INNOVATIONS IN ADVANCED REGULATORY CONTROL METHODS FOR MODERN DISTRIBUTED CONTROL SYSTEMS

by

Gustaf Zacharias Gous

Submitted in partial fulfillment of the requirements for the degree
Philosophiae Doctor (Electronic Engineering)

in the

Department of Electrical, Electronic and Computer Engineering
Faculty of Engineering, Built Environment and Information Technology

UNIVERSITY OF PRETORIA

September 2024

SUMMARY

INNOVATIONS IN ADVANCED REGULATORY CONTROL METHODS FOR MODERN DISTRIBUTED CONTROL SYSTEMS

by

Gustaf Zacharias Gous

Promoter:	Prof Ian K Craig
Co-promotor:	Prof J Derik le Roux
Department:	Electrical, Electronic and Computer Engineering
University:	University of Pretoria
Degree:	Philosophiae Doctor (Electronic Engineering)
Keywords:	Averaging Level Control, Advanced Regulatory Control, Ramp Horizon Control, Integral Gap Control, Simplified Optimal Averaging Level Control

Many modern Distributed Control Systems (DCS) in industry are new replacements of previous versions of the same DCS vendor's product line. During such upgrades the process is often automated using software to translate existing controller configurations as well as custom software to comply with the new system's requirements and syntax. Doing this makes the upgrade process much faster and reduces the risk of introducing errors. It does, however, rob the control practitioner from making use of new features and capabilities of the new system. Therefore, there are many DCS in industry where only a small fraction of their newer capabilities are used. Many improvements in advanced regulatory control (ARC) that would improve control performance are available, but are never used.

In order to show how modern DCS can enable more complex control solutions, four ARC level controllers and two stiction compensation algorithms, all more complex than current solutions typically found in industry are introduced as examples of how increased complexity may provide increased control performance.

ARC can for example be used to stabilize the flow out of storage vessels, which is referred to as averaging level control. Averaging level control techniques can be grouped in three groups, Proportional-Integral-Derivative (PID) based controllers, ARC and Model Predictive Controllers (MPC).

Four new ARC methods for averaging level control are proposed that can be implemented on a standard DCS. They minimise movement of the controller output (u) while decreasing the risk of the level (y) going outside of the desired range, offering the strengths of both DCS-based controllers and MPC. These controllers are tested using simulated levels to compare the ARC methods and how they minimize changes in u while preventing y from exceeding limits.

Two ARCs were compared on an industrial plant to a previously deployed non-linear Proportional-Integral (PI) controller. Plant tests showed that the standard deviation of u could be decreased by the ARC to less than 10% of what could be achieved by the original non-linear PI controller. This was done while keeping the level within the prescribed limits while the previous non-linear PI controller did allow the level to go outside the limits. Decreasing the variability in flowrate of the feed stream enabled improved pressure and temperature control, leading to improved product quality and decreased utility and energy consumption.

The performance of an ARC is compared to MPC in a 24-hour simulation using typical plant data as a disturbance. The performance metrics show that the performance of the ARC and MPC was similar. The MPC is however more costly to deploy, and requires additional hardware, software and communication protocols.

Sticking valves tend to cause cycles in control systems used in industry, degrading product quality and yield. Many attempts have been made to alleviate the impact of stiction. Mechanical knockers are used with success to knock loose the sticking components. Most other stiction compensation methods attempt to find ways to move the control output by an amount greater than the stiction band, while still getting the valve position as close as possible to the desired position. Instead of overcoming stiction and getting the valve position to the control output, the valve can be moved such that over time, the valve is on average at the correct position, while still moving the valve in increments that are larger than the stiction band.

Two new ARC methods for stiction compensation are introduced that use standard modern DCS

functionality without the need of additional hardware and software. The two stiction compensation methods are simulated and the results compared to the original system with and without stiction. The simulations show how time-based stiction compensation can bring the controller response back to where it resembles the original system without stiction, while time-based stiction compensation with move suppression is able to keep the controlled variable within a reasonable distance from the setpoint while minimising movement of the valve. Here it is also clear that the ability of the stiction compensation algorithms to minimise control error and/or minimise valve movement is worth the added complexity.

LIST OF ABBREVIATIONS

ARC	Advanced Regulatory Control
CV	Controlled Variable
DCS	Distributed Control Systems
DMC	Dynamic Matrix Control
FIC	Flow Indicator Controller
IGC	Integral Gap Control
LIC	Level Indicator Controller
MPC	Model Predictive Control
MV	Manipulated Variable
OPC	Open Platform Communications
PI	Proportional-Integral
PID	Proportional-Integral-Derivative
PLC	Programmable Logic Controller
RHC	Ramp Horizon Control
RHIGC	Ramp Horizon Integral Gap Control
ROC	Rate Of Change
SOALC	Simplified Optimal Averaging Level Control
TV	Total Variance

LIST OF SYMBOLS

d	Maximum deviation from setpoint
e	Control error
ΔF_{ST}	Change in flow during step test
F	Range of flow
f	Maximum flow deviation
$K_{c(gap)}$	Controller gain inside gap
k	Current execution cycle
k'	Process slope gain
K_c	Controller gain
$K_{c(PI)}$	Default or starting tuning value for controller gain for a PI averaging level controller
K_g	SOALC gain
K_p	Process gain
K_r	Controller gain ratio
N	Number of data samples
N_C	Control horizon (minutes)
N_P	Prediction horizon (minutes)
P	Tuning constant for exponential filter
Q_R	Penalty on error
Q_S	Slack variable weight
R	Input move size weight
σ	Standard deviation
S	Slack variable
SB	Stiction Band
t_{CI}	Stiction compensation interval
t_{Hi}	Time interval at high value
T_{RH}	Ramp horizon
τ	Time constant for first order system
τ_f	Multiplicative factor used for IGC tuning
τ_i	Integral time constant for a PID or PI controller

$\tau_{i(in\ gap)}$	Integral time constant for use when y is inside the IGC's gap
$\tau_{i(outside\ gap)}$	Integral time constant for use when y is outside of the IGC's gap
$\tau_{i(PI)}$	Default or starting tuning value for integral time constant for a PI averaging level controller
t_s	Interval between data samples or execution cycles
$\bar{u}$	Average valve position during current interval
u	Controller output
u'	Valve position
u_{RH}	Calculated change in control output for RHC
u_{Lo}	Time interval at low value
V	Drum volume
$\bar{y}$	High limit
$\underline{y}$	Low limit
y	Level measurement
y_{high}	Highest value of y in a dataset
y_{low}	Lowest value of y in a dataset
y^{SP}	Setpoint for level

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 INTRODUCTION

1.1 PROBLEM STATEMENT

Modern Distributed Control Systems (DCS) can be used to apply advanced regulatory control (ARC) techniques that can improve control system performance without additional cost associated with additional hardware and software. This advantage is often overlooked when DCS are upgraded to the latest version and the previous control system is transferred with as little changes as possible, essentially wasting the additional capabilities of the new system.

