the issues of how the new adaptive PPI should be integrated in the existing PID Control Module. Things that are discussed are when the adaptive PPI should be used, how it should interact with the existing adaptation and what data should be given to / required from the user.

5.1 Allowed use of the PPI adaptation

In theory, there are no restrictions when the adaptive version of the PPI can be used. In reality it has been shown though that it is practical to have some conditions that have to be fulfilled before the adaptation can be switched on. Although it is not necessary for the controller to be of PPI structure, there must be integral action in the controller in order to eliminate the control error (see section 4.3.2). This means that the adaptive algorithm can be used with a PI or PID controller and if the ratio between the dead time and the time constant is increased, a PPI structure can be selected. How this switching is done is discussed in the next section.

There is no use to have a PPI controller when controlling an integrating process, i.e. an open loop unstable process. If the process is integrating, then the output after the step will be the same as before the step. This makes it quite easy to detect if the process is integrating. This test is made in the step auto-tuning, but since this can be detected during the first step in adaptation mode, there is no requirement for the full auto-tuning procedure before PPI adaptation is enabled. The relay auto-tuning must have been performed though. This is to be able to choose a suitable time constant for the derivative filter.

5.2 Changing control structure

The PPI adaptation algorithm uses the same design function as the existing auto-tuner and adaptation. A PPI structure is chosen if the dead time is longer than two estimated time constants, but what should be done if the design function suggests a PID controller? In section 5.1 it was stated that the PPI adaptation algorithm could be used even if the actual controller is not of PPI structure. The question is then if this should be allowed, and if the PPI adaptation should be run in parallel with the already existing adaptation for PI and PID controllers. This section presents and motivates a solution that, however, has not been implemented.

A PPI controller is a quite extreme, not very robust controller. If safe step responses with no overshoots are more important then a fast response, then one should use an ordinary slow PI controller instead. It is therefore not a good idea to have an adaptation that automatically switches from a PID to a PPI design if e.g. the dead time is increased. The question is if the user should be able to enable the adaptation to switch from PID to PPI design, or if PPI adaptation only should be used when the adaptation is started using a PPI design. The solution that has been chosen is the following. PPI adaptation is only possible if the adaptation is started when a PPI controller is in use. The user then have the option to choose which switching technique that should be used if the adaptation suggests a PID design. In every case, a PID design is chosen, but the user can choose to stick to a PID design even if the process enters the PPI region again, i.e. the PPI adaptation is turned off as soon as the

controller enters the PID region. The second alternative is to use a PPI design again if the adaptation finds it suitable. The third, and maybe the most secure alternative, is to turn off all adaptation when a switch in design. Which one of these techniques that are best have not been verified during simulations, see further chapter 6. The switching requires that the existing adaptation algorithm for PID control is being run in parallel with the PPI adaptation algorithm.

There is another decision to take when the structure is changed from PPI to PID. In the system today there exists two different design methods. The first is the original design that only requires a relay auto-tuning. This method is therefore called *relay design* in the future. The other design method is the method that is presented in [Norberg]. This method requires a step tuning, and is accordingly called *step design*. The existing adaptation uses relay design since the parameters needed for step design are not updated. The use of PPI adaptation makes it possible to use the step design as the structure is switched. This is used when PPI-PID switching is allowed, not otherwise since PPI adaptation is switched off then.

As mentioned above, this solution is not implemented in the test version. Instead, a graphical interface is constructed, which makes testing easy. The interface is shown in Figure 5.1.

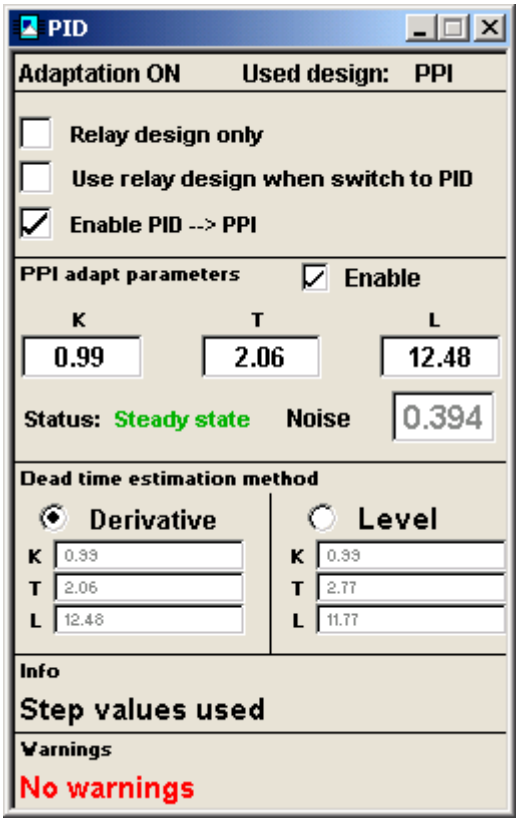

Figure 5.1 The adaptation interface.

5.3 User parameters

In this section, the issue of what parameters of the adaptation algorithm that should be available to the user is discussed. The goal is to leave the user with as few options as possible. The only choice the user has to make is which of the switching techniques above that should be used. To make the choice easer, the alternatives may be named safe (no switch), cautious (no PID to PPI) and normal (PID to PPI).

The user should also be warned if the adaptation works poorly, e.g. many estimates are rejected or ramped set point changes are performed in a bad manner. A warning must also be given if the adaptation is switched off.

6 Simulations

The best way of confirming that the adaptive PPI controller really works is of course to test it in several industrial environments. This would be very time consuming, and it might be hard to test all situations that can occur. It is therefore needed to test the controller in a simulated environment. This chapter gives a review over the simulations that have been carried out.

6.1 The processes

If finding processes with desired behaviour is a problem when testing in an industrial environment, one has the reverse problem when simulating. Since one is given the power to fully decide the dynamics, one might, with or without intention, use process models that give good results, but that does not capture the behaviour of 'real' processes. Further, the number of possible processes is infinitely, so how does one know which to use?

The PPI controller is, as mentioned in section 2.3, most beneficial for processes with short time constants in relation to the dead time. These processes can be found in the process industry. The continuous transfer functions of the processes that has been used in simulations are the following:

$$G_1(s) = \frac{1}{1+s}e^{-10s} \qquad G_2(s) = \frac{2}{(1+s)^2}e^{-10s}$$

$$G_3(s) = \frac{1}{(1+2s)^4}e^{-10s} \qquad G_4(s) = \frac{1}{(1+s)(1+4s)}e^{-10s}$$

$$G_5 = \frac{1}{(1+5s)^2}e^{-25s}$$

These models cover most of the process dynamics encountered in the process industry. The tests that have been carried out are presented bellow.

6.2 Test 1: Verifying good estimates

The first test is to check if the estimates are good enough to enable adaptation. The test was performed on the five test processes above with constant dynamics and no disturbances. For each process, several set point changes were performed and the estimates were studied. The estimates of the process gain, K_p, were in all cases good and the estimates of this parameter are therefore omitted in the results below. Set point changes of 5, 20 and 50% of the operating range were used in order to check the dependence of the step amplitude. Further, half of the steps were made using a ramped set point change. The test was performed with two parallel controllers, one with derivative dead time estimation and one with the level-method. The results are presented in Table 6.1 and Table 6.2 below.

Derivative:

Process	Calculated L	Estimated L	Calculated Tar	Estimated Tar	ΔL when ramp
1	10.00	10.00	11.00	11.29	10.13%
2	10.60	10.53	12.00	11.92	12.50%
3	14.00	14.29	18.00	18.11	7.76%
4	10.88	11.62	15.00	15.11	11.55%
5	27.90	28.76	35.00	35.14	0.40%

Table 6.1 Results from Test 1 using derivative dead time estimation.

Level:

Process	Calculated L	Estimated L	Calculated Tar	Estimated Tar	ΔL when ramp
1	10.00	10.14	11.00	11.37	0.89%
2	10.60	10.28	12.00	11.92	2.34%
3	14.00	12.69	18.00	18.03	0.01%
4	10.88	10.88	15.00	15.09	0.95%
5	27.90	26.70	35.00	35.38	-0.50%

Table 6.2 Results from Test 1 using level dead time estimation.

