ISSN 0280–5316
ISRN LUTFD2/TFRT--5559--SE

Improvements of a PID Controller

Edith Dufrêne

Department of Automatic Control
Lund Institute of Technology
May 1996

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
May 1996

Document Number
ISRN LUTFD2/TFRT--5559--SE

Author(s)
Edith Dufrêne

Supervisors
L. Pernebo (Alfa Laval), K. J. Åström (LTH), Tore Hägglund (LTH)

Sponsoring organisation

Title and subtitle
Improvements of a PID Controller. (Förbättringar av en PID regulator).

Abstract

This paper describes three improvements for a PID controller. These improvements have been created with SattLine, which is an object-oriented, distributed process automation system developed by Alfa Laval.

These improvements are a compensator for static friction in control valves, an interpolation between the PID values when the system is non linear and the creation of a table to save different gain schedules.

The two first improvements have been tested in simulation and in a plant, Danisco Sugar, the third one has just been tested at Alfa Laval.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
English

Number of pages
39

Recipient's notes

Security classification

Abstract:

This paper describes three improvements for a PID controller. These improvements have been created with SattLine, which is an object-oriented, distributed process automation system developed by Alfa Laval.

Theses improvements are a compensator for static friction in control valves, an interpolation between the PID values when the system is non linear and the creation of a table to save different gain schedules.

The two first improvements have been tested in simulation and in a plant, Danisco Sugar, the third one has just been tested at Alfa Laval.

IMPROVEMENTS OF A PID CONTROLLER

My Master Thesis has lasted for a bit more than four months. It is the final part of my engineer degree which has totally lasted five years. I have spent my last year of study at Lund in Sweden. So I have done my Master Thesis at Alfa Laval Automation In Malmö, in the development and research department. My supervisors are Lars Pernebo at Alfa Laval and Tore Hägglund and Karl Åström at LTH.

During these four months I have done three different projects. The first project consists in creating a compensator for static friction in the valves, the second consists in creating interpolation the PID parameters between different levels, and the third consists in a gain scheduler with several gain scheduling references. With my projects I have learned how to use SattLine which is a process control system developed by Alfa Laval.

The number between parenthesis corresponds to the reference book.

TABLE OF CONTENTS

ALFA LAVAL AUTOMATION

Alfa Laval Automation is one of the main suppliers of process control and automation equipment in Europe and is rapidly growing in others parts of the world. The company is part of the Alfa Laval Group, which has some 11.000 employees in 110 different companies worldwide.

The high level of knowledge and expertise is based on many years of experience with components and system solutions, backed by strong R&D activities.

The company is one of the five largest PLC suppliers in Europe. Apart from conventional PLCs the company also supplies DCS-PLCs, man machine interface (MMI), supervisory systems, management information systems and materials handling systems. Transmitters and single loop controllers and distributed valve control systems are also part of the range.

SattCon, SattLine, SattGraph, SattTop, ECA and SattStore are well known brand labels of Alfa Laval Automation.

SattLine:

Sattline is an object-oriented, distributed process automation system. Supervision and control are integrated into one total concept consisting above the control system that handles the process-related functions.

The unique feature is that all SattLine's functions are configured in the same system in one program. Operator functions and control programs can both be performed from one and the same workstation.

SattLine features sequence control, interlocking calculations, analogue control, operator interaction, graphics, supervisory functions such as history storage, recipe and batch handling, alarm and event handling, optimization calculations, production queuing, etc.

The logical graphical display gives the user a full, realistic and understandable picture of the process.

SattLine system offers a flexible choice of computer platforms, giving the user optimal performance in all situations. The workstation can run on PC systems using Windows NT operating system, DEC Vax Station or Alpha stations using Open VMS operating system.

SattLine is a truly distributed system, both the I/Os and the CPUs can be installed close to the process. This will save costs both in installation and maintenance.

Programming of SattLine is object-oriented. In effect this means that you define a number of process objects and the operations they are to perform in the application in question. Each object mirrors the build-up of the process and contains all the information it needs to perform its task in the process - graphic representation, control programs, operator interaction and a lot more. An object

may be small or large, simple or complex. It may be single pump, a mixer, a tank or a complete process stage comprising reactor tank valves, mixer, recipes, alarms, reports, etc. Object-orientation enhances system safety and also encourages users to make changes where perhaps they may not have done so before because of the time and inconvenience it caused. Process experience is automatically built into the system, stored in the powerful library function, and then used in other parts of a plant or in future projects. Grafcet is used for sequential operations while ladder diagrams, function blocks or textual representation are used as required for interlocking and calculations.

Sattline has a number of integral functions that extend the capabilities of the control language from the equation and sequence level to enable to handle FIFO queues, different kinds of arrays and tables for storing records of information, trigonometry functions, conversion of text to speech, etc.

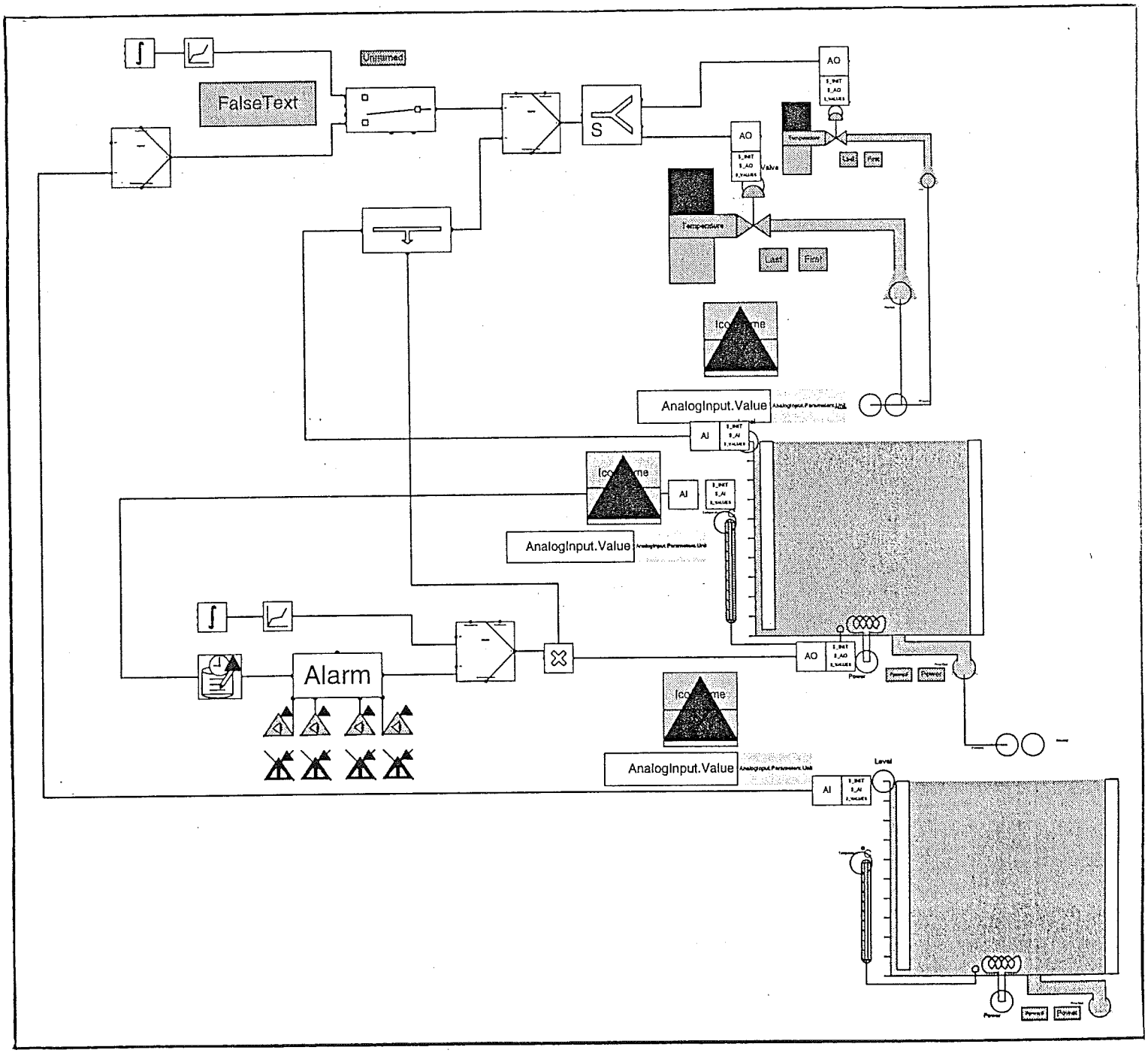
SattLine's basic functions are readily accessible from libraries. The functions have been created in SattLine language and are fully accessible so that, in turn, you can create user-defined modules with elements that are unique to a specific module.

All parameters can be changed on-line via pre-defined menu windows that are integrated in the control modules. Pictures of groups of controller loops can be created using predefined panels. All parameters are accessible to other programs in the system. This enables special functions to be integrated and gives you an extensive choice of options.

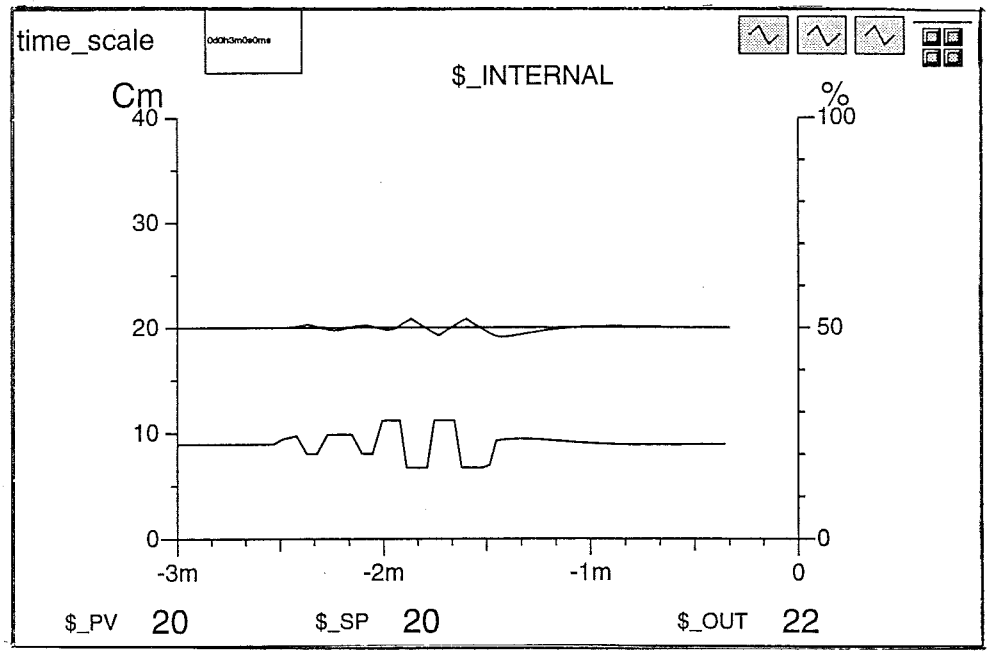
WHICH SIMULATION I USE

All my code are tested on a SattLine process which includes two tanks and two PID controllers. The tanks are 40 cm high and 3 cm of radius. The first tank is filled by two valves and the second tank is filled by the outgoing of the first tank. The outgoing of the valves is proportional to the flow.

In my simulation I have worked with the first tank, the first controller and the valves. And with my projects I have implemented some codes on the model of the valve and on the PID controller.



To get the values of the PID, we can use an autotuner. It calculates suitable PID parameters from a simple relay feedback experiment. When the operator wants to tune the controller, the controller is disconnected and a function which can be described as a relay with hysteresis is introduced in the control loop. The effect of the relay function is an ON/OFF control which generates a controlled and stable oscillation in the process signal. The amplitude of the oscillation is automatically controlled to a minimum value by adjusting the relay amplitude. From the period and amplitude of the oscillation, suitable controller parameters are calculated. It is possible to change the speed of the controller. Three speeds are available: Normal, fast and slow.



Project 1:

THE KNOCKER

COMPENSATOR FOR STATIC FRICTION IN CONTROL VALVES

When control valves are used, some friction appears. The friction brings a bad functioning of the valve, the controller is no more effective. The aim of this project is to find a compensator for this friction, called knocker. The nonlinearities in the control valves are the largest source of process variability in process control, they cause an increased energy consumption, vast of raw material and sometimes a less uniform end product.

This project is trying to improve the control when a stick-slip motion is detected . Such an improvement will not only yield more effective production. It may also delay the time until the next stop of the production.

This project is divided in six different parts

- An analysis of why there are oscillations.
- How oscillations are detected.
- Diagnosis.
- Control valve
- Knocker.
- Results

I - Reasons for oscillations in feedback loops

The oscillation loads are divided into two groups: these with low frequency and those with high frequency.