1.1.1 Averaging level control

Most liquid processing plants have drums or tanks between different parts of the process that are used to provide a steady flow of material to the downstream section, should upstream upsets cause a feed disturbance. The steady feed flow provided by a feed tank or feed drum improves downstream stability, improving product quality and yields as well as minimising utility and energy consumption.

A steady feed flow improves downstream stability, improving quality, yields and minimising energy and utility consumption. Maximizing the use of buffer capacity afforded by available storage is called averaging level control and requires the control engineer to let the level move away from the average or desired position in order to minimize the variance in the material flow towards the downstream section.

If this is not done carefully, the level may go too low or too high, leading to process conditions like pump cavitation or liquid carry-over. To prevent this, high and low alarms or trips are implemented to protect equipment and ensure product quality. In order to not trigger alarms or trips, the control engineer will attempt to tune the averaging level controller to stay between a high and low limit, conservatively set between the alarm and/or trip limits.

Existing averaging level control methods are either implemented on a DCS or Programmable Logic Controllers (PLC) or are implemented on servers linked to the DCS or PLC. Implementing the controllers on the DCS or PLC limits functionality but allows faster execution cycles. Using server-based control is more cumbersome as it requires additional engineering, additional hardware and software with communication protocols, but it enables more accurate control, providing improved averaging level control while simultaneously more successfully keeping levels between limits.

Level control differs fundamentally from control of self-regulating variables as it is the most common form of a non-self-regulating process in industry (King, 2016). It will not reach a new equilibrium after a process upset as the mass balance is not conserved, leading to the level either increasing or decreasing indefinitely. The only way to restore balance is when another disturbance occurs, or a controller move is made to restore the mass balance by setting the accumulation to zero.

If an integrator is defined to be at steady state when it is at a constant rate of change, and u is increased by δu, the controlled variable (CV) y will change dynamically and reach steady state, not at a new absolute value, but at a new rate of change, which will be the product of the process gain (K_p) and the change in u, or $K_p \times \delta u$ higher than the previous rate of change. This is why an integrating variable is defined to be at steady state when the rate of change of the level is constant, and not when the measurement value itself stays constant.

This tendency of levels to continue changing in value after an upset or control action has a big impact on the control systems applied. Should the control engineer be too eager to stabilize u, the risk of breaching the limits is increased.

When applying averaging level control on a plant in order to stabilise process flows, the control engineer is involved in a balancing act. Feed drums or tanks are part of a plant specifically to allow the control system to let the level vary in order to provide a smoother feed flow into the plant. Many engineers also use vessels not designed specifically for this purpose for averaging level control, like reflux drums or column bottom sumps on distillation columns. If the engineer is too conservative in tuning the averaging level controller, the buffer capacity is wasted. A feed drum running with tight level control may as well not be there, wasting the investment to provide the drum.

However, if the engineer is too ambitious in tuning the averaging level controller, process problems

may develop. If the level is allowed to go too low, cavitation may result in the pump taking material from the drum. If a column bottom sump is used as buffer capacity, the thermosyphon effect driving the heat exchange in the reboiler may be impacted. If levels are allowed to go too high, issues like liquid carry-over into vapour lines may result, which can damage downstream equipment. For this reason, levels used in averaging level control will typically have associated alarms or trips that will manifest should levels go too high or too low. If this happens frequently, operations will lose confidence in the averaging level control scheme and will force the control engineer to revert to a more conservative tuning set.

Existing averaging level controllers can be divided into model predictive controllers (MPCs) and Proportional Integral Derivative (PID) based controllers. MPCs react to the absolute value of the level as well as the predicted rate of change of the level and therefore are well equipped to ensure that y does not go outside of the set limits. However, they require additional hardware and software as well as more engineering to implement. They run slower than DCS-based controllers.

PID based controllers are faster and cheaper to implement and can run at much faster execution cycles, which improves feedback control. They are weaker in ensuring the y does not go over the set limits, mostly because the tuning of PID controllers is based on the assumed largest expected disturbance. This forces the control engineer to either set limits or tuning conservatively, limiting the performance of the averaging level controller.

Controllers that are able to run natively on a standard DCS, with minimal engineering, that are able to keep levels between set limits would be able to improve the performance of averaging level control, leading to increased stability in liquid processing plants.

PID based averaging level control methods are typically implemented on a standard DCS or a PLC. They are quick to implement and tune as they use standard functionality of the control system. They typically attempt to move the manipulated variable (MV) u as little as possible while keeping the level process variable (y) close to a setpoint (y^{SP}).

MPCs are typically implemented on servers linked to a DCS or PLC. They require more engineering to design and implement and they require additional hardware and software with communication protocols to the DCS or PLC. They often use high ($\bar{y}$) and low limits ($\underline{y}$) for y rather than a single

setpoint. They are more successful than PID based controllers at keeping levels from going over desired limits.

1.1.2 Stiction compensation

Stiction, short for static friction (Huba et al., 2011), refers to when an object withstands a force that is applied due to static friction that must be overcome to set the object in motion from a stationary position. It is the resistance to the start of movement between two surfaces that are in contact with each other. In process control it usually applies to a valve stem that will not start moving when the control system starts making changes to the output of the controller (u) that is sent to the valve.

Because of the valve stiction, u will not always be the same as the actual valve position (u'). The control system will keep moving u in the same direction until the change in u is large enough to overcome the static friction and the valve moves to a new position. Stiction is usually due to either a rough valve stem or the packing that prevents process material from escaping past the valve stem.

1.1.3 Research gap

Often control engineers in industry do not use a fraction of the functionality afforded by their DCS, due to the upgrade methodology and time constraints. Making use of advances in DCS will enable improved control systems which will improve control performance.

To illustrate this, four ARC methods for averaging level control are introduced, that outperform existing controllers while only using modern DCS functionality. In addition, two ARC methods for stiction compensation are introduced, using only DCS functionality.

1.2 RESEARCH OBJECTIVE AND QUESTIONS

The objective of this thesis is to design ARCs that can run on a state-of-the-art DCS system, without the use of additional hardware and software, at a fast execution cycle, that are able to combine the strengths of existing DCS-based and server-based techniques. These controllers should be able to:

- Run on the DCS, using native functionality.
- Run at a fast execution cycle, at 1 second or less.
- Be able to keep y between the set limits.
- Minimize variability in u.