Notice that the calculated values are not similar to the real dead times, but the values that are given by a first order approximation. The values for the estimated dead time L and average residence time T_{ar} in the table above are mean values for the values from the non-ramp steps. The deviation from the mean values for the dead time was quite small, at most 1.7%. The adaptation did never result in unstable control even though the performance was worse for some cases. For processes G_3 and G_4, the level estimation did show a switching behaviour between PPI and PID design. The algorithm did practically change control structure every step, which of course is an undesired behaviour. The last column shows the mean change in dead time estimates if ramped set point changes were used. One can notice that the level dead time estimation is almost unaffected while the derivative dead time estimate increases drastically for all processes except process G_5. This process is over all the most robust, and control was good for both level and derivative estimation. Furthermore, the step amplitude did not affect the derivative estimates but the level estimates became shorter when larger step amplitudes were used. The reason for this is that the limit for step response detection is independent of the step size, and so is the rise time of a linear system (compare section 4.3.2).

6.3 Test 2: Noise sensitivity

As will be shown in this section, noise sensitivity is a large weakness of the adaptation algorithm. The effects of a variety of disturbances are investigated below.

6.3.1 White noise

First, uncorrelated white noise was studied. Processes G_2 and G_3 were disturbed with white noise with a variance of 0.8% of the operating range. Several set point steps were made, and the estimates of L and T_{ar} were studied. The algorithm sometimes discarded 5% steps, and therefore amplitudes of 10, 20 and 50 % were used. The graphs in Figure 6.1 and Figure 6.2 show comparisons between the dead time estimates with and without noise. The variations of the estimates are considerably increased. One can also notice that the derivative estimates seem to have the same mean value as without noise, while the level estimates seem to become a little longer.

This is caused by the new level detection limit when noise is present. The T_{ar} estimates are changed in a similar way.

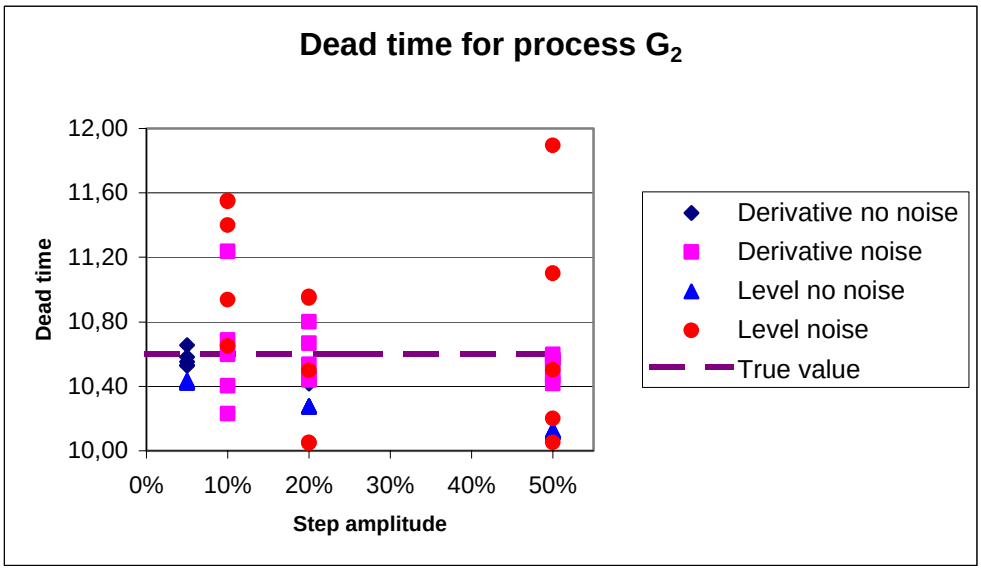

Figure 6.1 Comparison between dead time estimates with and without noise for process G_2.

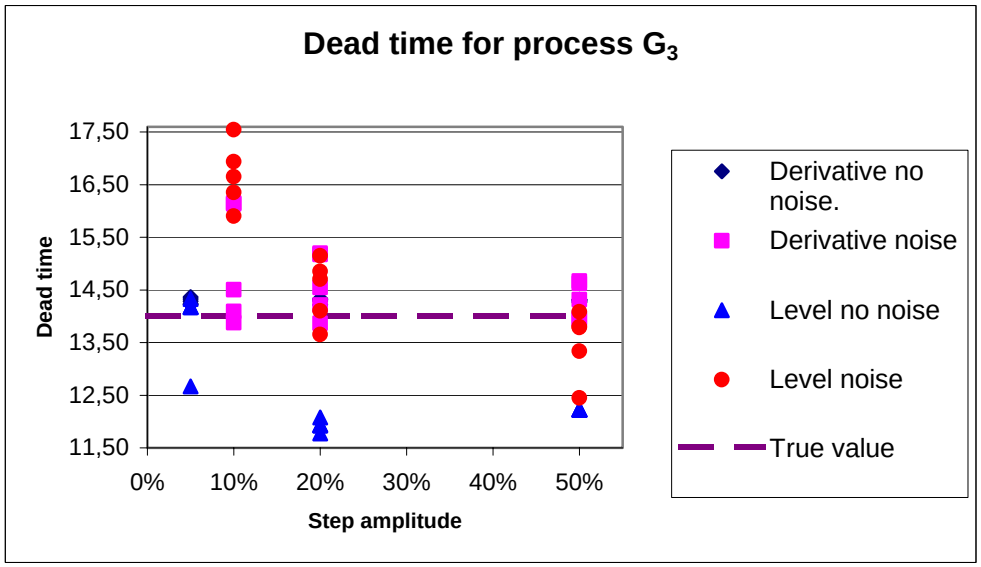

Figure 6.2 Comparison between dead time estimates with and without noise for process G_3.

Even though the quality of the estimates was decreased, the control was still stable. Figure 6.3 and Figure 6.4 show typical step responses for process G_3.

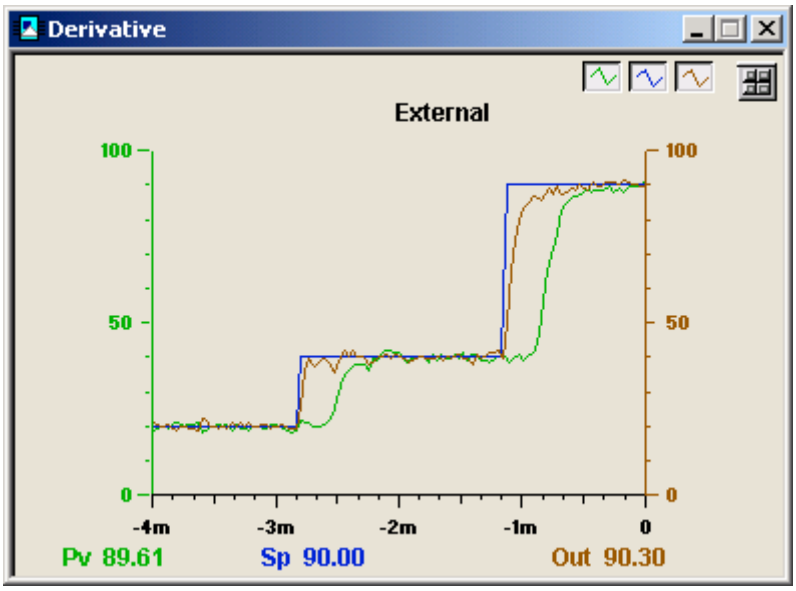

Figure 6.3 Step responses for G_3 with derivative dead time estimates.