I - 1 Low frequency loads

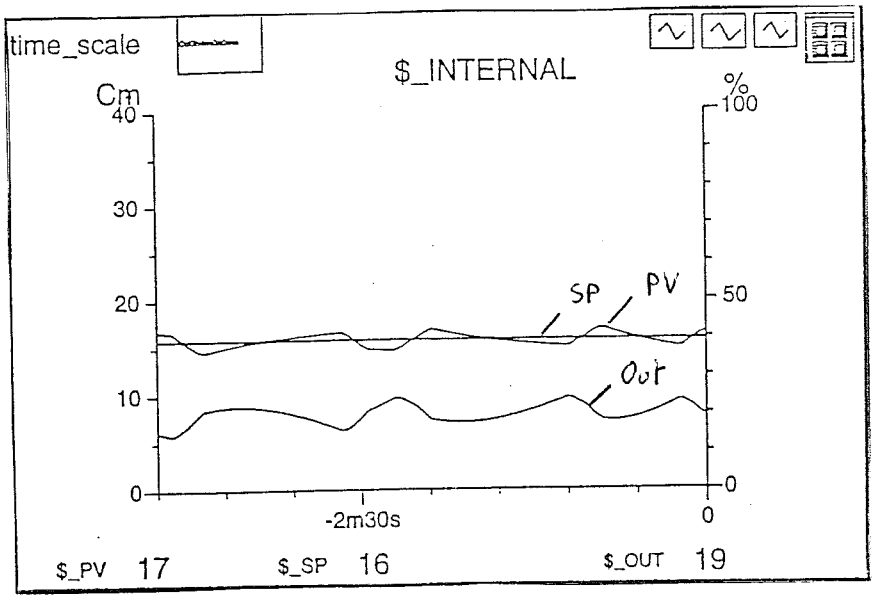
They are quite easy to eliminate if the controller has an integral part because of a high loop gain at low frequency.

I - 2 High frequency loads

High frequency loads should not appear because the process has often a low-pass character, so a filter can be used. But disturbances with a high frequency content near the ultimate frequency are too fast to be treated efficiently by the controller and they are too slow to be filtered out. Moreover the feedback can amplify these disturbances.

The oscillations can be caused also either by a bad tuned controller or by friction in the valve (our case here). The valve remains stuck then slips abruptly.

Stiction in a valve:



II - Detection of oscillations

A method for detection of bad control valves has been suggested by Tore Hägglund.

The problems for the detection are that the nature of oscillations can vary quite a lot, the frequency can be both low or middle range. The oscillations are never sinusoidal frames.

To detect oscillations, we study the magnitude of the integrated absolute error (IAE) between successive zero crossings of the control error.

$$IAE = \int_{t_{i-1}}^{t_i} |e(t)|dt$$

t_i, t_{i-1} times when e is equal to zero.

The controller has an integral part so the average control error is zero.

During good control |e| is small and $t_i - t_{i-1}$ as well. So IAE is small When a load disturbance occurs, |e| increases and $t_i - t_{i-1}$ as well. To know if there is oscillation or not, a IAE limit is defined.

Determination of IAE limit

The control error is considered to be a sine wave with amplitude a and frequency w.

$$\int |a*\sin(wt)|dt = 2*a/w$$

So IAE limit should be inferior at 2*a/w. Oscillations in the low and middle range have to be detected and about 2% peak to peak oscillations are acceptable, so IAE limit is equal to 2/wu where 2*pi*wu is the ultimate frequency. The value of wu is given by the relay autotuner. If ultimate time is not known, the integral time is used.

Tore Hägglund has taken the value 1 for a, but with this value very small load disturbances are detected. But if there is some noise, this value has to be larger.

So there is a load disturbance if IAE is larger than IAE limit.

To detected if there is some load disturbances, the frequency of load disturbances has to be large. If the number of load disturbances is larger a certain number, n_limit, during a certain period Tsup then there is some oscillations.

Choice of n_limit and Tsup

In this project n_limit is equal to 10. If n_limit detection, where every half period is detected as a load disturbance, are to be obtained during the supervision time, it is required that

$$Tsup > n\_limit * Tu / 2$$

where Tu is the ultimate oscillation period. The oscillations have a significantly longer time period than Tu. So we choose

$$Tsup = 50 * Tu.$$

Oscillations detection code

In my case I have n_limit = 10 and amplitude = 1 because I just do a simulation, so the signals are really good. (No noise,...).

III - Diagnosis [2]

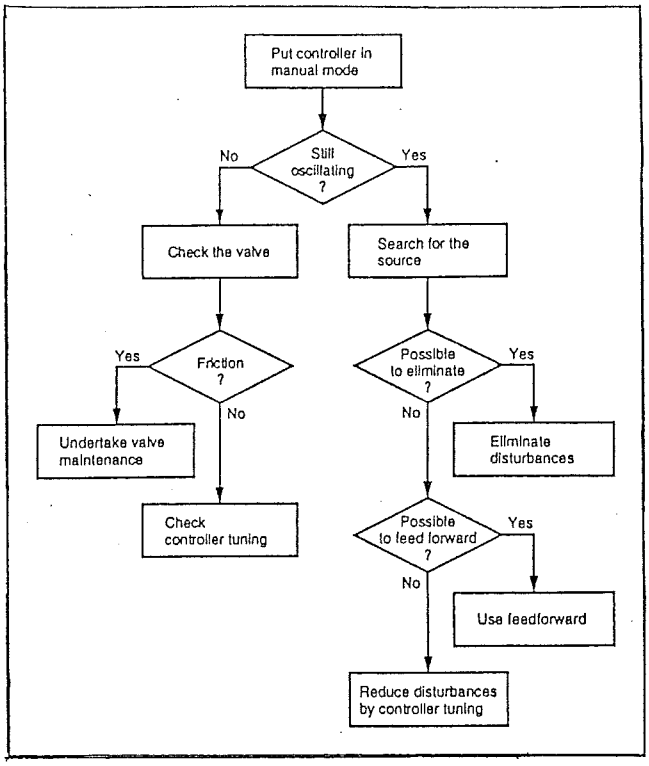
When load disturbances are detected, it remains to find out why. But in our case they come from stiction in a control valve because the model of the valve is a pneumatic valve with stiction. But if we test the code to detect load disturbances in a plant we have to know why we get load disturbances. There is a systematic way to find the cause of the load disturbances.

First we determine if the load disturbances are generated inside or outside the control loop. To know that, the controller is use in manual mode. If there are still load disturbances, the disturbances are generated outside the control loop if not they are generated inside the control loop.

If it is inside the control loop, there are two reasons for oscillations: bad controller tuning or stiction. If we make a small change in the control signal and if the measurement signal does not follow the control signal, then there is high friction or there can be a play.

If it is outside the control loop, the source of oscillations has to be found. Once the source is found, we should try to eliminate them. If it is not possible we try to feed the disturbances forward to the controller. And if it is not possible to use a feed forward, we try to reduce the effect of the disturbances by tuning the controller.

I have not used this diagnosis, method described by Tore Hägglund to find stiction because at Alfa Laval I have created a special model of the valve with stiction and in Danisco Sugar we have studied the graphs of the process value, the setpoint and the output of the controller to see any load disturbances.

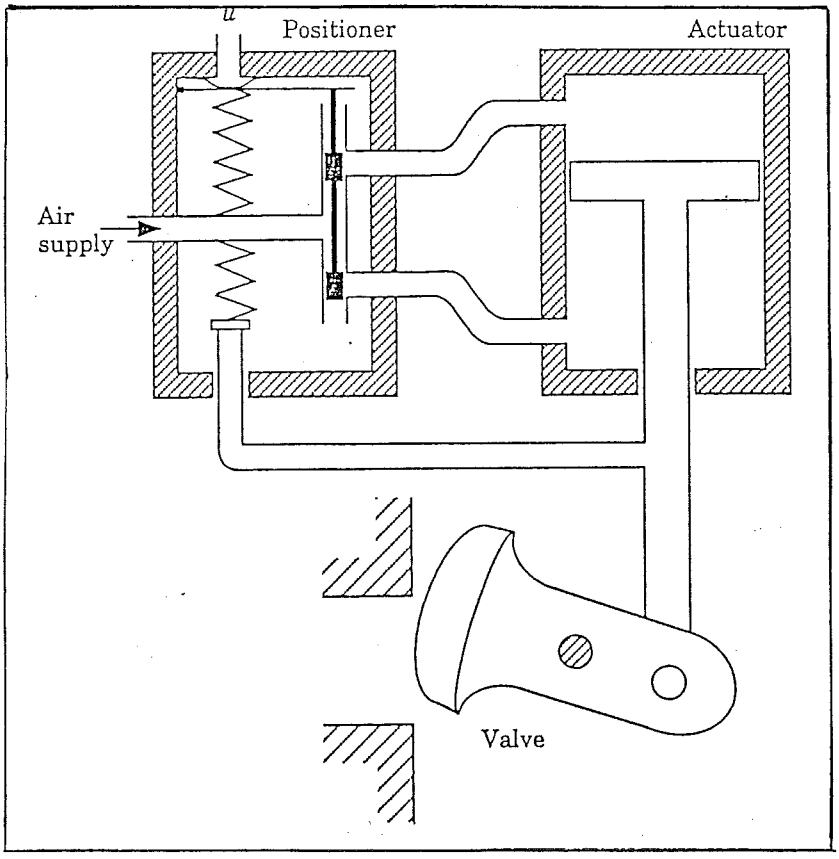


Oscillation diagnosis

IV - Control valve

A control valve consists of three main parts: - the valve
- the actuator
- the positioner

The most commonly used positioners and actuators are pneumatic, they can be hydraulic or electric as well.



IV - 1 Stiction in control valves

The most sever non linearitites in control valves are static friction and hysteresis. The problem is that stiction and hysteresis increase gradually during operation.

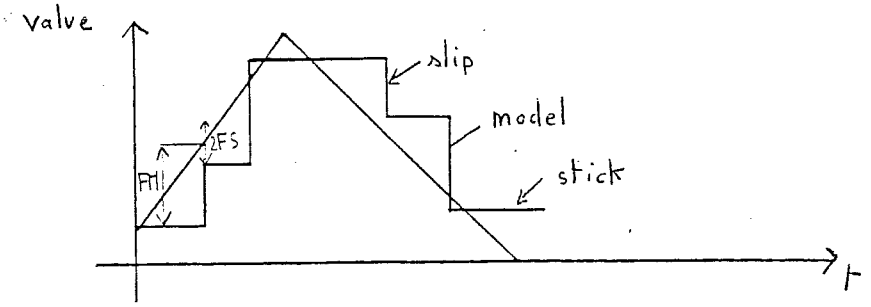
IV - 2 Model of the valve

A model of the valve with stiction has been created.

Behavior of the model

The valve keeps stick until the difference of the desired value and the real value is larger than a certain value, FM. This value can be changed on line. After the stick, the real value slides to the desired value more or less a random value between -FS and +FS. Then it get stuck until the difference is larger than FM .... See the code in Appendix

So the aim of the knocker is to obtain the difference FM earlier. So the valve moves earlier.



V - The knocker

An addition , called knocker, to a PID controller has been suggested by Tore Hägglund. [3].

The idea of the knocker is to add a sequence of pulses of equal amplitude and duration in the direction of the rate of change of the control signal to the control signal.

The basis level for the pulses will gradually change as long as the control is non zero because of the integrator in the controller. So the pressure drop over the actuator piston will increase gradually until the valve slips. When the valve slips, the difference between the measurement and the setpoint is then positive , so the pulses are reversed.

Often, just after the slip, there are several pulses of wrong sign because of delays. But it is not important because of a small energy contended in each pulse.

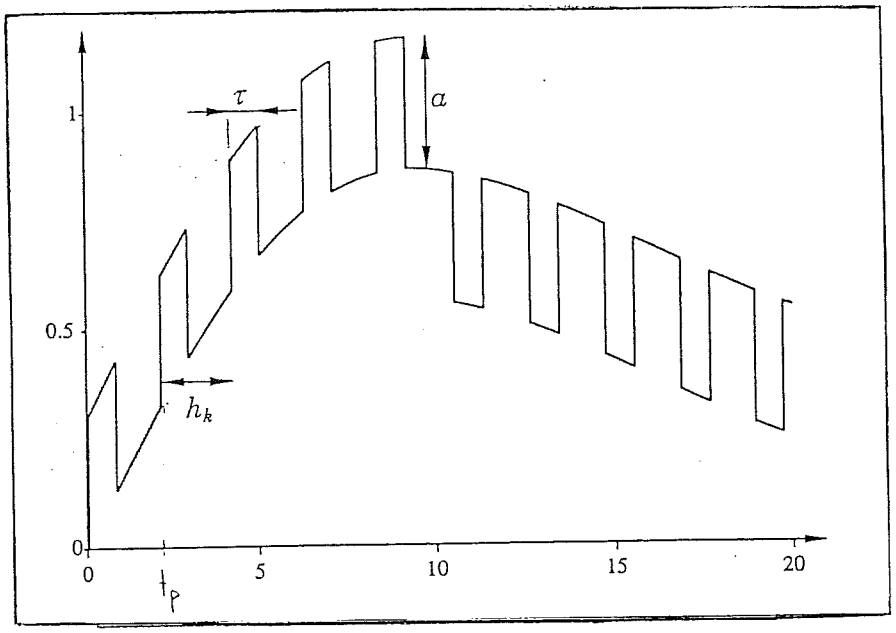
The control signal is with the knocker

$$U(t) = Uc(t) + Uk(t)$$

where Uc is the output of the controller and Uk are the pulses.

$$Uk = \begin{cases} am*sign(Uc(t) - Uc(tp)) & t <= tp + k + pe \\ 0 & t > tp + pe + k \end{cases}$$

where tp time when the previous pulse starts, k is the width of the pulses and pe is the period of the pulses.