The controller functionality is in response to the following research questions:

- Is it possible to simplify model based averaging level control techniques to the extent that they can run natively on a standard DCS?
- Can stiction compensation using only DCS functionality improve control performance?
- Will simplifying model based control techniques negatively affect the control performance?
- Can control engineers in industry improve control performance by making more use of existing functionality of a modern DCS?

1.3 APPROACH

A literature study was performed which confirmed the existence of the identified research gap. A simulated level was created on a Honeywell Experion DCS and the simulation was used to compare existing averaging level controllers from literature. The simulation was also used to develop and test several new averaging level control techniques.

The simulation was started with the process at steady state after which it was subjected to different types of disturbances as well as different magnitudes of disturbances.

Data was collected and analysed using several averaging level performance metrics from literature as well as a new performance metric developed as part of this research. The new averaging level control techniques were also evaluated using the same metrics in order to compare their success with the existing methods. The more successful new techniques were then tested on a process plant in industry and compared with the existing controller that they replaced.

Another simulation was developed to compare the new techniques with MPC. Actual plant disturbance data were used to perturb the simulated process. The same metrics were used to compare the performance of the new control techniques with MPC.

Two stiction compensation methods were developed and tested using a process simulator that can simulate valve stiction. The simulation tested if the methods could reduce control error when stiction is present, or if they would be able to reduce valve movement. The second method was also implemented and tested in industry.

1.4 RESEARCH GOALS

The goal of the research is to show that ARC embedded in a standard DCS is able to:

- outperform traditional PID based averaging level control methods,
- perform as well as model based techniques when used for averaging level control, and
- outperform traditional methods of stiction compensation.

1.5 RESEARCH CONTRIBUTION

The research contribution is four ARC systems for averaging level control that can execute using standard DCS functionality without the need for additional hardware and software while outperforming traditional averaging level control methods, both in terms of preventing y from going over limits, as well as minimising movement in u.

This research also contributes two ARC systems used for stiction compensation, that can be used to either keep y closer to y^{SP} when stiction is present or allow y to drift farther from y^{SP} in order to minimise movement in u.

The objective is not to introduce complexity for its own sake, the added complexity must be justified by improvements in control performance. Therefore the performance of the new methods are measured against existing methods using simulations. Some of the methods that showed performance improvements over existing methods in simulations were deployed on typical process plants and the performance thereof compared and discussed.

1.6 RESEARCH OUTPUTS

The following publications resulted from this study:

- Gous, G. Z., Wiid, A. J., le Roux, J. D. and Craig, I. K. (2023). Advanced regulatory control techniques for improved averaging level control performance, *Industrial & Engineering Chemistry Research* **62**(38): 15578–15587.
- Gous, G.Z., Le Roux, J.D., Craig, I.K, (In Press) Time Based Stiction Compensation, *IFACPapersOnline*.

1.7 OVERVIEW OF STUDY

In order to establish a baseline regarding the complexity of ARC, Chapter 2 details a literature study that was done to determine the current state of ARC. Current practises regarding averaging level

control, how it is done, how the performance of different methods are measured and compared, are investigated. Current stiction compensation methods are also discussed.

Chapter 3 discusses new methods that should improve process control performance. Four newly developed ARCs that are used for averaging level control are presented. These are the Integral Gap controller (IGC), Ramp Horizon Controller (RHC), Simplified Optimal Averaging Level Controller (SOALC) and a combination of the RHC and the IGC, called the Ramp Horizon/Integral Gap Controller (RHIGC). Secondly, two stiction compensation methods, time-based stiction compensation and time-based stiction compensation with move suppression are presented. All these new methods can run on a modern DCS without need for additional hardware or software.

In Chapter 4 the new averaging level controllers are deployed on a modern DCS and compared to a typical PI controller by using a simulated level that also runs on the DCS. The stiction compensation methods are also deployed on a DCS and are connected via Open Platform Communications (OPC) to a bespoke process simulator that can simulate valve stiction. The mechanism of the methods is shown and the results of the simulations are discussed. The performance of RHIGC is compared to an MPC using a 24-hour real time simulation, using plant data as a disturbance.

In Chapter 5 the application of RHC and RHIGC on an industrial process is discussed and their performance compared to the original PI controller. The time-based stiction compensation with move suppression algorithm is tested on a tank level in industry. The tank level controller was subjected to 7% valve stiction. The results are compared to previous attempts at stiction compensation as well as the original situation where there was no compensation.

Chapter 6 discusses the conclusion of this thesis regarding the potential existing in industry for making better use of the capabilities of modern DCS.

CHAPTER 2 LITERATURE STUDY

2.1 CHAPTER OVERVIEW

In this chapter a literature study is discussed to show the current state-of-the-art in process control, specifically ARC. A short history of process control is given to show how it initially developed. This sets the background for the discussion of current ARC methods used in industry. Current averaging level control methods are discussed, as well as stiction compensation to set the background for the new methods discussed in Chapter 3.

2.2 PROPORTIONAL-INTEGRAL-DERIVATIVE CONTROL

The most widely used process control algorithm in the petrochemical, mining and other processing industries is PID. These algorithms are deployed using industrial computer systems, such as DCS or PLC. DCS are more widely used in controlling continuous processes and is therefore favoured by the petrochemical industry. PLCs are aimed at controlling time-based or event based sequences that are more suited to batch processing, such as found in the mining and minerals industry. Since the 1980's these systems have been developed such that DCS are now very capable at programming and running sequences, while modern PLC have very capable PID algorithms as standard functionality.

A historically well-known use of proportional action at the start of the industrial era is when James Watt, a Scottish inventor created a governor to control the speed of a steam engine (Bennett, 1996). It used flyballs to mechanically apply proportional control to open or close a steam valve to regulate the speed of a steam engine. As the engine speed increases, the centrifugal force drives the flyballs away from a central spindle. As the flyballs move out, a mechanical arm closes the steam valve proportionally to the error in the speed.

PID has been widely used for process control in industry since the early 1900's, with the first pneumatic

PID controllers from 1920 onwards. Since then improvements and refinements were made such as:

- applying tuning rules,
- normalising the range of the controller input and output,
- using valve position control to ensure that intended changes in u are applied as calculated, and
- filtering the derivative contribution to u to minimise the impact of noise in y.

Yet despite a century of technological advancements, the core principles of the PID control loop have remained unchanged, solidifying its status as the industrial backbone of control systems.