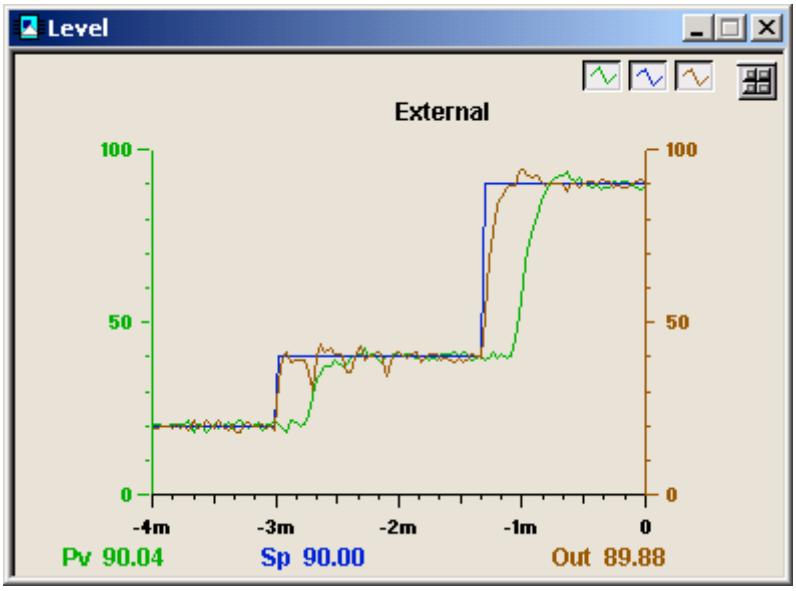

Figure 6.4 Step responses for process G_3 with level dead time estimates.

The performance was worse when ramped set point was used, especially for process G_2. The system became unstable several times, see Figure 6.5.

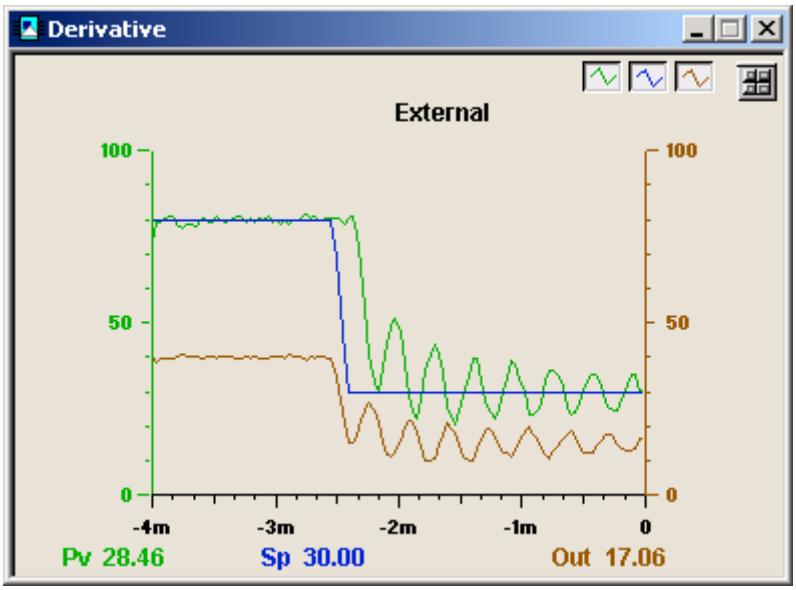

Figure 6.5 Unstable ramp response for process G_2 disturbed with white noise.

6.3.2 Coloured noise

The coloured noise used in this section was created by filtering white noise with a low-pass filter. The filter was designed to reduce frequencies above the sampling frequency. The white noise had a variance of 5% of the operating range. Process G_2 was used and the result was quite similar to the one in the previous section. The level estimates became much worse though. The dead time was estimated much too short in some cases resulting in a PID design. This was caused by a ramp-like disturbance that the algorithm interpreted as the step response.

6.3.3 Sinusoidal disturbances

For sinusoidal disturbances with low frequency in comparison with the sampling frequency, the behaviour was similar to the case with coloured noise, i.e. too short level dead time estimates. For medium high frequencies, the result was somewhat different. Figure 6.6 and Figure 6.7 show typical behaviour of process G_2 when disturbed by a sinusoidal signal with amplitude of 2% and a frequency of 10 rad/s or 1.6 Hz. Too long dead time estimate made the level method oscillatory. The derivative alternative was good as long as ramped steps were not used. If they were, the result became worse, which can be seen in Figure 6.8. Higher frequencies than the Nyquist frequency will lead to frequency folding and similar result as for low frequencies.

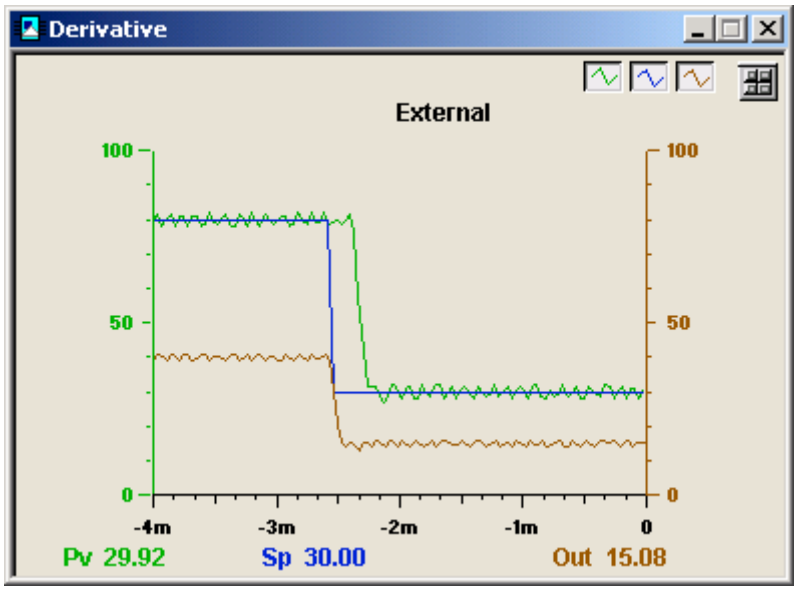

Figure 6.6 Stable step response with a sinusoidal disturbance and derivative dead time estimation.

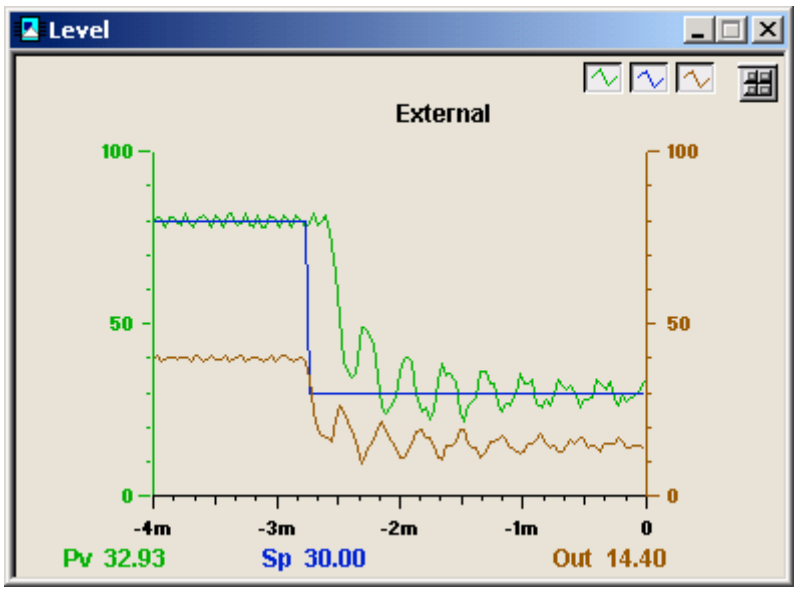

Figure 6.7 Unstable step response with sinusoidal disturbance. Level dead time estimation is used.

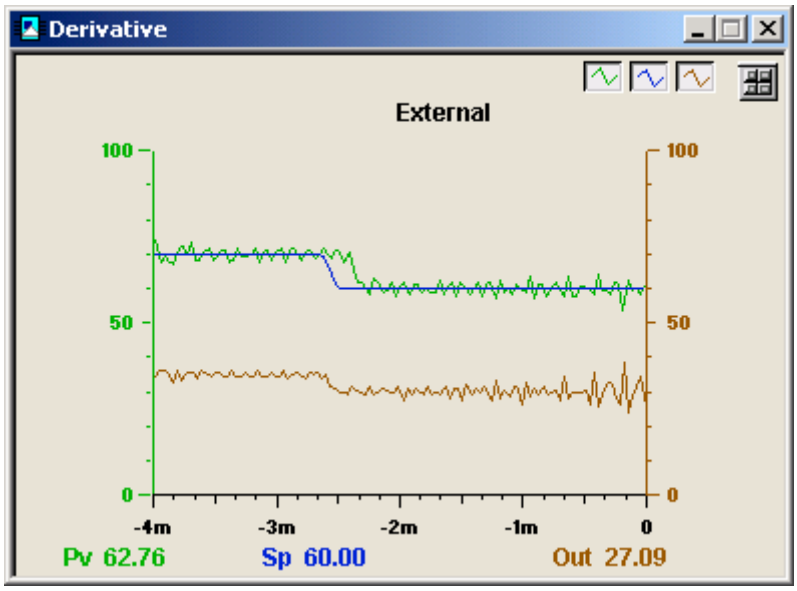

Figure 6.8 Unstable ramp response when process G_2 is disturbed by a sinusoidal signal. Derivative dead time estimation is used.