The control signal using the knocker

Choice of parameters

The pulses should be sufficiently large, so that the valve slips at an earlier stage than without the knocker but small enough, so that they do not cause any extra slip before the flow change is observed by the controller.

The disturbances are proportional to the product am*k because

$$Y = Gp*Uk/(1 + Gp*Gc) = Gp/(1+GpGc) * (1- \exp(-s*k))*am/s = Gp/(1+GpGc) * am*k$$

The product am*k determines the energy of each pulse.

a) Pulse amplitude (am)

If the amplitude is too large, the pilot valve will open so much that an uncontrolled evacuation from the low-pressure side of the actuator occurs. So the amplitude is chosen between 1 and 4%.

b) Pulse width (k)

The width has not to be too large because it is important to not feed to much energy and the width has to be larger than the sampling period, h. In my case the sampling period is equal to 1s.

$$k > h$$

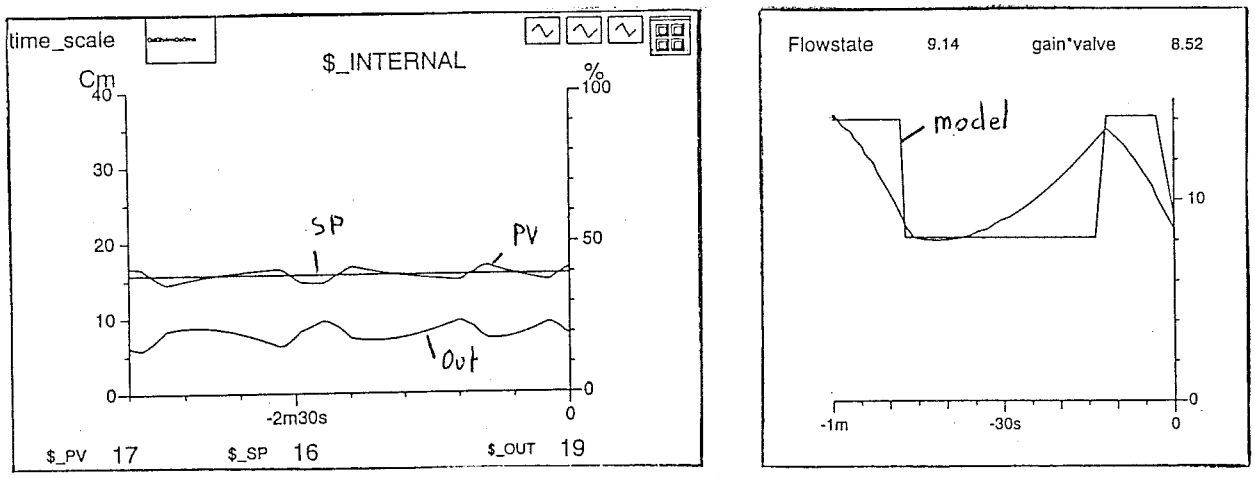
c) Period of pulses (pe)

Pe must be larger than the sampling period and the pulse width. But it must be shorter than the integral time, so that the basis level for the pulses does not change too much between successive pulses.

$$k < Pe < Ti$$

VI - Results

First the knocker is not used. Two graphs, graph 1) are obtained, one shows the stiction in the valve, how the valve behaves with the stiction and without the stiction and the other shows the set point ($_SP), the output of the process ($_PV) and the output of the controller ($_OUT).
As expected, the output of the process oscillates around the set point. These oscillations have a large amplitude and a long period. So the integrated absolute error is large, so oscillations are detected with the method described previously.



graph 1

In my case FM = 5 cm and FS = 1 cm

Then, the knocker is used. Different values for the three parameters: the amplitude of the pulses (am), the width of the pulses (k*sampling period) and the period of the pulses (pe*sampling period). The sampling period oscillates around 1 second.

VI - 1 With a PID controller

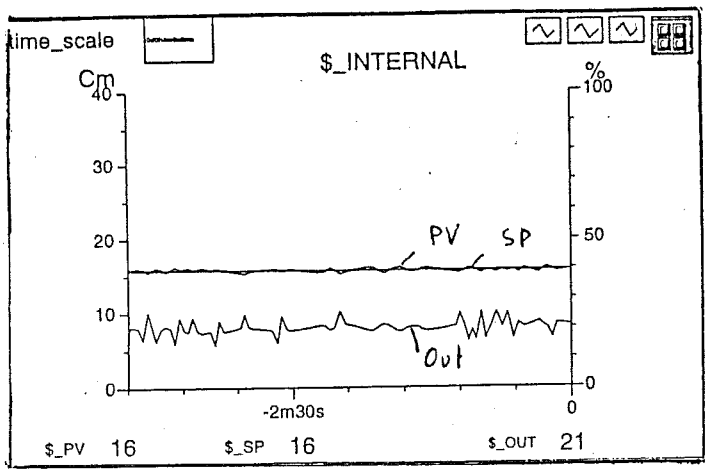
First a PID controller is used, the gain is equal to 1, the integral time to 10s and the derivative part to zero and I change the different parameters of the knocker.

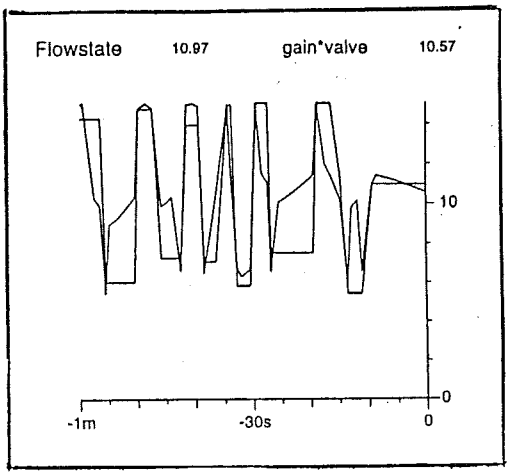
a) In this case, graph 2, the following parameters are used:

am = 4%
k = 1s
pe = 8s.

So the oscillations of the process value around the set point have a smaller amplitude and a higher frequency than without the knocker. The integrated absolute error is smaller because the period between two times when the error is equal to zero and the absolute value of the error are small. The error is the difference between the set point value and the process value.

graph 2

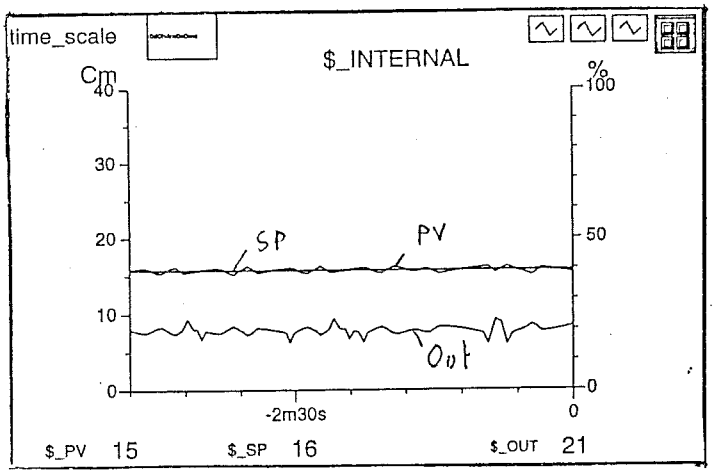




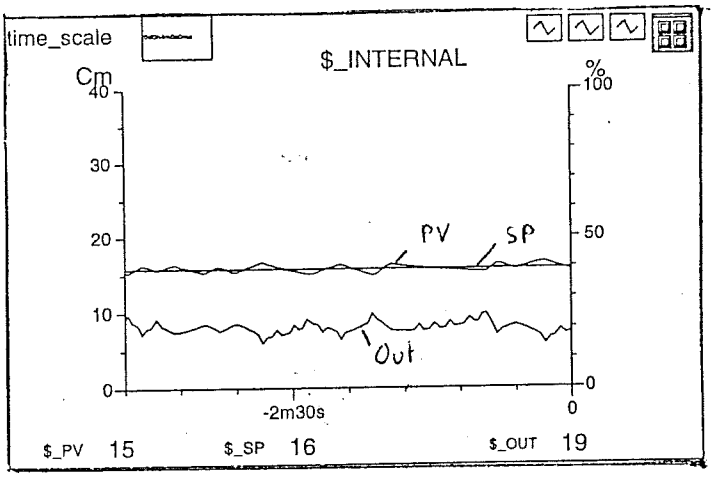
b) The parameters k and pe are not changed, but am takes different values to see what happens.

If the amplitude of the pulses decreases, equal to 3% in graph 3, at the beginning the knocker works but after a certain period it does not give enough energy. So the oscillations become less acceptable. If the amplitude of the pulse is equal to 2% as in the graph 4, the knocker does not give enough energy. This result is more visible in this case than in the case before.

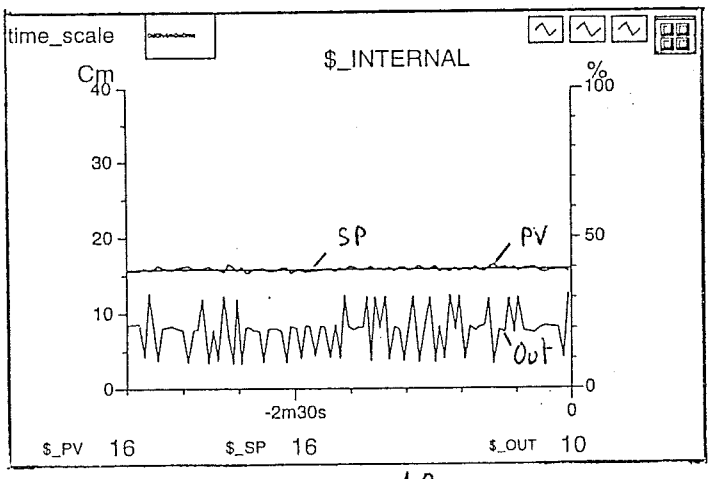
If the amplitude of the pulses increase a lot , equal to 10% in graph 5, the oscillations of the process value are very small. But the valve oscillate too fast, that can damage the valve.



graph 3 am = 3

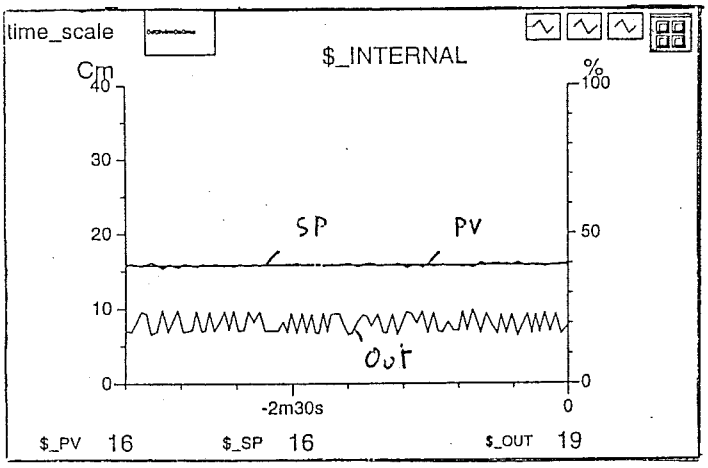


graph 4 am = 2

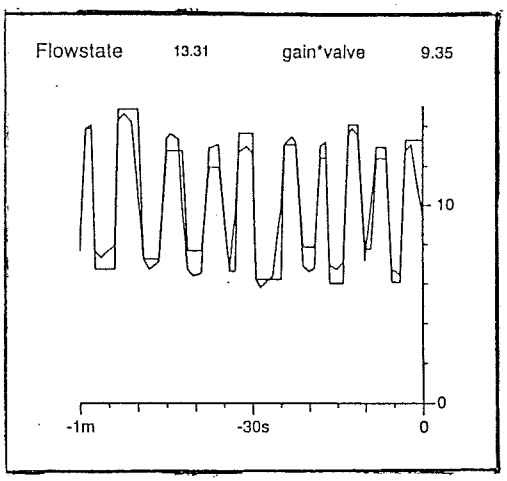


graph 5 am = 10

c) Now the amplitude and the period are not changed but the width of the pulse increases. In the graph 6, the width is equal to 6s, which is the three quarts of the period. The same result as the previous case is obtained. The oscillations are really small but the valve can be damage because of too many and too quick variations. The knocker supply enough energy to make move the valve on time, the stiction disappear.