A typically digital PID representation used in industry is the non-interactive velocity form (King, 2016; Seborg et al., 2016; Corripio, 1982):

$$\begin{aligned} \Delta u_k &= u_k - u_{k-1}, \\ &= K_c \left[e_k - e_{k-1} + \frac{t_s}{\tau_i} e_k + \frac{\tau_D}{t_s} \left(e_k - 2e_{k-1} + e_{k-2} \right) \right], \end{aligned} \tag{2.1}$$

where Δu_k is the change in manipulated variable, K_c is the controller gain, a tuning variable, e_k is the error between y and y^{SP} at the current execution cycle (k), e_{k-1} is the error at the previous execution cycle, e_{k-2} is the error two execution cycles ago, t_s is the execution cycle time of the DCS or PLC, τ_i is the integral tuning constant and τ_D is the derivative tuning constant. A PID controller for the level in a vessel will calculate movements in u, which is typically a valve opening or the required flowrate, either flowing into or going out of the vessel under consideration. The controller implements these moves to prevent y from moving away from y^{SP} or to bring y back to y^{SP}.

2.3 ADVANCED REGULATORY CONTROL

ARC includes many different techniques that can be applied in a modern DCS using standard functionality. Different opinions exist regarding what techniques are included in ARC and therefore the most reasonable approach is to include all techniques that are more complex than a basic single-input single-output (SISO) feedback PID control loop (Skogestad, 2023).

2.3.1 Cascade control

Cascade control is a widely used control strategy deployed in industrial processes to improve control performance by employing multiple control loops in a hierarchical structure (Seborg et al., 2016). It uses nested control loops, where the output of one controller regulates the setpoint of another controller as shown in Figure 2.1. The primary purpose of a cascade control system is to improve the overall

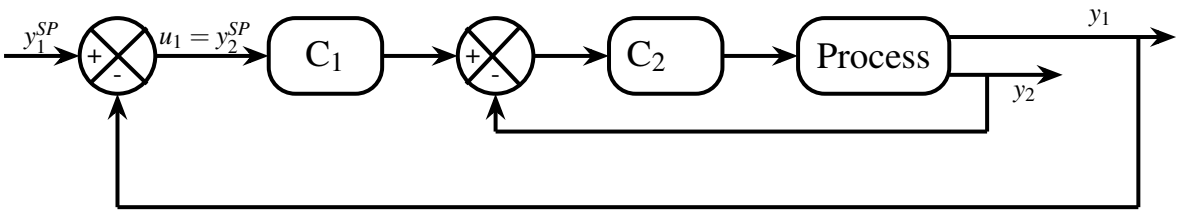

Figure 2.1. Cascade control.

performance and stability of a process by addressing certain challenges associated with single-loop control. Typically, the u of an outer loop is used as a setpoint for an inner loop. Cascade control is often used to address non-linear process responses (Skogestad, 2023). The process response of the outer loop must be significantly slower than the inner loop to prevent instability. The outer loop is also known as the master loop and the inner loop as the slave loop.

A typical example in industry is a reactor or distillation column temperature that is controlled by manipulating a steam valve (Wolff and Skogestad, 1996). This has several advantages. Firstly, it compensates for the non-linear steam flow to valve opening response by creating a fast inner loop, where the speed of the flow response is used to overcome the non-linear valve behaviour. The response of the much slower temperature controller to the steam flow is approximately linear, enabling the slower outer loop to improve its control response. If the temperature controller directly manipulates the steam valve, the slow temperature dynamics will cause significant deviations caused by the non-linear response of the steam valve.

Secondly, steam headers in industry usually have several major consumers that cause sharp demand changes (Majanne, 2005; Dzedzemane et al., 2018). When this happens, the steam supply pressure is upset and will cause severe temperature deviations in a SISO control loop. When cascade control is used, the much faster steam flow to valve opening response will negate steam pressure upsets much quicker, minimising the impact on the outer temperature control loop.

2.3.2 Ratio control

Ratio control is used in industrial processes to maintain a specific ratio between two or more process variables. Typically a manipulated variable such as a valve or flow setpoint is adjusted to set the flow of one stream as a ratio to another. Feedback control may then be used to ensure the ratio setpoint is reached. Ratio control is often used to reject flow disturbances in process streams (Skogestad and

Morari, 1987; Seborg et al., 2016).This is useful in processes where the relationship between different components needs to be maintained for optimal performance or product quality.

A typical example of ratio control are reagents that are fed in stoichiometric ratio to a reactor (Wang and Wong, 2006). The obvious advantage here is that a ratio controller will be better at keeping to the stoichiometric ratios between reagents fed to a reactor at the desired value, than an operator changing setpoints to several flow controllers. By keeping the ratio between the reagents at setpoint, operator intervention is reduced, the possibility of operator error is reduced, improved stoichiometric ratios can minimise reagent or chemical consumption and can improve product quality.

Another example of ratio control is when energy sources that are kept at a ratio to a process flow that varies significantly (Luyben, 2022). By using a ratio controller to immediately increase or decrease fuel or energy sources when the flow of a process stream that is being heated changes, the flow upset is minimised when compared to a temperature controller that will have to bring the temperature back to setpoint after the impact of the flow change becomes evident in the temperature. The benefits here are improved temperature control that can lead to improved product quality and yield, as well as decreased energy use.

Another often used example of ratio control is chemical dosing (Miller, 2018), deploying systems that ratio chemicals with process flows. Keeping the ratio between the chemicals and the process flows constant, rather than just using a flow controller again reduces operator intervention, minimises the possibility of operator error. A more substantial benefit could be the reduction in chemical consumption that is enabled by the more accurate control of chemicals dosed that allows operations to minimise the safety margin on the amount of chemicals dosed.

Ratio control is also very useful is setting fuel to air ratios (Williams, 2023) in furnaces or setting fuel and air in ratio with a process stream entering the furnace. This can be beneficial in reducing the excess oxygen, which causes sub-optimal combustion. This can save energy as well as improve temperature control.