6.3.4 Load disturbances

In order to check the dependence of load disturbances during the step response, this test was performed on process G_2. Before a set point step, a load disturbance of 5% amplitude was introduced in the control signal. First, the load was made in the same direction as the step response. The results can be seen in Figure 6.9 and Figure 6.10. After the step was finished, a new load disturbance was introduced in order to check the performance of the controllers. The level method gave a too short dead time, which led to a switch to PID design. The derivative method was not that disturbed.

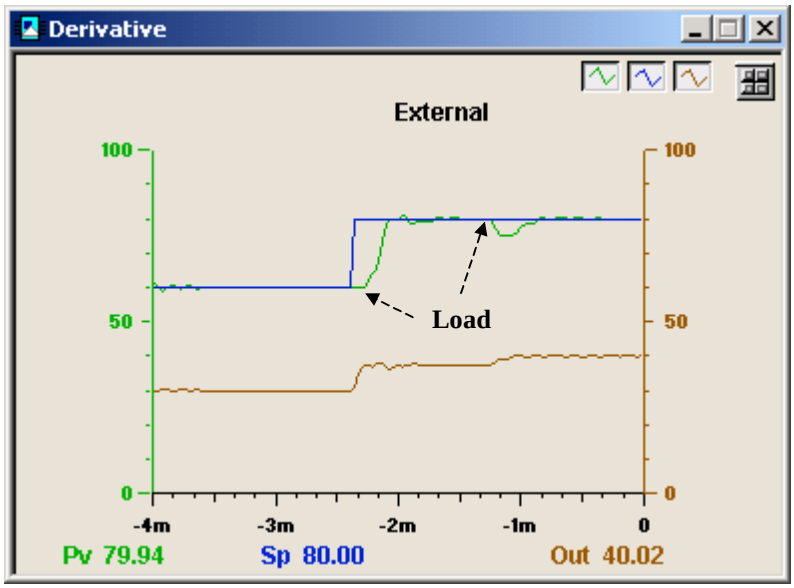

Figure 6.9 Step response inferred with a load disturbance.

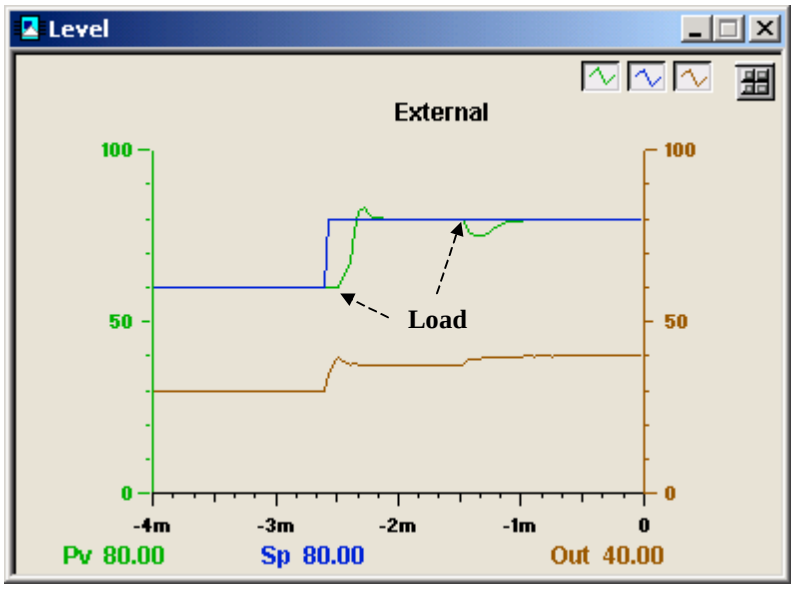

Figure 6.10 Result with load in same direction as the step response.

Then, the same procedure was repeated, but with the load acting in the opposite direction. The result can be seen in Figure 6.11 and Figure 6.12. The load disturbances are clearly shown as a dip just before the step response. Now, the derivative estimate was too long, and the result was an oscillatory control. The level estimate was still to short, and a PID design was chosen.

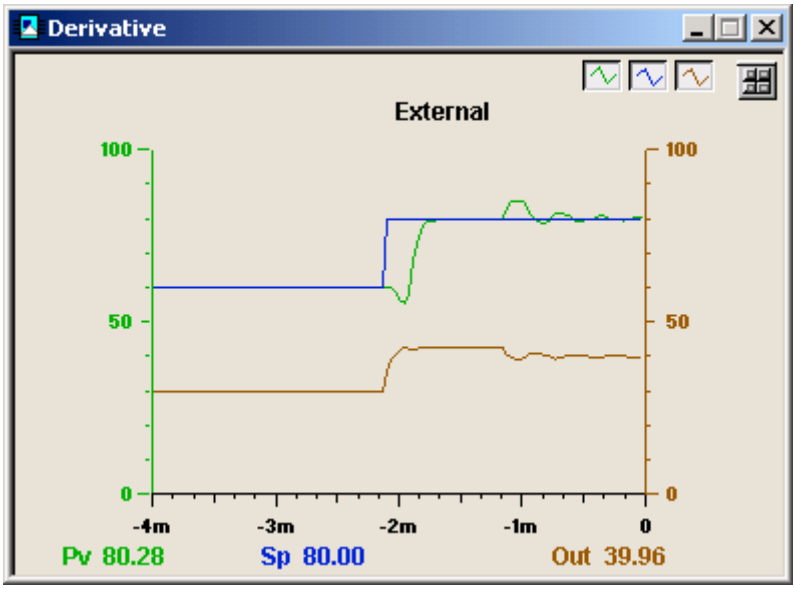

Figure 6.11 Step response disturbed by a load in the opposite direction.

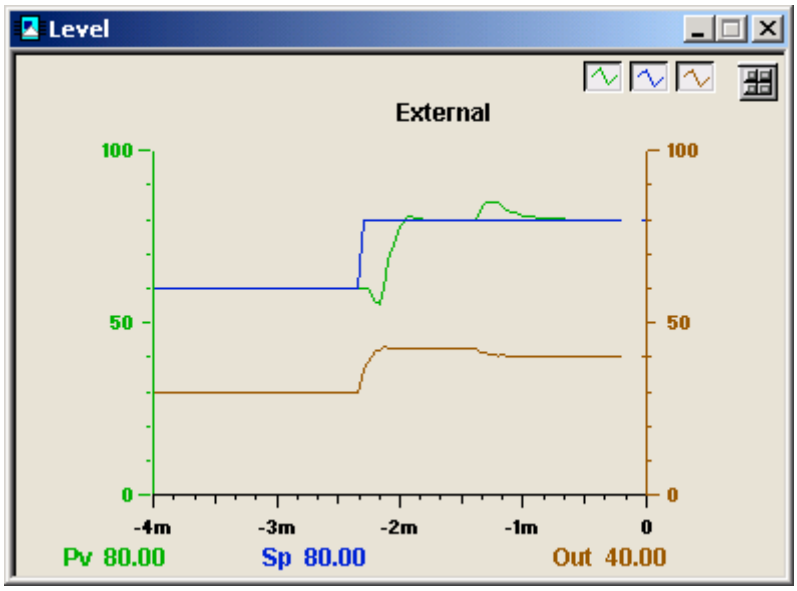

Figure 6.12 Result with load in opposite direction of the step response.

6.4 Test 3: The utility of adaptation

In the two previous tests, only processes with constant dynamics are considered. This might seem a little strange since the meaning of adaptation is to track changes in the process. It is very important though that the behaviour when the dynamics are fixed is good. If the adaptation makes the control worse when there are no changes in dynamics, the whole idea is of no use. This is the reason why constant dynamics has been used so far. Now, when the stability of the estimations is proved, the focus can be turned towards the ability to track changing dynamics.