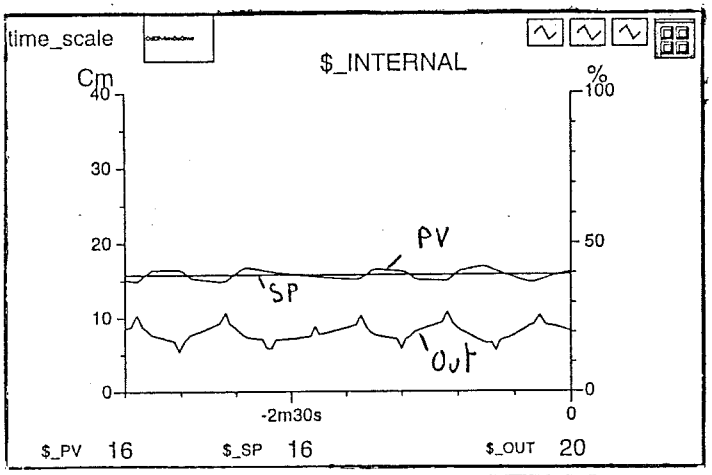


graph 6

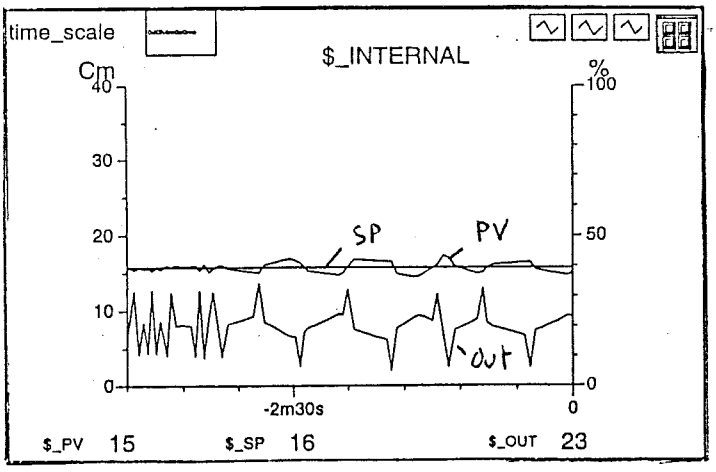


d) Then the amplitude of the pulse is equal to 3, the width to 3s and the period to 24s. There is still the same ratio between the width and the period as in the first case (1/8). The control signal will build up almost the pressure that is needed to generate a slip of the valve. When the pulse is applied, the pressure drop over the valve piston will be much larger than what is need to overcome the stiction. The result is an oscillation that is significantly larger than what is obtained during pure PI control.

If the amplitude of the pulses become equal to 10, the knocker supply more energy but it is not enough. So the valve keeps stick and some oscillation are got.



am = 3
graph 7



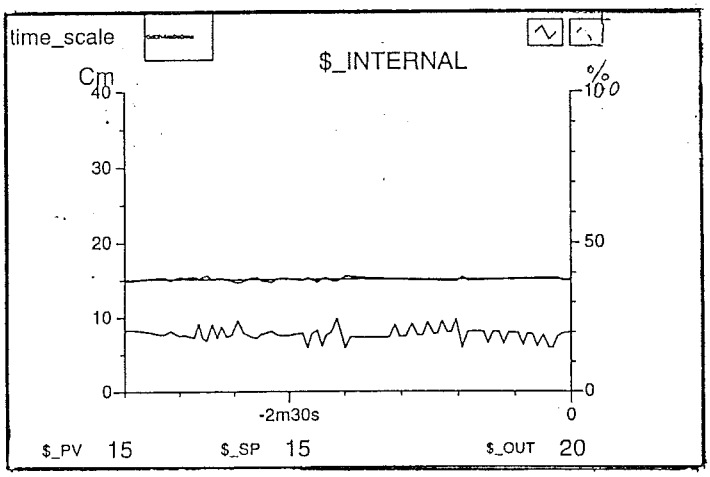
am = 10
graph 8

VI - 2 With different controllers

Now different designs of the controller are used with a knocker. The knocker has the following parameters:

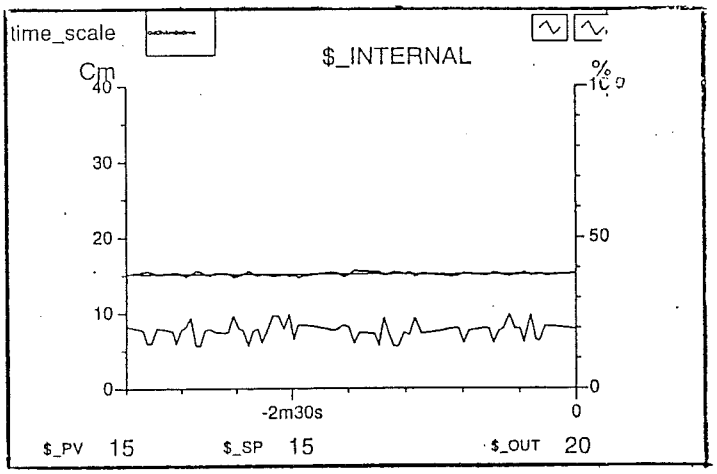
pulse amplitude = 4%
pulse width = 1s
period = 8s.

a) First we use a PID controller and the speed of the controller varies. The more the speed of the controller is high the more the frequency of oscillations is important. The output of the process, the output of the controller and the output of the valve are plot for three different speeds: slow, normal and fast. (graphs 9,10,11). The difference between these three controllers is easily visible. So when the knocker is used, it is better to use a fast controller, because the oscillations are fast.



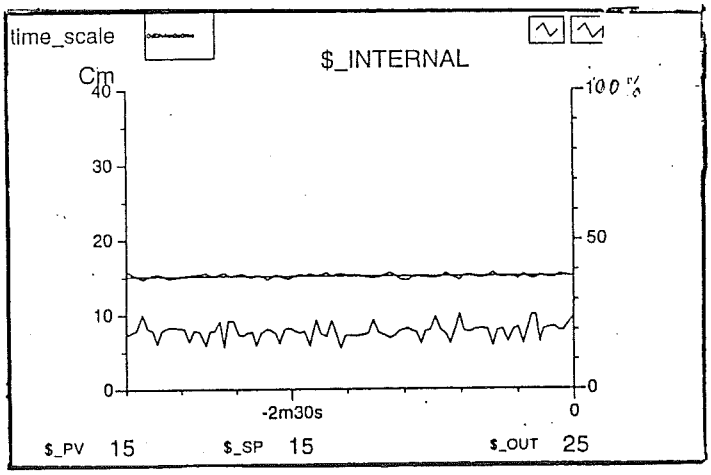
graph 9

Slow speed



graph 10

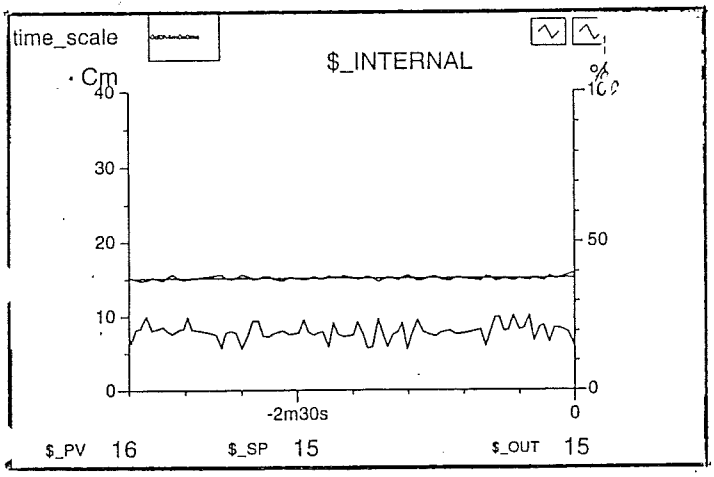
Normal speed



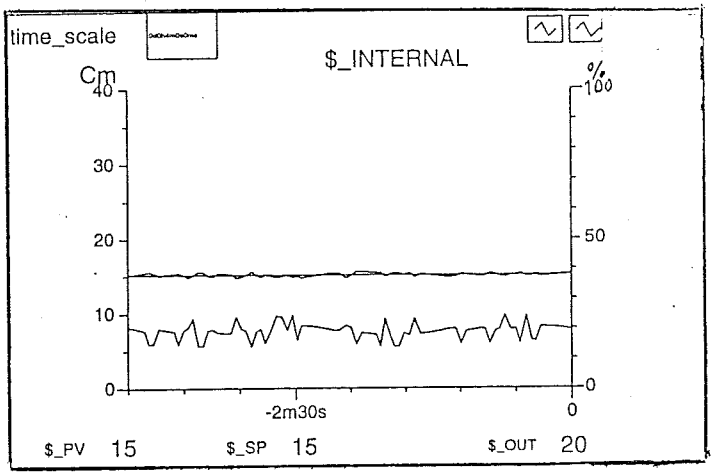
graph 11

Fast speed

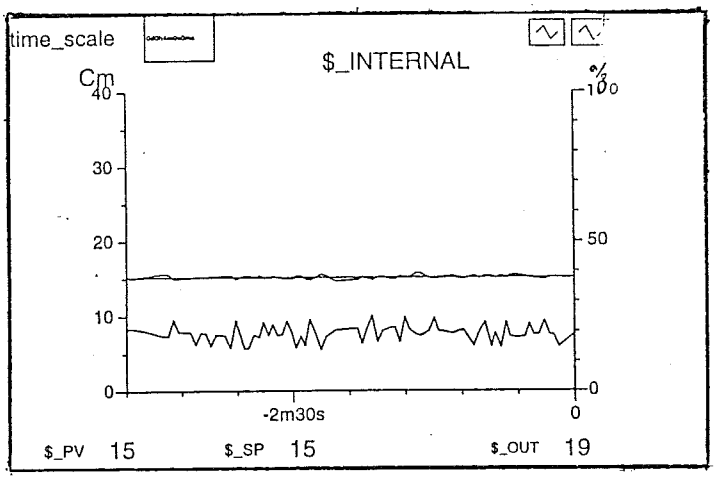
b) Secondly we use a normal speed controller, but a different controller is used: a PID, a PI and controller with deadtime. The difference between these three designs is quite small.(see graphs 12,13,14). But anyway better oscillations are got when a PID controller is used because their frequency is higher than in the other cases.



PI graph 12



PID graph 13



Dead zone graph 14

To conclude a compromise has to be found between very small oscillations of the process value with large and quick variations of the valve and small oscillations of the process value without large and quick variations of the valve. Because the valve will not support such variations during a long time, it will be broken quite rapidly. And to have a better result, a fast PID controller has to be used.

Project 2:

IMPROVEMENT OF A PID CONTROLLER IN SMOOTHING THE DIFFERENT VALUES OF THE PID.

When in a controller there is a gainscheduler, the gain, the integral time and the derivative time are different following the zone where we work. And if we pass from a zone to the next zone, the PID values change abruptly which is not always appreciated by the process and the controller is not every where as efficient as it should be. So the aim of this project is to smooth the PID values between them so that there is no abrupt jump.

In my case, the area is divided at most in four sub-areas. And the user decides which reference signal he is going to use. He can choose between the output of the process, the set point, the output of the controller or an external signal. The user defines the different limits between each zone.

Previously each zone had its own gain, its own integral time and its own derivative time. These values were kept constant for all the sub-areas. As the desired PID values were sometimes quite different from a sub-area to another, the controller did not work very good.

To have a better controller, I have done an interpolation for the gain, the integral time and the derivative time between the sub-areas, the previous area and the next area. It is a linear interpolation which has an index from zero to one. For index zero there is no interpolation at all and for index one, the interpolation is maximal.

I - Description of the interpolation:

The total area is divided in four sub-areas, the first sub-area is between zero (lim0) and lim1, the second between lim1 and lim2, the third between lim2 and lim3 and the forth between lim3 and lim4.
The interpolation has an index i, I calculate two values:

$$A_{n+1} = (lim_{n+2} - lim_{n+1})*i/2 + lim_{n+1}$$

$$B_n = (lim_n - lim_{n+1})*i/2 + lim_{n+1}$$

$$\text{for } 0 <= n <= 2$$

When the reference signal is smaller than B0, the values for the gain, the integral time and the derivative time are the current values. When the reference signal is between B0 and A1, the PID values do not kept constant until lim1 and jump to the values of the second sub-area, but they reach the values of the second sub-area following a linear function. Then when the reference signal is between

A1 and B1, the PID values are the current values of the second sub-area. And it is exactly the same for the other sub-areas.

If there are less than four sub-areas, it is exactly the same as before, but instead to have lim0, lim1, lim2, lim3, lim4 we have lim0, lim1, lim2, lim3 or lim0, lim1, lim2, lim4 or lim0, lim1, lim4 or lim0, lim1, lim4.