Ratio control can be used to set either the distillate or the bottoms flow of a distillation column at a ratio to the feed. The desired ratio can then be changed by a master controller in a cascade control scheme. Ratio control is often used with cascade control to introduce feedback, where a master controller writes

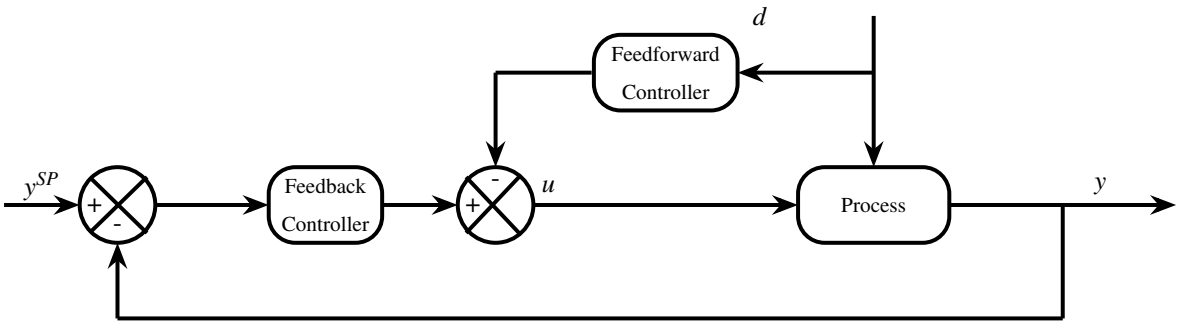

Figure 2.2. Biased feedforward control.

a setpoint to a slave ratio controller. An example is a level controller that manipulates the ratio between a distillation column bottoms flow and the feed flow (Gous, 2000).

2.3.3 Feedforward control

Feedforward control is used to compensate for the expected effects of measured disturbances before they influence the system's performance (Seborg et al., 2016). The effect of a change in the disturbance variable on the controlled variable is modelled and the inverted model is used as the required response of the controller output (Morari, 1987). In modern DCS the response to the disturbance is either added as bias or multiplied as a ratio to the controller output (King, 2016). Biased feedforward control is shown in Figure 2.2.

In the processing industry, typical examples of feedforward control include using the steam pressure or the feed flow as a feedforward signal to heaters to improve temperature control in heaters. A higher steam pressure is an indication of more energy that will increase the outlet temperature of a heater, while increased feed flow will cause the temperature to decrease. Here the obvious benefit is improved temperature control that can lead to reductions in energy consumption, improved conversion in temperature sensitive reactions and better separation in downstream distillation columns.

Feedforward control is usually used in combination with feedback control (Faanes and Skogestad, 2004; Liu et al., 2019) to improve control performance and stability. Feedback control allows for adjustments based on the actual system response, addressing any discrepancies between the predicted and actual behaviour.

Considering steam pressure as a feedforward signal, a typical deployment will have the steam pressure as a feedforward signal that is multiplied with the output of a feedback temperature controller. The

temperature controller will control the heater outlet temperature by manipulating a steam flow setpoint of a steam flow controller. This combines feedforward, feedback and cascade control. Should the steam pressure drop suddenly, the feedforward component will immediately increase the steam flow setpoint by increasing the output of the temperature controller. The temperature controller will use its PID abilities to address any remaining deviation, while the steam flow controller will act very quickly to the desired increase in steam flow. Gous (1993) implemented this feedforward scheme on all the heaters of a petroleum refinery. The outcome was that whenever the steam pressure decreased, all the feedforward controllers immediately demanded more steam, exacerbating the shortage of steam. Therefore it is important to prioritise where this will provide the greatest benefit, and only use feedforward control in those limited instances. It is also necessary to be aware that other steam consumers may suffer as a result.

2.3.4 Adaptive tuning

Adaptive tuning refers to controllers where the tuning constants are changed to adapt to process conditions. This may be to adapt to non-linear process responses such as pH or it may be to change the aggression of the controller based on the error (King, 2016). Many DCS have built in adaptive control capabilities or some coding may be required.

A very well known example of adaptive tuning is pH control, where the response of the pH to reagents is very non-linear depending on the current pH. This can be addressed by changing the tuning depending on the current value of the pH, or by transforming the pH so that the transformed variable exhibits a linear response. Henson and Seborg (1994) and Gustafsson and Waller (1992) showed that adaptive tuning in a pH controller improved robustness and control performance.

Adaptive tuning can also be done on MPC. An example is the temperature control of a batch reactor in a precious metal refinery (Singh et al., 2010). A jacketed batch reactor is loaded with reagents and the heating valve is opened fully to heat up the batch. When the desired temperature is reached, the MPC controller is activated and starts controlling the reactor temperature. Initially the control performance was poor because batch sizes varied dramatically, impacting the amount of steam or coolant required for controlling the temperature. The steam supply pressure is also subject to disturbances and this also impacts control performance. This is solved by measuring the rate of change of the temperature during the initial heating phase. A gain multiplier is calculated based on the speed at which the batch heats up. The model of the steam valve opening to the reactor temperature is adjusted by multiplying the model

gain with the gain multiplier. This solution addresses both the impact of the size of the batch and the pressure of the steam. The advanced controller delivered a greater than 20% reduction in temperature standard deviation from the setpoint value, ultimately resulting in a 10.5% reduction in total batch duration, and 2.3 % saving in reagent consumption.

2.3.4.1 Gain scheduling

Gain scheduling is a form of adaptive tuning where K_c of (2.1) is changed according to process conditions or to change the controller response (Seborg et al., 2016). It is a control strategy used in systems where the process response varies significantly over the operating range. It involves calculating different values for K_c to be used during different operating states and switching to the correct value when the process state changes. This approach helps maintain optimal control as the process moves over the operating range.

One often used application of gain scheduling is in the control of batch processes. As the reaction progresses and the properties of the mixture change, the gain is adjusted according to a predefined schedule to compensate for these changes and maintain the desired reaction conditions (Singh et al., 2010).

In the petrochemical industry, gain scheduling is particularly useful due to the non-linear and time-varying nature of many processes. For example, in a distillation column, the separation efficiency can vary with changes in feed composition, temperature, and pressure. Gain scheduling allows for different PID controller settings to be applied depending on the current state of the process, ensuring better product quality and process stability. An example is where K_c of a temperature controller is increased, based on an increase in feed flow to a distillation column that needs to be heated to boiling point, in order to compensate for the increase in heating demand. Here the value for K_c to be used is calculated in real time, based on process conditions. McDonald and McAvoy (1987) used gain scheduling and time constant scheduling that resulted in significant improvement in regulatory control performance on a non-linear, binary distillation column.

Another use for gain scheduling is to increase controller aggression by increasing K_c when e_k increases (Reyes-Lúa et al., 2018). The value of K_c is increased, based on the error or the rate of change of the error. This could help the control system to recover from large disturbances quicker, while reducing the impact of the controller when the error is small.