In this section, only changes in the dead time are considered. This is because the PPI is most sensitive to this kind of changes and a change in the dead time effects both the L- and T_{ar}-estimate. Simulations with changes in other dynamics have also been performed, but not as extensive as in the case with changing dead time. The results from these simulations are omitted here since they do not differ in quality from the one that are presented.

First, a change in the dead time for the process G_4 was considered. The measured signal was distorted with white noise. The estimate sequence for L and T_{ar} are shown in Figure 6.13. After the 9^{th} estimate, the dead time was changed from 10 to 15 seconds. The performance of the 10^{th} step response was of course not that good, but the important thing is that the estimates quickly assumed their new values. Since the first step after the change is bad, it sometimes happens that the T_{ar}-estimate is poor. One can see that the L-estimate converged immediately, whereas it took three steps before the T_{ar} estimate had found its correct value. This is of course an undesirable feature since set point changes are not supposed to occur that often.

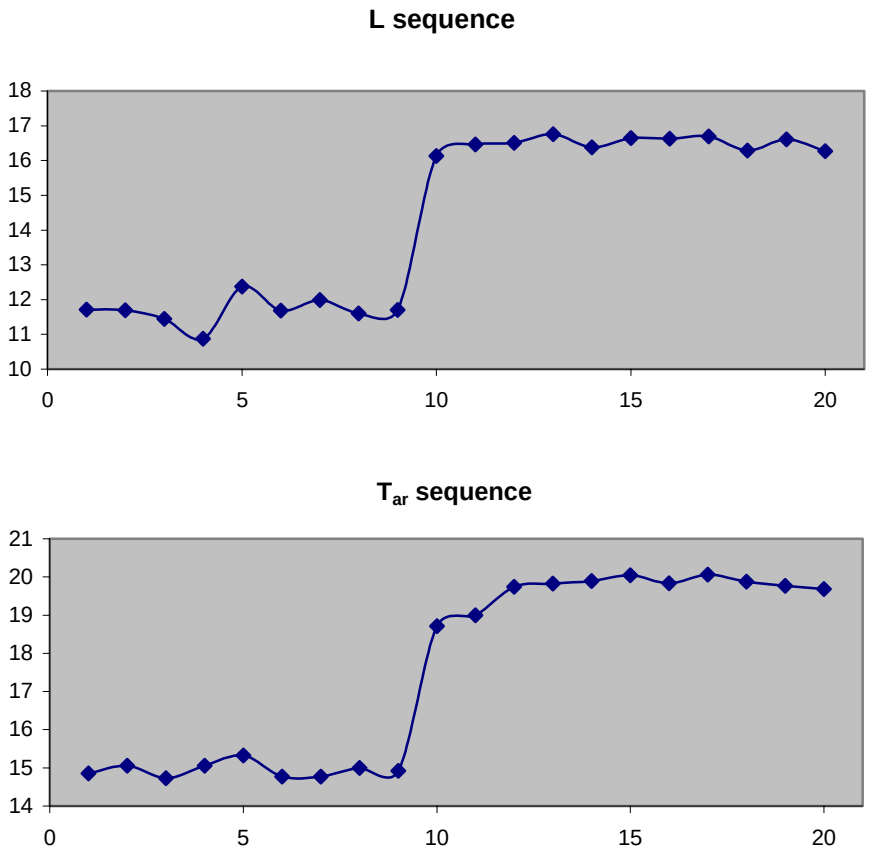

Figure 6.13 L and T_{ar} estimates. Before the 10:th estimate the dead time is changed from 10s to 15s.

In the previous example, the change was sudden and quite large. This might of course happen, but a more realistic scenario is a slowly drifting dead time. If there is set point changes reasonably frequent compared to the drifting rate, then the adaptive PPI should be able to handle this and increase the performance. To verify this, a parallel test was performed. The process used was G_4 but with static gain of 2. Two controllers, one with and one without adaptation were tuned identically. Then, the process dead time was increased in small steps. The changes were made simultaneously for the two controllers. A set point step was made after each change in dead time so that the adaptation could track the new dynamics. Figure 6.14 and Figure 6.15 show the results of a set point step when the dead time has changed from 10 to 12 seconds. The first graph shows the adaptive controller while the second is from the controller without adaptation. It is clear that the adaptation improves the performance, even though the difference is not that big.

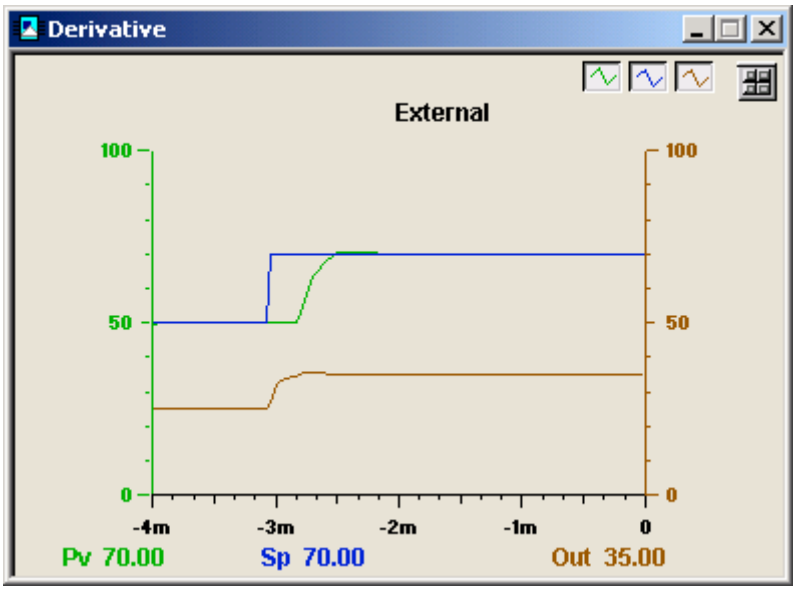

Figure 6.14 Step response with adaptation when dead time is 12 seconds.

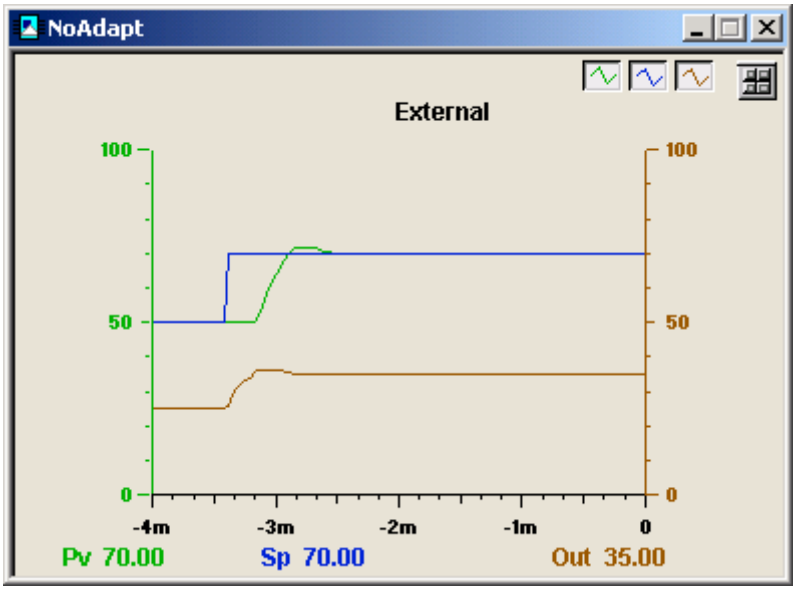

Figure 6.15 Step response without adaptation when dead time is 12 seconds.

Then, the dead time was increased further. In Figure 6.16 and Figure 6.17, the result when the dead time was 14 seconds is presented. After the step, a load disturbance was introduced to the system. The difference in performance between the adaptation and the static controller are much more noticeable here.