The linear functions for the PID parameters are:

If $B_n < Refsignal < A_{n+1}$ then

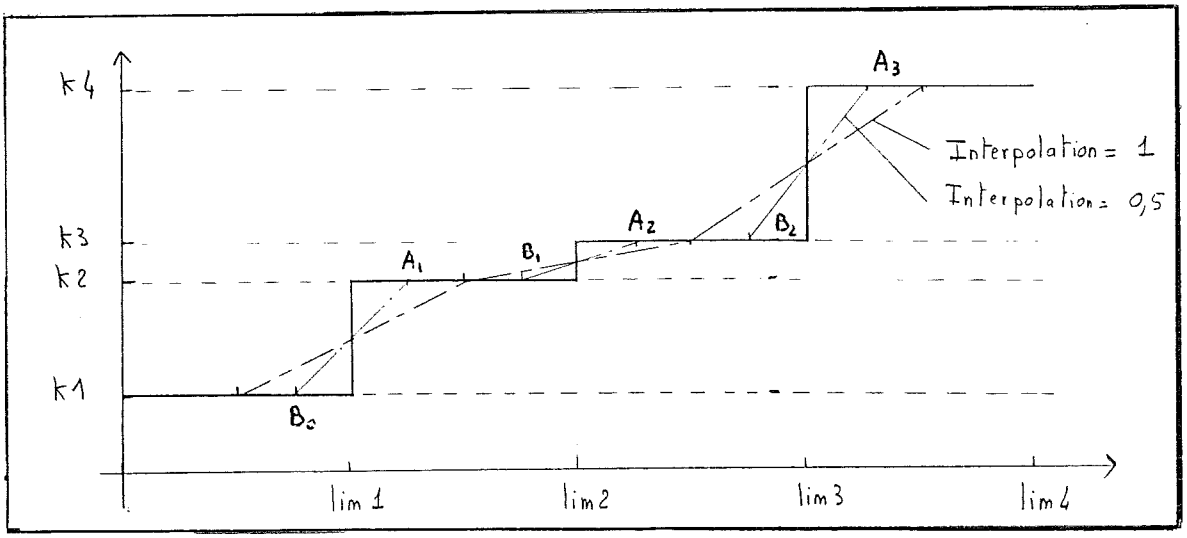
$$GainOut = ((Gain_n - Gain_{n+1}) * Refsignal + B_n*Gain_{n+1} - A_{n+1}*Gain_n) / (B_n - A_{n+1})$$

$$TiOut = ((Ti_n - Ti_{n+1}) * Refsignal + B_n*Ti_{n+1} - A_{n+1}*Ti_n) / (B_n - A_{n+1})$$

$$TdOut = ((Td_n - Td_{n+1}) * Refsignal + B_n*Td_{n+1} - A_{n+1}*Td_n) / (B_n - A_{n+1})$$

otherwise the PID parameters are constant.

Graphical representation of the interpolation for the Gain:



Remarks about the interpolation:

To see a difference between with interpolation and without interpolation, the process has to increase or decrease quite slowly. Because if it is too fast, the process does not take care about the different PID values.

More over the process has to be non linear because if it is linear we do not need a gainscheduler.

And to finish the process has to be sensible for the different PID values. We have to see the difference when the controller gets different PID values.

II - Results:

First of all I have defined my sub-area and for each area I have used the autotuner to calculate the PID values as good as possible. To get these values, I tune when I am in the middle of the zone. And I use these values for the values at

the limits. The process value is my reference value, because it shows exactly how the process behaves.

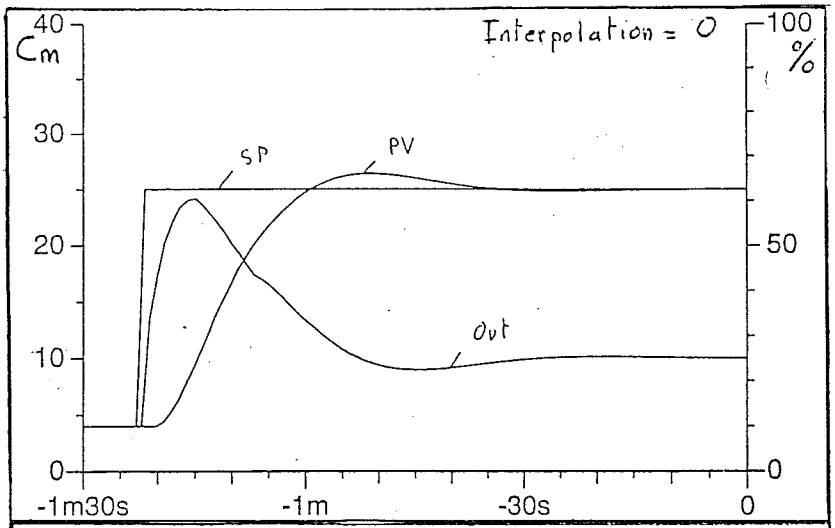
I have tested the interpolation on different simulation models.

II - 1 Normal tank and normal valve

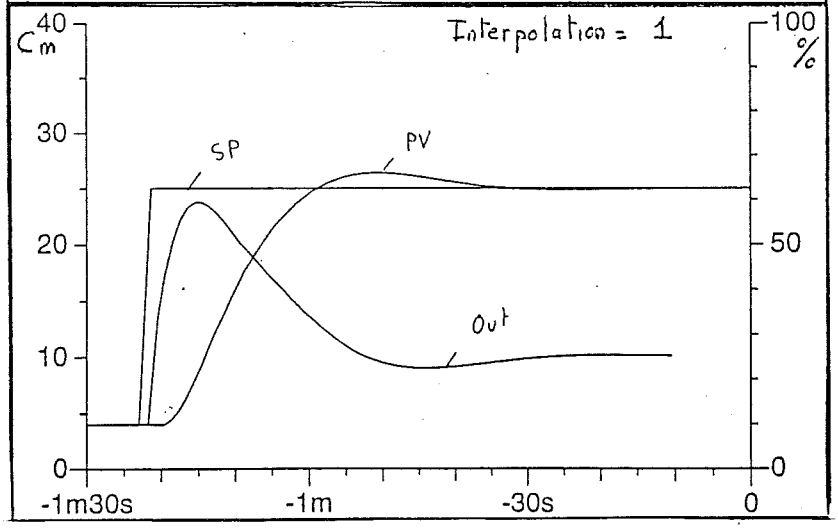
First I have tested the interpolation on the normal tank with the normal valve. The tank has a cylindrical frame of 40 cm high and 3 cm of radius. The output of the valve is proportional to the flow.

II - 1 - 1 Process value as reference value and a large jump of the SetPoint

The curves inter-1 and inter-2 show how the process behaves when there are four sub-areas and the SetPoint jumps from the first area (here SP = 4 cm) to the third sub-area (here SP = 25 cm). In this case the reference value is the Process Value, the curve inter-1 is without interpolation and the curve inter-2 is with the maximal interpolation. If I compare these two curves, there is small a difference around the delay, with the interpolation the answer is slower than without. The OutPut of the controller has a smoother answer with the interpolation than without.



Inter 1

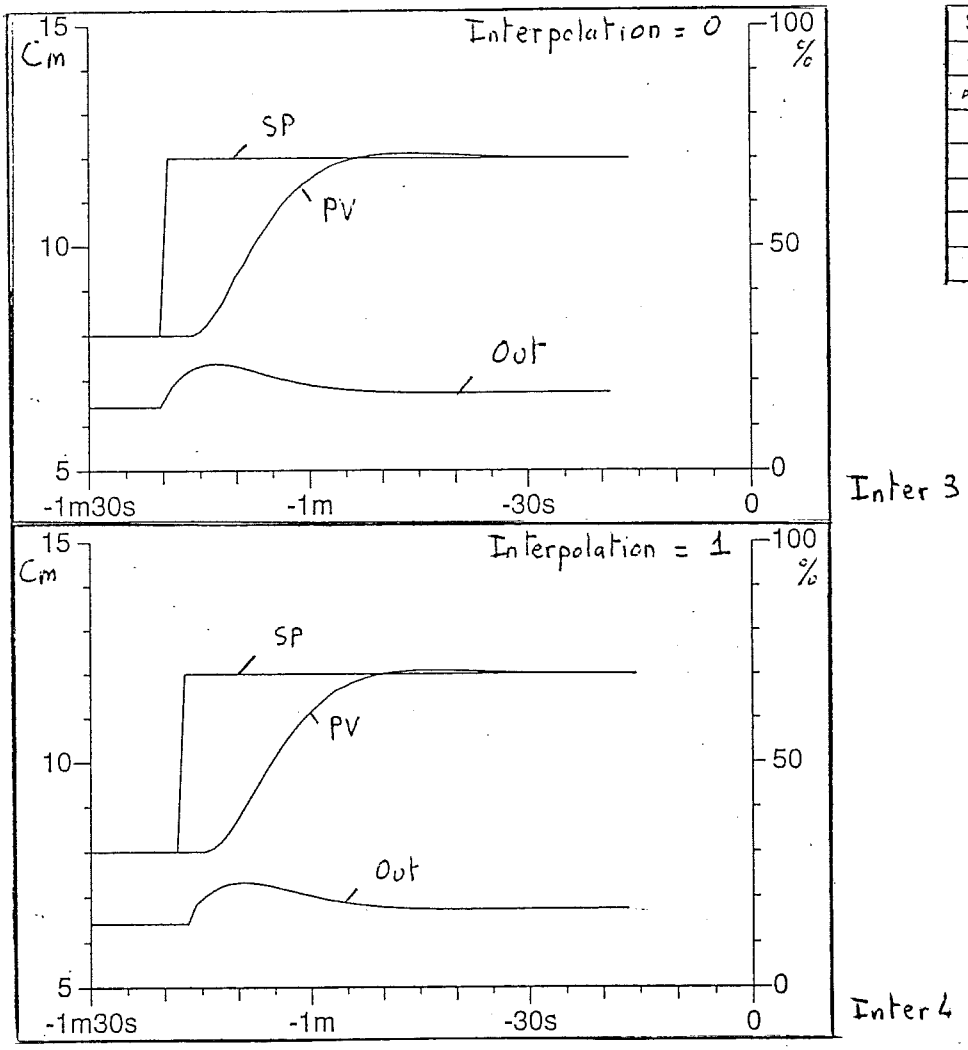


Inter 2

$_GAIN	$_TI ($_S)	$_TD ($_S)
1.62	10.2	0.00
$_LIM_3		30
1.79	7.4	1.18
$_LIM_2		20
1.81	10.2	0.00
$_LIM_1		10
2.03	8.9	0.00

II - 1 - 2 Process value as reference value and small jump of the SetPoint

Then I keep the same process but the SetPoint value jumps from 8 cm to 12 cm (from just below a limit to just over a limit). As we can see with the curve inter 3 and inter 4 there is no difference.



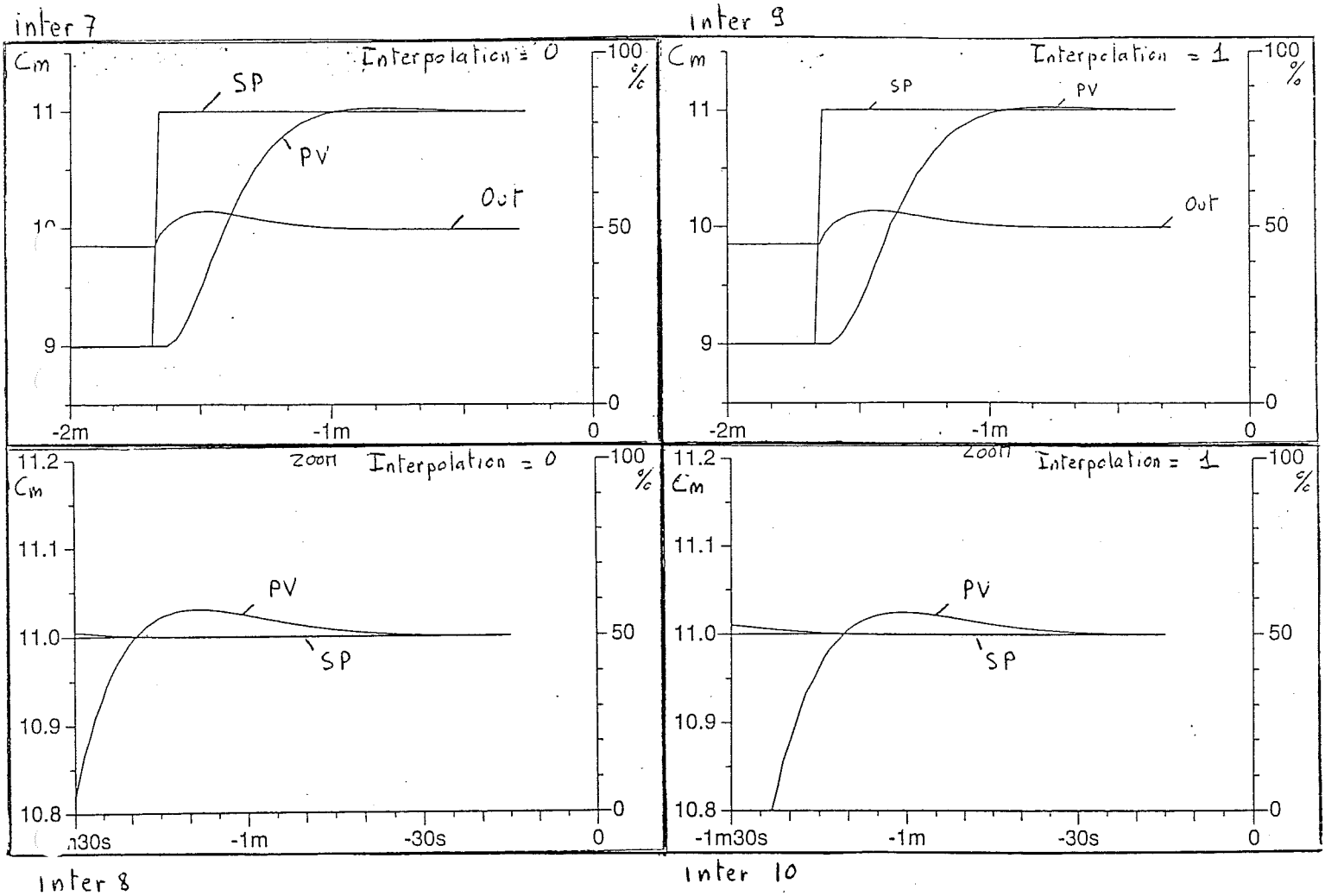
$_GAIN	$_TI ($_S)	$_TD ($_S)
1.48	12.1	0.00
$_LIM_3		30
1.80	7.4	1.18
$_LIM_2		20
1.81	10.2	0.00
$_LIM_1		10
2.03	8.9	0.00

These tests are not decisive on the effectiveness of the interpolation. But the problem comes from the small gap of the PID values between the sub-areas.