2.3.4.2 Gap control

Gap control is a form of gain scheduling that is often used in industry when operations are comfortable with y deviating from y^{SP} to some extent. Gap control is used to have the controller react less to smaller disturbances, and react more when larger disturbances occur (King, 2016).

Gap control is done by defining a band around y^{SP}. If y is inside this band, a smaller value of K_c is used, while K_c is increased when y goes outside of the band. This is often used for averaging level control (King, 2016) where the larger value of K_c is used to increase control action in order to prevent the level from breaching alarm or trip limits. The smaller value of K_c is chosen to let the level move around while make smaller moves in u. Care must be taken because of the nature of integrators, where the absolute value of the level is not dependent on u, but the rate of change of y depends on u. If the smaller value of K_c is used in the gap, it may allow the level to slowly drift through the gap, until it leaves the gap and the larger value of K_c is used. If this value is too large, it will turn the level until it once again enters the gap and is allowed to again slowly drift through the gap. This will easily lead to a steady cycle in the level.

2.3.4.3 Error-squared control

Error-squared control (King, 2016) is a more advanced algorithm than gap control. Here the range of controller error is normalised to 1 by dividing the error by the range of y or by the desired maximum deviation from y^{SP}. Equation (2.1) is changed by multiplying K_c by the absolute of the error:

$$\begin{aligned}\Delta u_k &= u_k - u_{k-1}, \\ &= K_c|e_k|\left[e_k - e_{k-1} + \frac{t_s}{\tau_i}e_k + \frac{\tau_D}{t_s}\left(e_k - 2e_{k-1} + e_{k-2}\right)\right],\end{aligned} \tag{2.2}$$

K_c is multiplied by the absolute of e_k to preserve the sign and retain the controller direction. As the error increases, so will the controller response, which will ensure increased controller action as y moves away from y^{SP}. Error squared control also increases the control action as y moves away from y^{SP} but does so smoothly and there is no marked transition between weak and strong control action. Because error squared control greatly increases controller action as the error increases, it is often used for slug control in the oil industry(Sausen et al., 2012). Later Sausen, Sausenand and de Campos (2014) showed that imposing limits on the non-linear gain in error-squared control improves control performance.

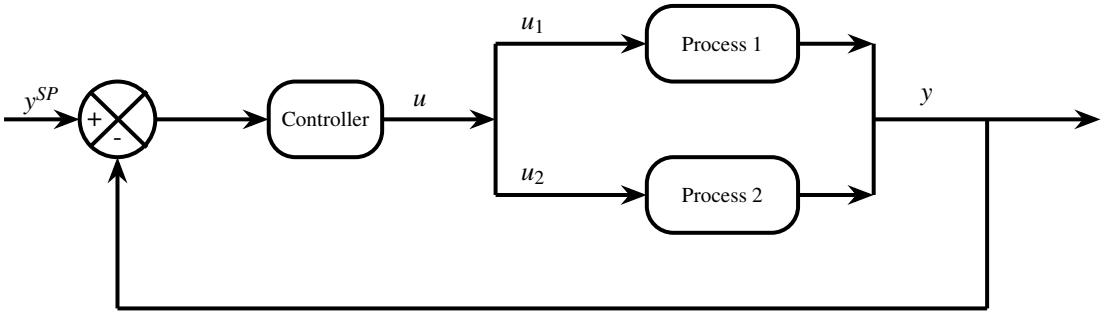

Figure 2.3. Split range control.

2.3.5 Split range control

Split range control is used where u is split into multiple outputs that are used in sequence to control the input (Seborg et al., 2016; King, 2016). The output signal of the controller is mapped to more than one output to let the next control element take over when the previous one saturates. Examples of split range control can include level controllers that open valves to multiple downstream equipment, one at a time, or a pressure controller that either lets excess gas off to flare, or introduces gas to a system, based on the measured pressure (Mitra, 2015).

An example of a split range controller (Reyes-Lúa and Skogestad, 2019) is shown in Figure 2.3 where 70 to 100% of u of a pressure controller is mapped to 0 to 100% of u_1 of a flare valve (Process 1). It also maps 70 to 0% of the pressure controller's output to 0 to 100% of u_2, a nitrogen supply valve (Process 2). In this way, if the pressure is very high the controller's output should be at 100%. This translates to a signal of 100% being sent to the flare valve, flaring maximum gas. As the pressure decreases, the controller output will decrease, decreasing the flaring until the flare valve closes when the controller output is at 70%. From 70% down to 0% the nitrogen addition valve will open from 0 to 100%. Choosing the switch over point as 70% means that the control engineer believes that the flaring decreases the pressure faster than what the nitrogen addition can do to increase it.

Another example of split range control is when pH is controlled by either bubbling CO_2 through a slurry when it is too alkaline, and by transitioning to dosing Soda-Ash when the pH is too low (Gous, 2021). As the impact of the two reagents will not have the same effect on the pH of the slurry, it is customary to scale the range of each reagent accordingly to compensate for the different process gains. As an example, if using stronger acid and a weak base to control pH, 0 to 30% of u of the pH controller may be mapped to 0 to 100% of the acid valve while 30 to 100% of u will be mapped to 0 to 100% of

the soda ash valve. It is also possible to compensate for the differences in dynamics, (Reyes-Lúa and Skogestad, 2020), but this is not done often in industry due to technical difficulty and time constraints when done using typical DCS or PLC functionality.

Reyes-Lúa and Skogestad (2020) showed that splitting the range and sending the signal to different controllers instead of different MVs, while using a baton system whereby the active controller decides to pass control to the next when required, improved control performance when controlling room temperature using air conditioning, hot and cold water and electric heaters.

2.3.6 Decoupling control

Decoupling is used when controller outputs affect multiple process outputs (Skogestad, 2023; Seborg et al., 2016). Decoupling is done by selecting the controller output with the largest influence on a certain process output to control that output, and then attempting to minimise the influence of other controller outputs on that variable. This is done similarly to feedforward control, by compensating for the expected effects of other controller outputs.

Dynamic or static decoupling can be done. Static decoupling negates the impact of the gain of the other controller outputs, while dynamic decoupling uses an inverted process model to predict the impact of the other controller outputs on the process input under consideration (Skogestad, 2023).

A typical example of a process where multiple inputs impact multiple outputs is a distillation column. In theory it should be the ideal candidate for dynamic decoupling control to enable the control engineer to control the temperature in the column bottoms as well as the distillate temperature. However, because the impact of the different MVs available have very similar models to the CVs, the large elements in the relative gain array (RGA) (Chen and Seborg, 2002) show that the system is prone to instability, because small errors in the process model used will have a large impact on the control performance (Wolff and Skogestad, 1996).