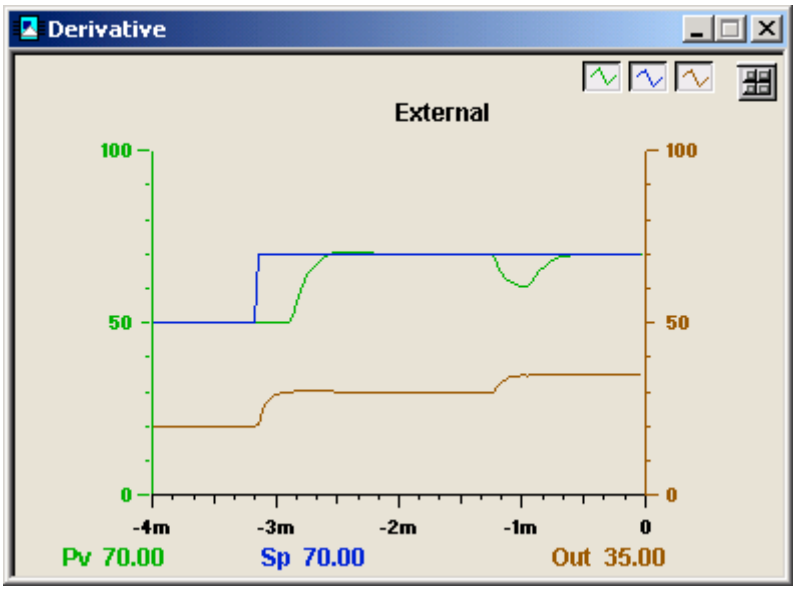

Figure 6.16 Step response with adaptation for dead time equal to 14 seconds.

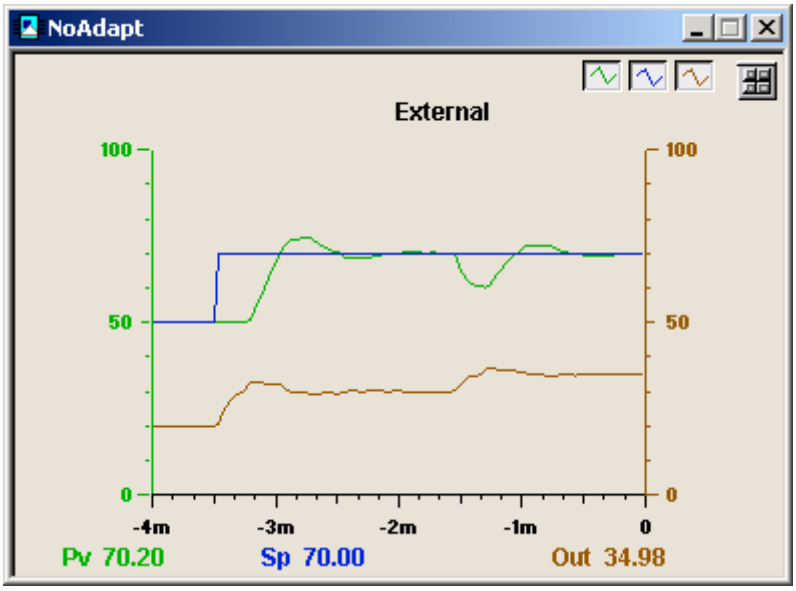

Figure 6.17 Step response without adaptation and with dead time of 14 seconds.

The conclusions that can be drawn from these simulations are that the algorithm is able to handle changing processes. If the changes are too big between two set point steps, however, lack of robustness in the PPI controller can make the system unstable, and adaptation will fail.

6.5 Test 4: Switching to PID control

Three different techniques for switching between PID and PPI design during adaptation are presented in section 5.2. The test in this section was performed in order to check the performance of these. The test was done using process G_4 on two parallel controllers. The controllers were first auto-tuned. One controller used the 'relay when

switch' option, the other used step-design all the way. Both used derivative dead time estimation in PPI mode and level estimation in PID mode. A third controller was also used without adaptation, as a reference.

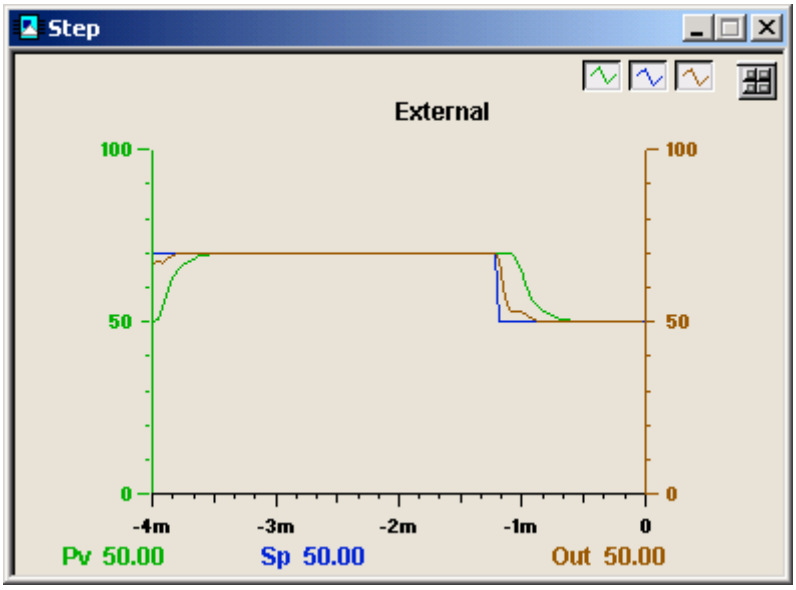

Figure 6.18 Step response for the controller that always uses the step adaptation. The step is the last one before the system switched to PID design.

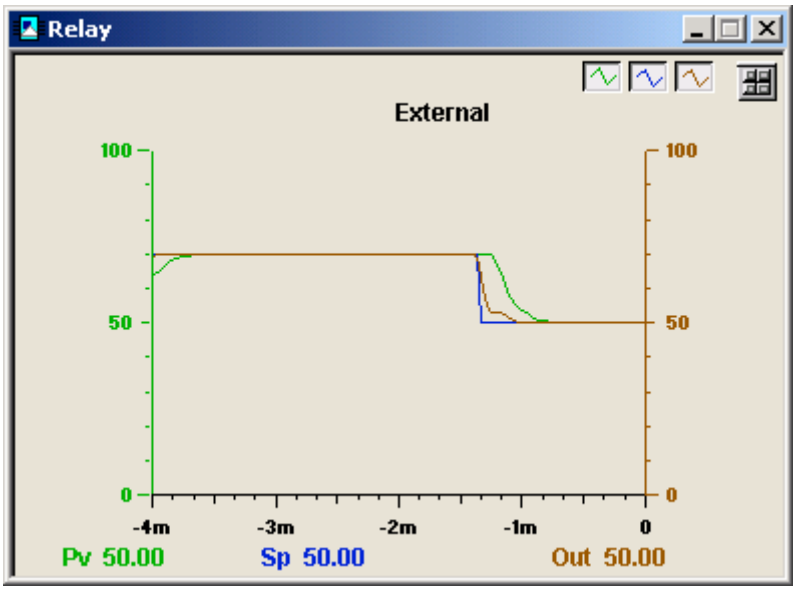

Figure 6.19 Last step response before switch for the controller that uses relay adaptation in PID mode.

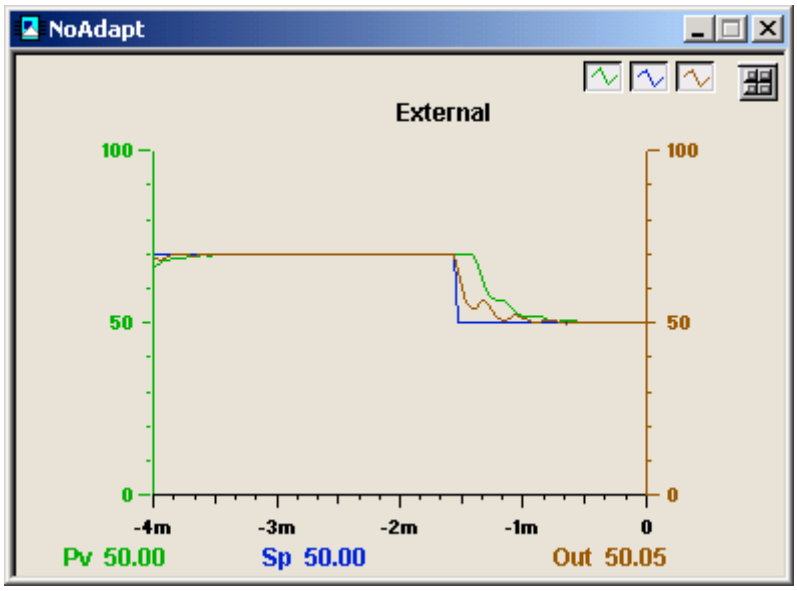

Figure 6.20 Last step response before switch to PID design for reference controller.