II - 2 Normal tank but narrow valve

Secondly I have tested the interpolation on the normal tank but with valve whose the flow is divided by three, the process is slower than before.

With the Process Value, the Set Point jumps from 9 cm to 11 cm, I get four curve, inter-7 and inter-8 are without interpolation and inter-9 and inter-10 are with interpolation. If I compare inter-7 and inter-9, the over shoot is smaller with the interpolation, the difference is more visible with the zoom inter-8 and inter-10.



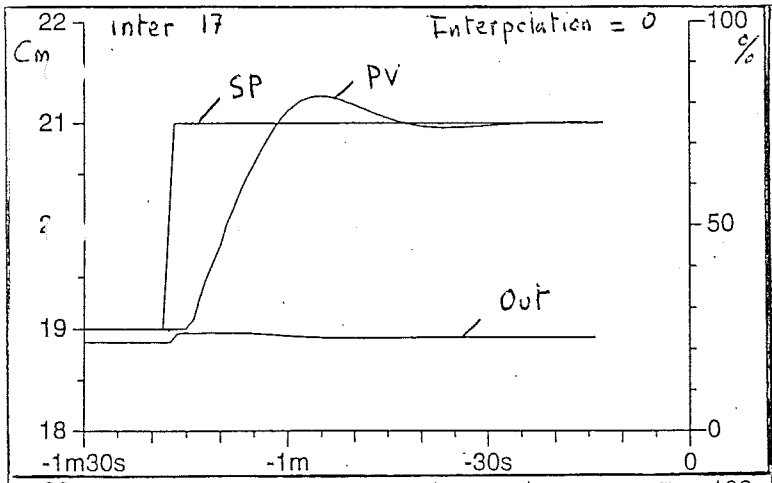
$_GAIN	$_TI ($_S)	$_TD ($_S)
1.00	10.0	0.00
$_LIM_3		30
3.95	13.4	0.00
$_LIM_2		20
3.90	14.0	0.00
$_LIM_1		10
4.33	12.7	0.00
4.07	13.50	0.00

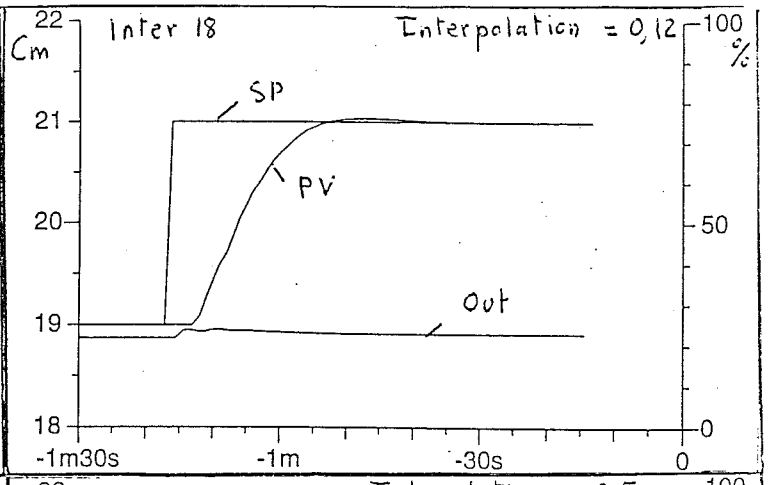
With theses tests I can conclude that the process is not very sensible to the PID values.

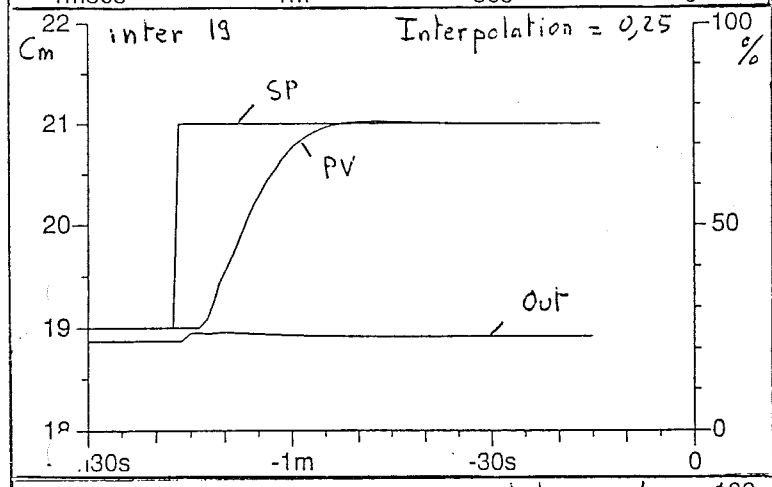
II - 3 Tank with slope sides and normal valve

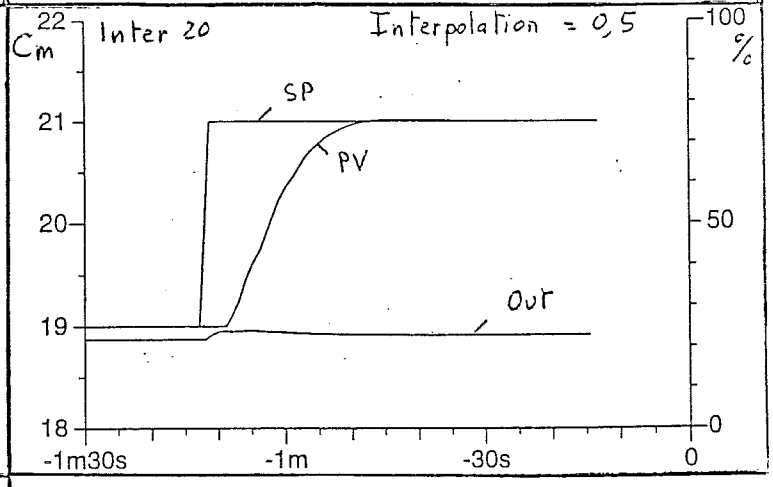
Then to get a more difficult process to control, I have used a tank with slope sides. The angle between the bottom and the sides is alpha and in my case cosinus(alpha) is equal to 0.06. The control in this case is harder specially when the liquid reaches the top of the tank because the radius is smaller.

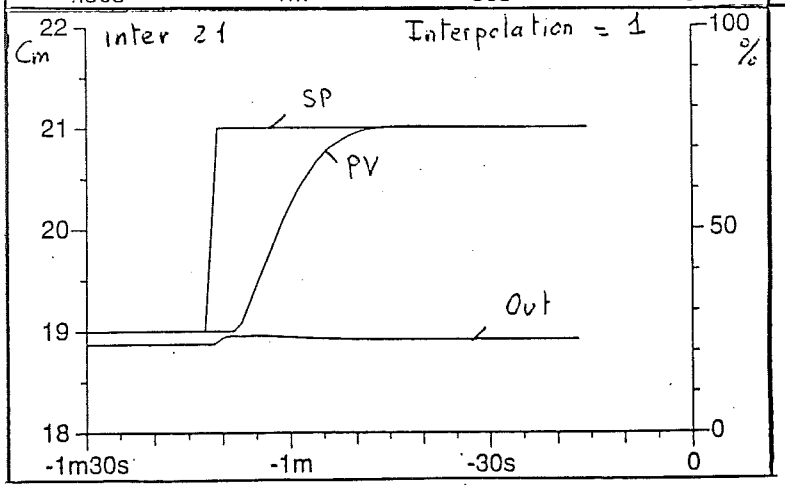
The SetPoint jumps also from 19 cm to 21 cm. In this case I get five curves: inter-17, inter-18, inter-19, inter-20, inter-21. Here as well, the difference is important between with and without interpolation. I get the same as the previous case, the process becomes stable faster but reaches the SetPoint value slower. I have tried to find the value of the interpolation to get the better result, it seems to be between 0.25 and 0.12.











$_GAIN	$_TI ($_S)	$_TD ($_S)
$_LIM_3		
$_LIM_2		
0.51	3.2	0.51
$_LIM_1		20
1.44	5.3	0.85

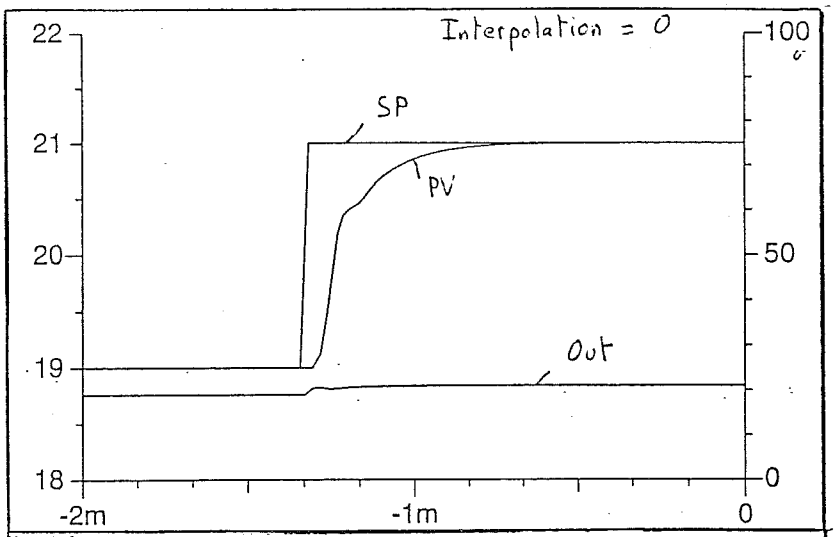
To conclude with this tank, we can see the effectiveness of the interpolation: The interpolation does not need be large to get any good effect. As the process changes continuously its PID values, the Process value becomes stable faster. This can be explained with the integral time. With a large integral time, we get a slower answer but without overshoots, and with a small integral time we get a fast answer with oscillations. In my case of the Process Value as reference value, before 20 cm the integral time is equal to 5.3 s and after 20 cm to 3.2 s. With interpolation the process does not get abruptly a small value of the integral time but reaches it progressively.