Decoupling control was successfully deployed on coal fired boilers that burn coal to generate steam (Rambalee et al., 2010). Each boilers furnace has two coal supply conveyors next to each other and the speed of each conveyor can be controlled independently. The boiler also has two forced draft fans with dampers that are used to control air flow into the boiler and through the coal conveyors. After combustion of the coal, the gas from both coal conveyors are combined and one induced draft

fan per boiler controls the flow of gas leaving the boiler through a chimney. The pressure inside the furnace must be kept at a slight vacuum in order to prevent the hazard of flames exiting the furnace. The furnace pressure is measured at both coal conveyors. The obvious variable pairing is to control all variables on one side using the manipulated variables on that side. However, the coal supply on the left side would also increase the pressure on the right, albeit to a smaller extent. Opening the dampers on the left side would increase the pressure on the left, but it would also slightly increase the pressure on the right as the left side is now demanding more from the induced draft fan in the chimney. Improving the balance of pressure on the sides of the furnace improved safety as well as combustion efficiency.

2.3.7 Selectors

Selectors are used to direct or switch the flow of signals in a control system (Seborg et al., 2016). They can select between different:

- controller inputs,
- controller outputs,
- controller modes, or
- different setpoints

based on process conditions or states (Skogestad, 2023). Selectors help in automating decision-making processes within the control system to achieve desired control objectives. Different controller inputs can improve safety by e.g. selecting the highest of multiple tubeskin temperatures in a furnace, while different outputs can be selected in order to change the behaviour of an averaging level controller (Gous et al., 2023). When validation of control inputs in an outer cascade loop fails, selectors can be used to switch the mode of an inner loop from cascade to auto. Selectors are also often used to select different setpoints during batch processes.

2.3.8 Override control

Override control is also called CV-CV switching or MV-MV switching. An excellent example of this is adaptive cruise control (Skogestad, 2023), where the controller will switch y between selecting the speed of the vehicle, or the distance to the vehicle ahead, while it will also switch u between using the fuel flow or the brakes, depending on different process states.

What is also clear from the cruise control example is how ARC techniques and terminology overlap with different application areas, as adaptive cruise control can be grouped under and makes use of CV-CV switching, MV-MV switching, override control, selectors and split range control.

2.3.9 Deadtime compensation

Deadtime in a process response is caused when the response in y takes a substantial amount of time after a change in u (Seborg et al., 2016). Deadtime may be caused by transportation delays such as material being transported on a conveyor belt or feeder belt. If more ore is put on the conveyor, the effect will only be seen by downstream equipment once the increase in ore has travelled the full range of the conveyor. Deadtime can also be caused by equipment such as analysers that provide periodic updates of y.

Deadtime creates severe difficulties for a control engineer. Because of the delayed response a PID controller that is tuned as if there is no deadtime will start a cycle. The controller will take proportional, integral and derivative action on the current value of the error, but y will not change until the deadtime has elapsed. During this time, constant PID action will be taken as there will be no change in error. When y eventually starts responding, too much movement of u that has gone beforehand will cause y to overshoot y^{SP}. The only way to compensate using a normal PID controller is to choose tuning constants that will move u very slowly, which will severely impact controller performance (King, 2016).

2.3.9.1 Smith predictor

Smith described the Smith Predictor in 1957 (Smith, 1957). The Smith predictor compensates for deadtime (d) by using a PID controller with a fast model that calculates y_{fast}, which is the value that the process output y should be at steady state, based on past changes in u. This model ignores the deadtime. Because the model has no deadtime, the PID controller can be tuned more aggressively than would be advisable for the original process, in order to provide high performance and stability.

A second model that includes the deadtime is also used to predict y_{slow}, which is the actual value of y at steady state based on past changes in u. If this is done, the result of the slow loop should be equal to the result of the fast loop, after a period of time equal to the deadtime. This means that $y_{slow}(t)$ should be equal to $y_{fast}(t+d)$, which should also be equal to y. Because of model error and noise and unmeasured disturbances, this will not be the case. Therefore feedback is introduced by calculating a correction factor by subtracting y_{slow} from y and adding this amount y_{fast}. The calculation of the correction factor is shown in Figure 2.4.

Grimholt and Skogestad (2018) found that robustly tuned PID controllers can outperform Smith predictors in processes with significant deadtime.

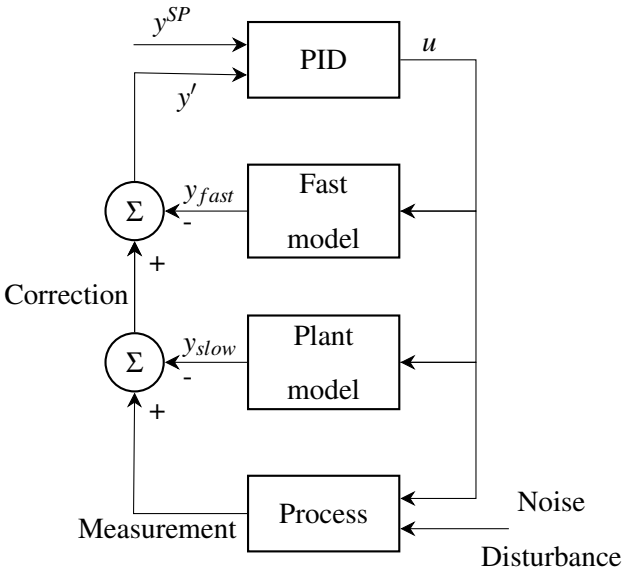

Figure 2.4. Smith predictor configuration.

2.4 AVERAGING LEVEL CONTROL

When applying averaging level control in a process plant, the control engineer will consider the available capacity of storage vessels such as feed drums and will implement controllers that will allow the level to move up and down in order to minimise movement of the manipulated variable (u). The engineer will also consider what range of the level in the vessel can be used in order to minimize variability in u. The upper limit of the level may be limited by process factors such as liquid carry-over, as well as alarm or trip limits. The lowest level allowed may be limited by process considerations such as pump cavitation, as well as alarm or trip limits. The engineer will choose limits that are a safe distance away from such constraints and will implement and tune controllers that will rarely go over these limits (Seborg et al., 2016).

When controlling self-regulating variables, a change in u will typically lead to a change in y, where y will follow a dynamic path to a new steady state value. The dynamic change in y can typically be described in Laplace transform format as a first-order-plus-time-delay (FOPTD) response:

$$Y(s) = \frac{K_p e^{-\Theta s}}{1+\tau s} U(s) \tag{2.3}$$

where K_p is the process gain, Θ is the time-delay and τ is the process time constant or lag.