First, the dead time was decreased in small steps from 10s to 6.5s. After every change in dead time, a set point step was made in order to activate the adaptation. The results from the steps taken when the dead time was 6.5 seconds are shown in Figure 6.18, Figure 6.19 and Figure 6.20. After this step response, both the adaptive controllers (named Step and Relay) switched to PID structure. In Figure 6.21 and Figure 6.22, the first step responses after the switch for the two adaptive controllers are shown. The step design gives a slightly faster response.

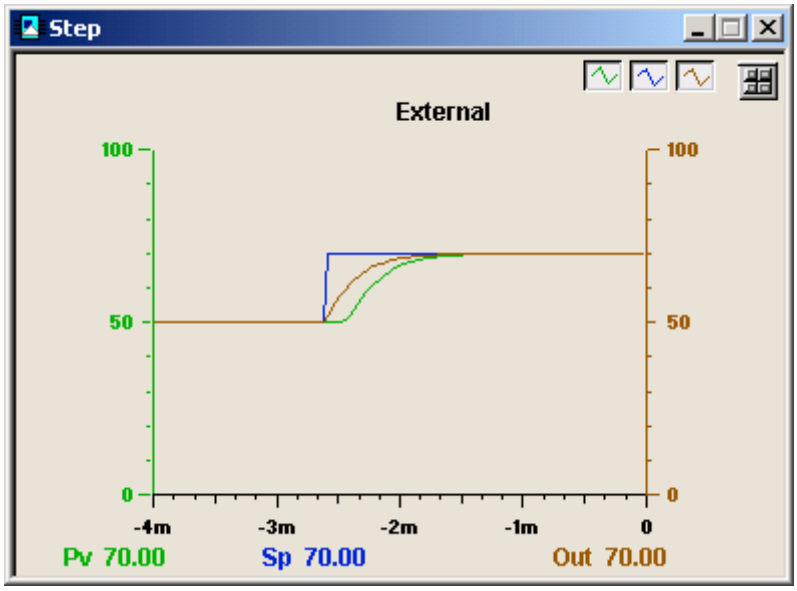

Figure 6.21 First step after switch for the step adaptive controller.

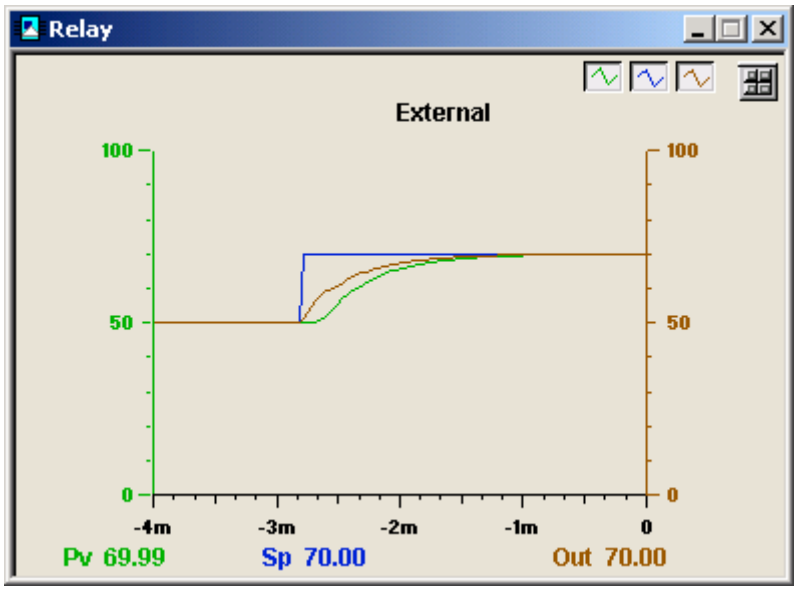

Figure 6.22 First step response after switch for the relay adaptive controller.

The PPI adaptation was then switched off for the relay controller according to section 5.2. The dead time was decreased further, and as shown in Figure 6.23 and Figure 6.24, the relay design was now a little better. This is probably caused by the fact that the relay adaptation updates more frequently. The dead time was 5 seconds in this step.

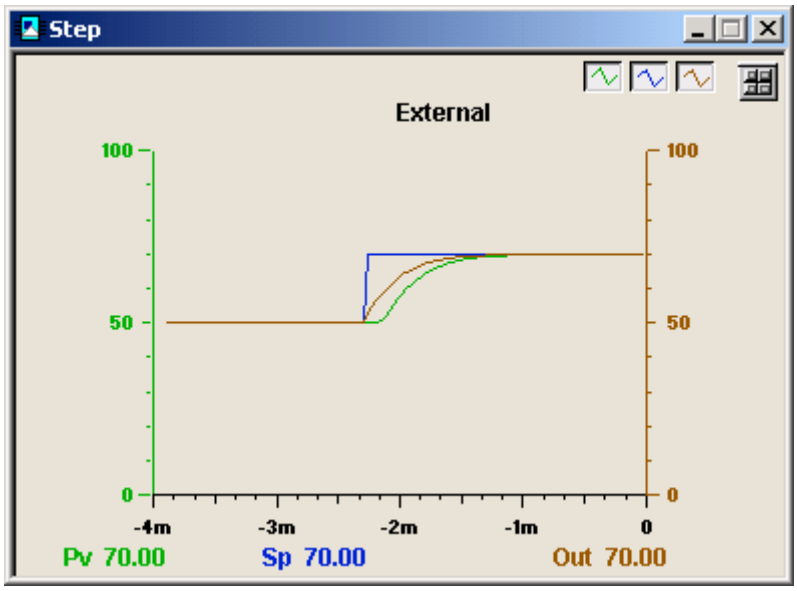

Figure 6.23 Step response for step adaptation when dead time is 5 seconds.

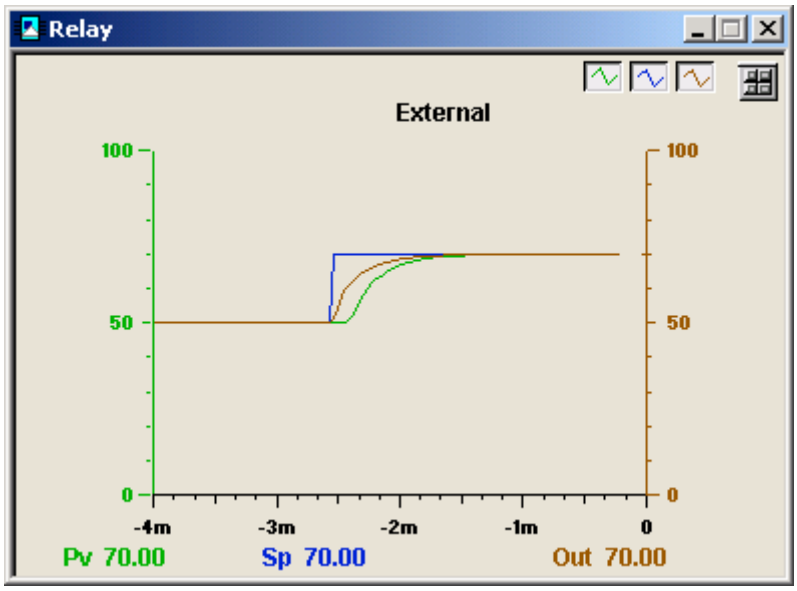

Figure 6.24 Step response for relay adaptation when dead time is 5 seconds.

Now, the dead time was increased again and at 9 seconds did the step adaptation switch back to PPI. In Figure 6.25 and Figure 6.26 a step response when the dead time is 11 seconds is shown. Notice the very slow result for the relay adaptation.