II - 4 Second order process

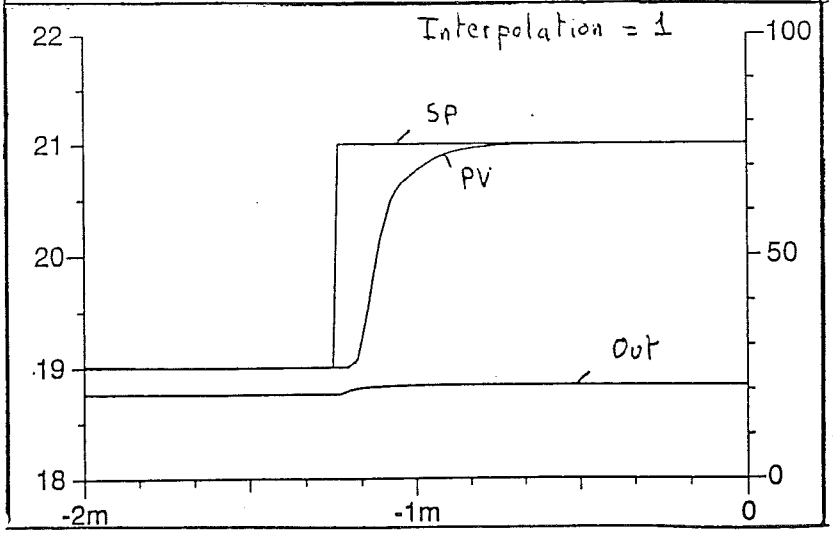
To end with my different process I have used a second order simulation model.

$$y = wo*wo / (s*s + 2*Z*wo*s + wo*wo)$$

I can change the values of wo and Z on line. I have done two tests one with interpolation and the other without interpolation. I have used the Process Value as reference value. In these two cases the SetPoint jumps from 19 cm to 21 cm, Z is equal to 0.7 and wo is 0.2*Pi rad/s. If I compare the two tests, inter-22 and inter-23, the difference is situated when the Process Value goes from 19 cm to 21 cm. The answer is smother with the interpolation. But in both cases, they reach the value 21 cm in the same time.



inter 22



inter 23

$_GAIN	$_TI ($_S)	$_TD ($_S)
$_LIM_3		
$_LIM_2		
0.93	4.3	0.69
$_LIM_1		20.0
1.63	3.8	0.00
0.43	2.17	0.38

If I change the values of Z and wo, either I cannot control the process because it is too unstable so I don't know if the interpolation is effective or I can control the process and I get quite the same results as the first case.

To conclude with the interpolation, I can say that the interpolation is effective if the process is sensible to the different PID values. When the interpolation is effective, the Process Value answer slower but becomes stable faster. I think it is because the integral time changes progressively and not abruptly.

Project 3:

IMPROVEMENT OF A PID CONTROLLER WHEN THERE ARE DIFFERENT GAIN SCHEDULES

At the Heat Production Plant in Nyköping different sets of fuel are used and all the time when they change the fuel they have to tune the PID controller for the used fuel even if they have used this fuel previously. So they wanted a program which save the PID parameter to avoid to tune every time they change the fuel.

This project proposes a faster and easier use for a PID controller when there are different gain schedules. I have done this work for SattLine. I have used an already existing controller. This controller is composed of a Gain Scheduler, a Relay and a PID controller.

Before my work, when several gain schedules were used, the tuning for the used gain schedule was done each time that the gain schedule was used. For example there were two gain schedules A and B. A was used in first, the PID controller had to be tune for A. Then B was used, the controller had to be tuned for B this time. And if A was used after, the controller had to be tuned again because the previous tuning for A could not be saved. So it was a waste of time.

With my improvement, now, the parameters which are used for each gain schedules can be saved. With this saving, a tuning is not needed if a previous tuning has be done and saved.

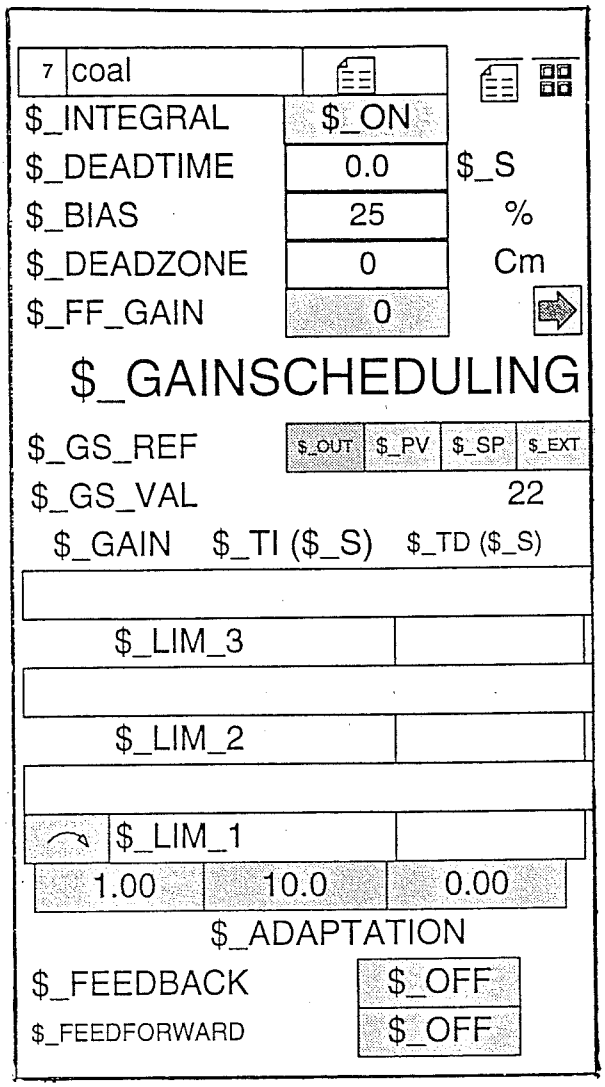
I - How this improvement has been done

From the main window of the gainscheduler, which has already existed, different possibilities are offered

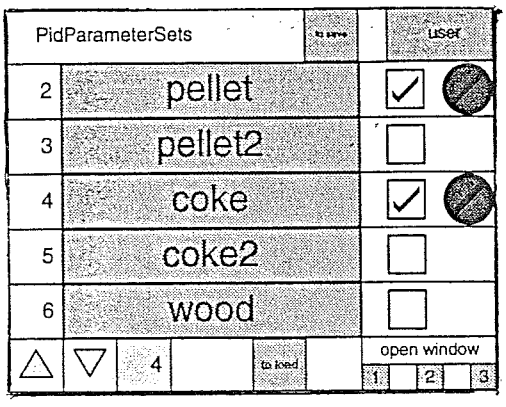
- a window (parameter set table) can be opened showing a list of the different parameters sets.
- create a new parameters set.
- change the values of a parameter set.

From the window, parameter set table, it is possible

- to save a parameter set whose the parameters are visible in the window, gainscheduler if the parameter set is not protected.
- to protect a parameter set or not protect.
- to load a parameter set to the PID controller.
- to open three windows (window1, window2, window3) showing the values of three different parameters sets. In the window, parameter set table, it is possible to show five different names of the parameters sets. But to see the others it is possible to scroll up or scroll down. It is possible to enter the address of a parameter set. Only twenty different parameters sets can be saved. But we can change this number really easly or it is possible that the user defines a limit.



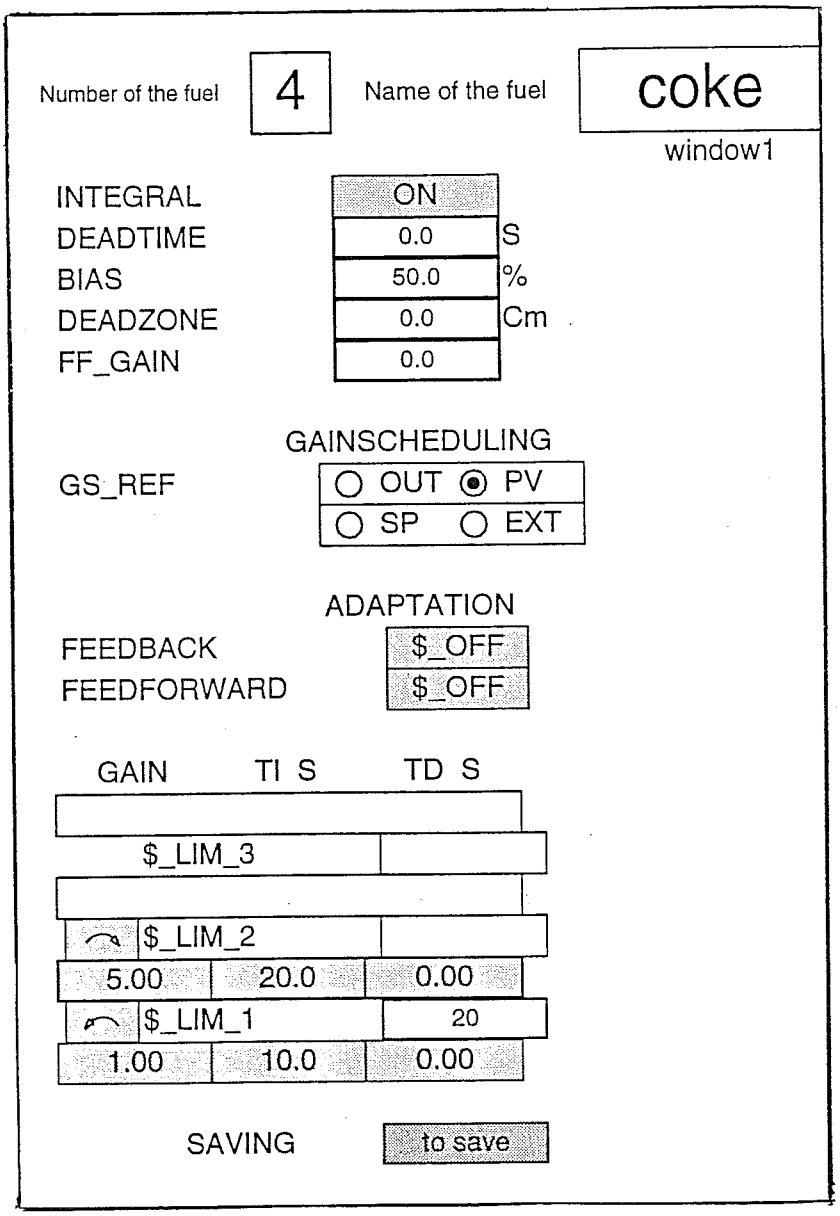
Main window of the gainscheduler



Window, parameter set table

Window1, window2, window3

In these windows it is possible to see the parameters useful for the Pid controller for a parameter set. This parameter set is not necessary that is used by the controller. It is possible to change the values of the parameter set if it is not protected. Three windows can be opened in the same time showing the pid parameters of three different parameters sets. The name of the parameter set can be non visible in the window parameter set table but the Pid parameters of this parameter set can be showed in the window.



II - Directions for use

First of all the window of the gain scheduler has to be opened. On the top of the Gainscheduler Window there a line which is connected to the parameter set table window. To open the Parameter Set Table window, the user clicks on the text icon which is situated in this line. In the Parameter Set table Window, it is possible to open three windows, Parameter Set Window, showing the PID values of three different parameters sets. To do this, the user clicks on one of the three numbers.

II - 1 Gainscheduling Window:

On the top of this window, the user enters the name and the address of the parameter set at which he wants to attribute the values which are showed in the gainscheduling window. It is possible to enter 20 different parameters sets. To save the parameters of the parameter set, the user has to go to the Parameter Set table Window and clicks on the "save" button.

II - 2 Parameter Set Table Window:

When this window is opened, the five first parameters sets appear in the list. But to see the others it is possible to scroll. To scroll the user clicks on the up arrow to scroll up and on the down arrow to scroll down. If the user knows the address of a parameter set, it is possible to enter directly the address. The user clicks on the rectangle on the bottom and enters the address.

On the right top of the Parameter Set Window, there is a button to choose which mode is going to be used. If it is the "user" mode the user can choose which parameter set is going to be used by the PID controller and if it is the "application program" mode, the user can do nothing and it is an application program which decided which parameter set is going to be used.

To protect a parameter, the user clicks on the check box which is situated in the same row as the name of the parameter set which has to be protected. When the parameter set is protected a yellow and black symbol is visible. To take away the protection, the user clicks again on the check box. When a parameter set is protected, it is not possible to save some new PID parameters neither from the gainscheduling window nor from the parameter set window.

To load a new parameter set to the controller, the user clicks on the name of the parameter which he wants to load and then he clicks on the "load" button which is situated on the bottom. The user can choose which parameter set the controller is going to use only if it is the "user" mode.

To open the Parameter Set Window, the user clicks on one of the three buttons "1", "2" or "3" which are situated on the right bottom corner.

II - 3 Parameter Set Window:

To see the values of a parameter set, the user opens one of the three Parameter Set Window. Then he enters the address of the parameter set whose he wants to see the PID parameters. He clicks on the square which is situated on the top of the window and writes the address. So the PID parameters and the name are visible.

It is possible to change some PID parameters or the name of the parameter set. The user has to click on the parameter which he wants to change and to enter a new value.

To save these modifications, the user clicks on the "save" button which is on the bottom but it is possible to save them only if the parameter set is not protected.

To conclude, with this parametersets table, the user does not need to tune the controller when he changes his parameterset.

I have done this transformation in two versions of SattLine 0.37 and 0.38. The version 0.38 is more modern than the version 0.37.

TESTS AT DANISCO SUGAR COMPANY

At Alfa Laval, my codes improve the control of valves but it is just simulation. There is no unknown factors, everything is known in advance. So I have tested my code, the knocker and the interpolation, in a real plant with some real problems to know if they improve the controller as well.

I have tested my codes at Danisco Sugar, which is a Danish company located in the suburbs of Malmö. This company produces all kind of sugar.

First we had to find a valve which does not work correctly. We have found one which is used to feed a centrifuge. Then we have studied the behavior of the valve to know if there is statistic friction. So now we have a valve which sticks and slips. So I can test my knocker on it.