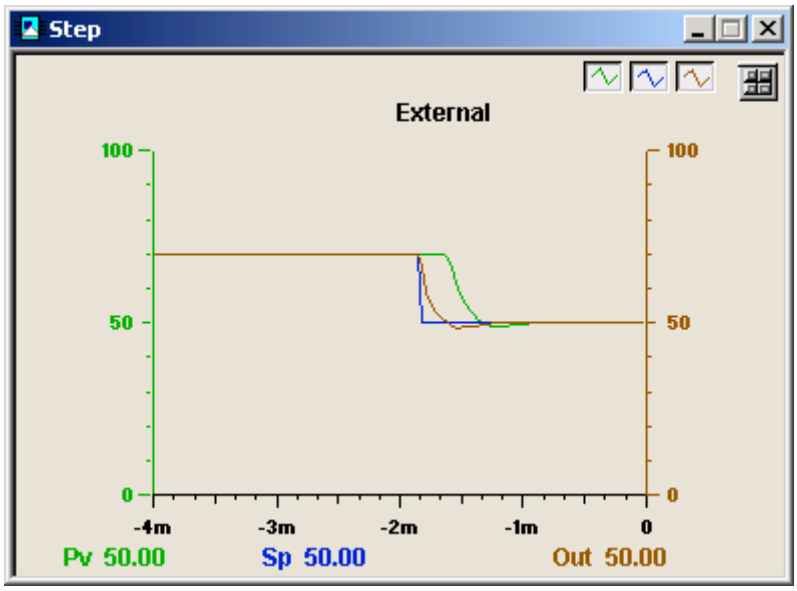

Figure 6.25 Step response when dead time is 11s. PPI design is used again.

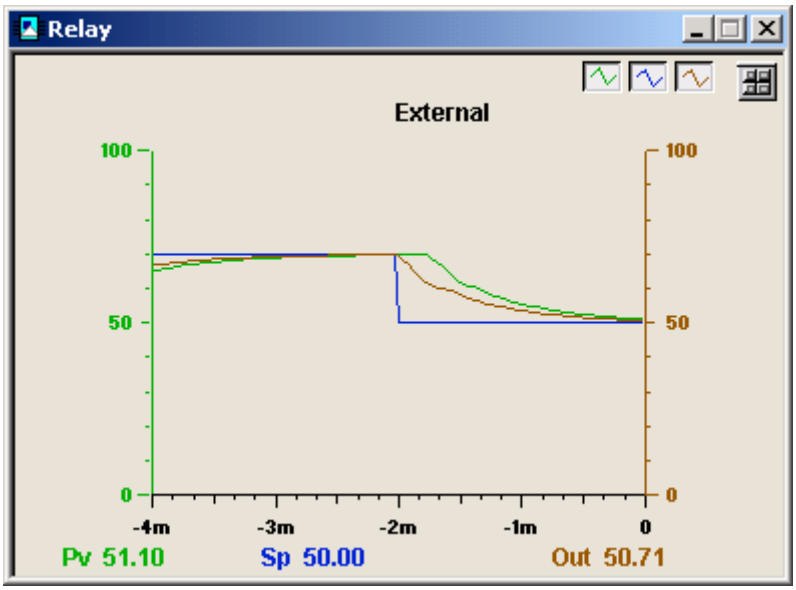

Figure 6.26 Step response when dead time is 11s. PID design is still used.

One remark is suited here. The step adaptation works fine above, but one should remember that the algorithm is very depending on frequent set point changes. Furthermore, only variations in dead time are considered.

7 Conclusions and future improvements

In this chapter, the results from the simulations are discussed and conclusions about the adaptation are made. In section 7.2 improvements are suggested.

7.1 Conclusions

One can see from the simulations without noise (section 6.2) that the choice to use the level estimating technique when ramped set point changes are used is well motivated. The problem is that the level method seems very sensitive to noise. Further tests are needed in order to see if this is a possible solution. If this isn't so, and if the level estimates cannot be done more noise insensitive, the only solution is to disable the adaptation during ramped set point changes. This would be a big drawback.

There are some processes that are more sensitive to noise than others, especially G_2 shows bad noise behaviour. The reason for this is that the time constant is so fast that even small deviations in the dead time estimate will lead to a very small (<0.05 s) T estimate, which in turn lead to an unstable controller. This is of course not good and maybe should the supervision from section 3.2 be modified to discard such small time constant estimates. Another remark is that there obviously is no good to use a PPI controller if the system is that fast. This indicates that the condition for when a PPI design should be chosen is insufficient. This will be further discussed in the next section.

The problem of load disturbances during adaptation is a known problem. The existing PID adaptation is turned off if a load disturbance is noticed. The problem is that it is not a trivial thing to detect a load disturbance.

The test of the switching techniques is not really giving any clear answer on which method that should be used. There are already problems in the existing adaptation when switching between PID and PI control. This indicates that switches might not be allowed at all. This may be a good solution since the whole idea of adaptation is that the changes in the process must be small. There might be problem when the process is close to a switching point though. In these cases, some kind of hysteresis in the switching would be good to avoid repeated adaptation shut downs.

In total, one must say that the results are good though. Test 3 shows that a higher performance is given when the adaptation is used. The breaking point for the PPI adaptation is, however, the fact that the controller can be updated only once per set point step. If set point steps are that rare that is feared, there is no meaning in putting any effort into including the adaptive PPI in a product. To verify if this is the case, industrial studies must be performed.

7.2 Improvements

One thing that has been discovered is that some processes that are suggested a PPI controller by the auto-tuner, in reality is more suited for PI or PID control. They seem to be very sensitive to changes in the process dead time. This indicates that the criterion for when PPI design should be used is somewhat insufficient, or a more robust tuning technique is needed. There is some work done on this subject (see [Ingimundarson]) in which the delay margin, i.e. the amount the dead time can vary from the used L before the system becomes unstable, is used. If one could somehow estimate this parameter, it could be used in tuning and design.

As mentioned above it seems like the criterion for when to use a PPI controller is not hard enough. Today the rule is to use the PPI controller when the estimated dead time is about twice the estimated time constant, i.e. $\frac{L}{T} > 2$. Some additional simulations have proven that there is a need to limit the use of the PPI when the dead time is too long in comparison with the time constant. The suggested new rule is therefore to use the PPI controller when $2 < \frac{L}{T} < 5$ holds. This would then give that processes G_1 and G_2 from the simulations should be controlled with PID controllers instead. It has also shown that first order processes are less suited for PPI control. The lack of dynamics in the system makes it very hard to tune the controller and the control loop becomes sensitive to changes in the process dynamics. More research can be done on this topic.

On the matter of improving the PPI adaptation algorithm, the most important subject is the noise sensitivity. There is also some work to do on the supervision. One idea that has not been tested is to somehow use the old estimates, i.e. not to use the new estimates straight off but a mean value of the most recent estimates. This would lead to a more slowly adapting controller that will need set point steps more often. On the other hand, the adaptation might be more robust if this averaging is done.

Acknowledgements

First of all I would like to thank my advisors, Lars Pernebo at ABB Automation Products and Prof. Tore Hägglund at the Department of Automatic Control at Lund Institute of Technology. Their inspiration and support have made it a pleasure to complete this thesis. I would also like to express my gratitude to all the people at ABB that have helped me, especially Bengt Hansson and Mikael Petersson.

References

K. J. Åström and T.Hägglund (1995): PID Controllers: Theory, Design and Tuning. Instrument Society of America, second edition.

K. J. Åström and B.Wittenmark (1997): Computer-Controlled Systems. Prentice Hall, third edition.

T. Hägglund and K.J. Åström (1991): Industrial Adaptive Controllers Based on Frequency Response Techniques. Automatica, Vol.27, No.4, pp. 599-609.

T. Hägglund (1996): An industrial dead-time compensating PI controller. Control Engineering Practice, Vol. 4, No.6, pp. 749-756.

A. Ingimundarson (2000): Robust tuning procedures of dead-time compensating controllers. Master thesis, Department of Automatic Control, Lund Institute of Technology.

A. Isaksson, A.Horch and G. Dumont (2000): Event-triggered deadtime estimation – comparison of methods, Control Systems 2000, pp 209-215, British Columbia.

A. Norberg (1999): Kappa Tuning – Improved relay auto tuning for PID controllers. Master thesis, Department of Automatic Control, Lund Institute of Technology.

Control^{IT}, AC800M/C Control Functions: Alarms, Analog Control and Controllers. Version 2, User's Guide. ABB Automation Products.

Hands on Control Builder Professional. Manual, ABB Automation University.