Moreover, the operators told us that when they change the quality of the sugar, more or less thick, the controller does not always work correctly. So we can test my interpolation code on this valve and the reference value is linked to the thickness of the sugar.

I - The knocker

We have filtered the process value because there was a lot of noise. The time constant of this first order filter is 5s. And the sampling period equals to 0.4s.

I - 1 Load disturbances detection

First I have checked if there were some load disturbances. I have used the code described in the first project. But I have had to change the values of the integrated absolute error sup, the n_limit and Tsup. N_limit is the minimal number of oscillations during Tsup to detect oscillations. I have had to change these values because of the noise. The code detected oscillations not because of stick and slip but because of the noise. I have increased the value of IAE limit, instead to have IAE limit equals to 2/wu where wu is the ultimate frequency divided by 2*pi, IAE equals to 6/wu. And as the system is quite slow, instead to have Tsup equals to 5*n_limit*Tu, where Tu is the ultimate period, Tsup equals to 7*n_limit*Tsup. All these values have been defined by experiments. To choose the value of Tsup, I have taken the smaller possible value because if Tsup is too large some oscillations are detected which are not oscillations in fact.

With these values for the oscillations detection code, I have detected oscillations.

I - 2 Oscillations compensator

As I know there are load disturbances which are caused by the stick and slip of the valve, I use the knocker to check if it improves anything on the behavior of the valve.

First I have tuned the knocker. I have to choose good values for the amplitude, the width of the pulses and the period between each pulse. As I have learned with the first project the amplitude has to be small and the period has to be smaller than the integral time.

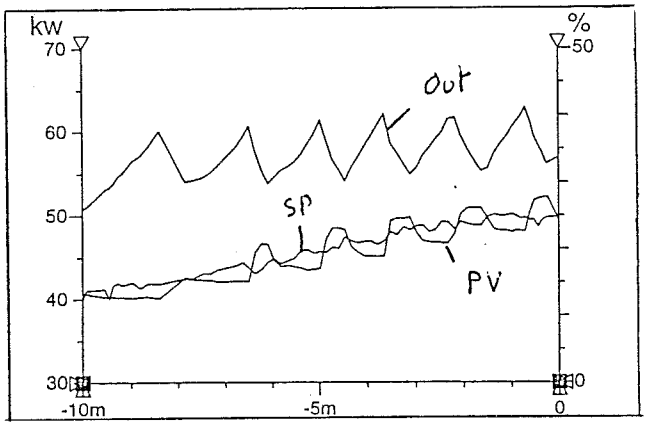
The values of the PID controller are:

Gain = 1
Integral Time = 10s
Derivative Time = 0s

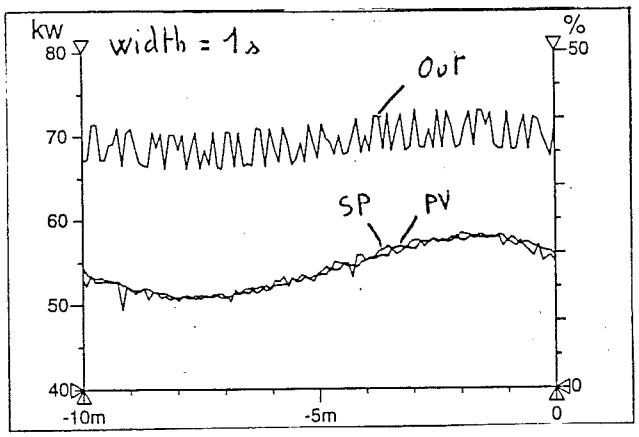
The values for the knocker are:

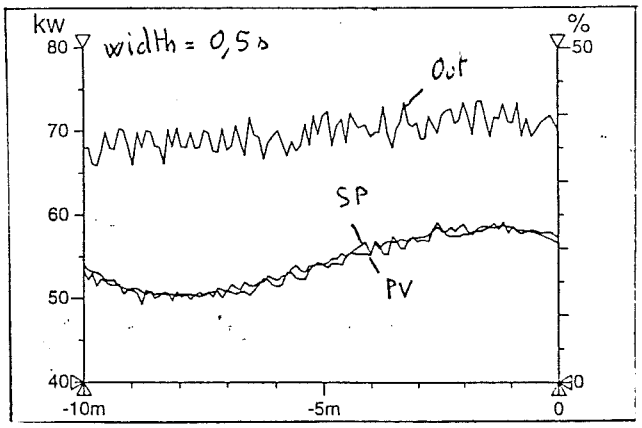
Amplitude = 2.5
Width = 1s
Period = 1.5s

With these values for the knocker, the valve does not stick and slip, it moves continuously. But perhaps with these values the valve moves too much, that can damage the valve. So I have tried a smaller value for the width, Width = 0.5s, and in this case, the valve slips and sticks a bit.



Without the knocker





With the knocker

To conclude, the knocker works on a real case, but it is more difficult to tune it. Sometimes, the knocker works perfectly and sometimes it does not. It depends on the behavior of the valve. But generally speaking the knocker improves the control of the valve. The difference is bigger between with and without the knocker when the power increases than when the power decreases. It is harder to increase the power than to decrease, so we get more stiction when the power increases.

II - The interpolation

As I said before, the control is not good if the sugar becomes thicker. So the controller need to have different PID parameters following the thickness of the sugar.

The reference value for the interpolation depends on the thickness of the sugar. If the valve is opened with a certain value, the centrifuge needs more power when the sugar is thick than when the sugar is fine. So to get a relation showing the thickness of the sugar, we calculate the ration between the Output of the controller (the opening) and the Process Value (the power).

ratio = A*(OutPut + B)/(PV + C)

In my case A = 100, B = 0 and C = 0.01.

II - 1 The tuning

First of all, I ask to the operators to change the thickness of the sugar, I write down the value of the ratio and I try to tune manually the controller. I do this for different values of the ratio.

During the tuning and the tests, the knocker is on, its values as the same as previously.

It is quite hard to find good values for the controller, because the SetPoint changes all the time. We cannot fix it as in the simulations.

As I have learned with my second project - the interpolation, I need two different gainschedules, and the process oscillates around a certain value.

In my case, I have divided my area in four sub-areas:

- ratio smaller than 45.
- ratio between 45 and 60.
- ratio between 60 and 85.
- ratio between 85 and 100.

The ration between the Process Value and the OutPut of the controller oscillates around 60. So I have tried to find good PID values for the sub-area 45-60 and the sub-area 60-85.

For sub-area 45-60: Gain = 2.19.
Integral time = 15s.
Derivative time = 0s.

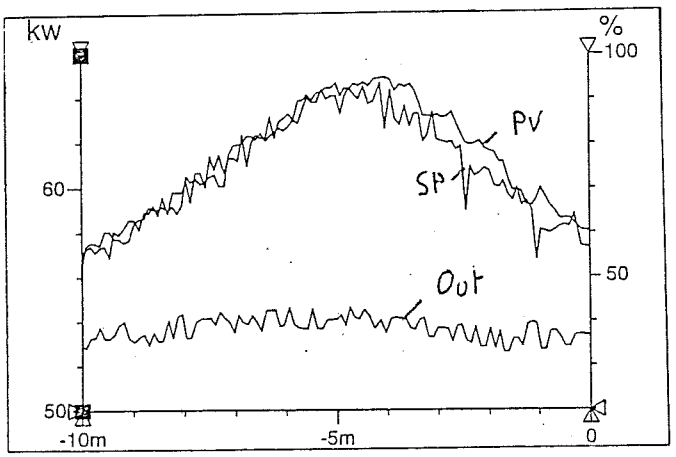
For sub-area 60-85: Gain = 0.94.
Integral time = 7.4s.
Derivative time = 0s.

The values between these two sub-areas are quite different. We are in a good case to see anything.

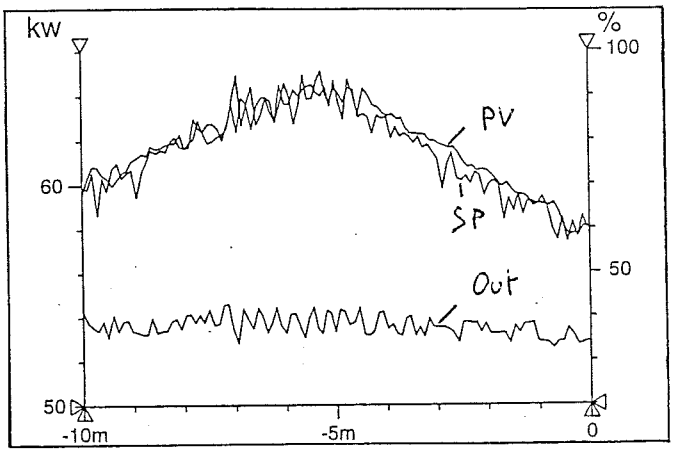
II - 2 The results

To see a difference between with the knocker and without the knocker, we study the process when the power increases. On the other hand, to see the effectiveness of the interpolation, we study the process when the power decreases because there is less stiction.

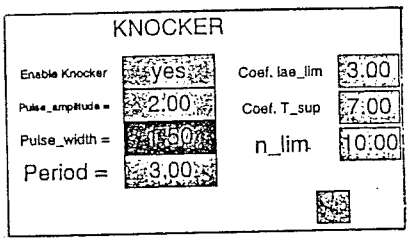
I get two graphs one with interpolation and the other without interpolation. The difference between theses two graphs is not big. But we can see that with the interpolation, the Process Value does not oscillate. But it is valid only when the power decreases.



interpolation = 0



Interpolation = 1



To conclude with these tests, the effectiveness of the interpolation is not really proved. The reason is that there is too much stiction in the valve. I have done some tests with the interpolation but without the knocker, the interpolation does not help the controller, we see no difference between with the interpolation and without the interpolation, both cases are bad.

These tests of the knocker and the interpolation are showed that the simulation is quite far from the reality. But anyway, the knocker improves really the controller and the interpolation works if the stiction is not really important.

To improve the control when the sugar becomes thicker, we have to tune the knocker. So instead to have a table for the PID parameters, I think it will be good to have a table for knocker's parameters (width and period). Because I had to tune the knocker following the ratio.

CONCLUSION

My work at Alfa Laval Automation was very interesting. In four months I have learned how to use three different versions of SattLine, which has not always been easy specially when I started a new version, but I could ask some help to every body. At the beginning I had some problems with the connections. And the project which consisted to create a PID parameterset table has taught me a lot on how to use SattLine. It was the project that I found the hardest to realize because at the beginning I did not know a lot about the procedures and so I did not know how to start. The two others have been easier to realize but they have been more interesting because I had to analyze the results, try to improve them.

So my master thesis work in a Swedish company has permitted to me to know a work mentality different of the one in France. If I compare the two, I think in Sweden they are less stressed but we can see this not only in the work but in the life of every day. More over they do not argue each other. They do not usually stay after 5 pm to work. If they have to do extra-work, they like better to come the week-end.

In the way of working, If I compare it with the French, they have more meetings here than in France to give their opinion, to say what is wrong. Even every two weeks, an important person inform all the workers on one or several points: the financial results, security, etc. during the lunch. Another thing which is different of the France is that Alfa Laval wishes the birthdays. Once a month for the employees who have their birthday in the month have a meeting where they can say their opinion and they get a piece of cake. In Sweden the opinion of all the employees is important and it is perhaps for this reason there are less demonstrations than in France.

Because I am French, I have helped to do the French translation. But it was far to be easy because either I have never learnt the French expression or I have forgotten the French Translation of some expressions. Moreover these expressions without their context mean nothing so I have had to think in which context they are used.

ACKNOWLEDGMENTS

I want to thank you Karl Åström who has helped me to find this Master Thesis, Tore Hägglund for his help.

I want to thank all the employees of the R&D department because even if I am not a Swedish speaking people, they speak to me, they help me when I had some problems with Sattline or the computer in general. I want specially to thank Lars Pernebo, my supervisor, Finn Brodersen, who has solved all my problems with SattLine, Ingemar Hansson who helped me when I was hanging up my computer and Helene Tagesson, who drove me to Alfa Laval.

REFERENCES

1. Adaptive Control - second edition - Karl J.Åström & Björn Wittenmark Addison Wesley 55866.

2. "A control-loop performance monitor" - Tore Hägglund. Control Engineering Practice, **3**, pp.1543-1551, 1995.

3. The knocker, a compensator for stiction in control valves - Tore Hägglund Internal report, Department of Automatic Control, Lund Institude of Technology, Novembre 1995.

4. SattLine Reference Manual - part I & II

```
IF nostiction THEN
   FlowState = Gain*Valve;
ELSE
   t = t + 1;
   b = b + abs(sortie:New - Gain*Valve);
   IF abs(sortie:New - Gain*Valve) < FM THEN
      sortie:New = sortie:Old;
      t1 = t1 + 1;
   ELSE
      sortie:New = Gain*Valve + FS*sin(2*3.14159*t/10);
      b = 0;
      t1 = 0;
   ENDIF;
   FlowState = sortie:New;
   A = Gain*Valve;
ENDIF;
TempState = Temperature;
```