If a system is at steady state and u is increased by δu, y will increase over time to reach steady state at a value of $K_p \delta u$ higher than before.

When considering integrating variables, a change in u will not lead to a new steady state value in y, but y will rather follow a dynamic path to steady state at a new rate of change, as shown in (2.4).

$$Y(s) = \frac{K_p e^{-\Theta s}}{s} U(s) \tag{2.4}$$

Averaging level control is often used to control integrating variables in order to improve stability of processing plants and has been widely studied in literature. Taylor and La Grange (2002) used P-only control to make optimal use of vessel surge capacity. Reyes-Lúa et al. (2018) compared the performance of a P-only, a PI and a gain-scheduled PI controller with an MPC controller and found that they could get the same performance with less engineering effort. Cheung (1979) studied the impact of having multiple vessels in series. Other authors focus on proper tuning of PI controllers for averaging level control (Wade, 2005; King, 2016; Seborg et al., 2016). Cheung and Luyben (1980) and Wu et al. (2001) showed that non-linear control could outperform P-only and PI controllers. Van der Burg and Djavdan (1995a) showed how the impact of disturbances can be modelled and mitigated using MPC, and Gupta (1998) demonstrated the advantages of using MPC for averaging level control. McDonald et al. (1986) proposed a controller called the optimal averaging level controller that calculates a constant value for u that is implemented until $\delta y = 0$ and y is at the limit. Ye et al. (1995) tested an optimal averaging level controller on the Tennessee Eastman problem (Downs and Vogel, 1993) and Lakerveld et al. (2013) showed that an optimal averaging level controller could outperform a PI controller, with the aim of reducing the production of off-specification pharmaceutical products.

The control engineer can typically choose between PID based control algorithms such as proportional-only, gap and non-linear PID or model based control such as model predictive control (Qin and Badgwell, 2003; Gupta, 1998) or optimal averaging level control (McDonald et al., 1986).

2.4.1 PID used for averaging level control

In industry, the most widely used form of process control is the PID controller. PID controllers typically run on either a DCS or a PLC. A typically used PID equation in industry is shown in (2.1). Several additions or manipulations of the base PID algorithm have been applied in order to improve the control performance for specific systems being controlled. This includes P-only control, PI control, proportional gap control, PI gap control and non-linear control, where certain aspects of the PID algorithm are either enhanced or not used.

2.4.1.1 P-only control

P-only level control (Taylor and La Grange, 2002; Rosander et al., 2011) is a commonly used form of averaging level control found in industry. Typically, P-only control is done in one of two ways. The first approach is to use a PID controller in a DCS or PLC, where only the proportional action term in (2.1) is used. The second approach is to map u of a controller to the current value of y to do P-only level control (Sanchis et al., 2011).

With P-only control, the error will increase as y moves away from y^{SP}, with the control action moving u in proportion to the increasing error. Once u has moved enough to restore the mass balance, the error will stay constant because y does not change. Therefore, no control changes will be made to return y to y^{SP}, the controller will only reduce the rate of change of the level to zero.

The use of P-only control is based on the assumption that the most likely event after a feed disturbance to a vessel, is another feed disturbance in the opposite direction (Taylor and La Grange, 2002). The advantage of P-only control is that if the assumption is correct, movement in u will be less than when using a PI controller. The disadvantage of the P-only controller is that, if the assumption is incorrect, y will probably go over the desired limits.

Tuning a P-only controller is done by deciding on a maximum offset (d) that y may deviate from y^{SP}. This is normally chosen to be a safe distance from the nearest alarm or trip limit. The controller gain (K_c) in (2.1) is then set as $K_c = \frac{50}{d}$ (King, 2016).

2.4.1.2 PI control

Typically averaging level control using PID is done without derivative action, i.e., the last term in (2.1) is neglected (Seborg et al., 2016). This is because derivative control action can not only result in fast changes in u, but it can also overreact to process and measurement noise which will then increase variability in u. Therefore the most common control used for averaging level control is a PID controller with τ_D set to zero, also called a PI controller.

PI controllers are quick and easy to implement but will allow the level to go outside of the limits when disturbances are larger than the assumed maximum disturbance used during tuning. When using PI controllers, the upper and lower limits will be the same distance from the desired average value of the

level or setpoint (y^{SP}) as the controllers will attempt to minimise the distance between the current level measurement or process variable (y) and y^{SP}.

A common tuning mistake made in industry when using a PI controller is implementing too much integral action in order to move y back to y^{SP}. When a level is subject to a disturbance, y will start moving away from y^{SP}, causing the error (e_k) to increase if the disturbance moves y upward, above y^{SP}. While y is moving away from y^{SP}, the proportional action, $K_c(e_k - e_{k-1})$ and the integral action, $K_c \frac{t_s}{\tau_i} e_k$, will have the same sign. Therefore, while y is moving away from y^{SP}, both proportional and integral action will work together to reject the disturbance.

When enough control action has been taken to reduce the rate of change of the level to zero, proportional action will be reduced to zero because $(e_k - e_{k-1})$ will be zero. The integral action will still move u in order to reduce e_k to zero and y will start returning to y^{SP}. During this phase proportional and integral action will oppose each other. The integral action will still attempt to reduce the error to zero while the proportional action will react to the reduction in error by opposing the integral action. The integral action drives the error to zero while the proportional action prevents y from overshooting y^{SP}. The PI tuning should be such that the proportional control is strong enough to prevent overshoot. If this is not done correctly, a slow, sustained cycle in the level may result.

When implementing averaging control, the dilemma is that weak proportional control is needed to allow y to deviate from y^{SP}, thereby minimizing the movement of u. However, the need to have the integral action weaker than the proportional action when y is returning to y^{SP} remains. Therefore, in averaging level control the integral action must be tuned to be very weak to prevent the level from cycling. Very weak integral action results in the level taking a long time to return to y^{SP}. Depending on tuning and range allowed for y, it will take much longer for y to return to y^{SP} than the time it takes to initially balance the level. This may be a problem depending on the frequency contents of the disturbances. Djavdan (1993) suggested a PI controller with filtered proportional action for improved averaging level control performance. Rosander et al. (2012) proposed and tested a P-only and PI controller that mimic the behaviour of MPC controllers.

There are tuning rules in literature that will provide default or starting tuning values for K_c and τ_i. These values are based on an assumed maximum flow deviation (f) in the disturbance stream shown in Figure 2.5. This disturbance is assumed to occur when y is at y^{SP} (King, 2016; Shinskey, 2010)

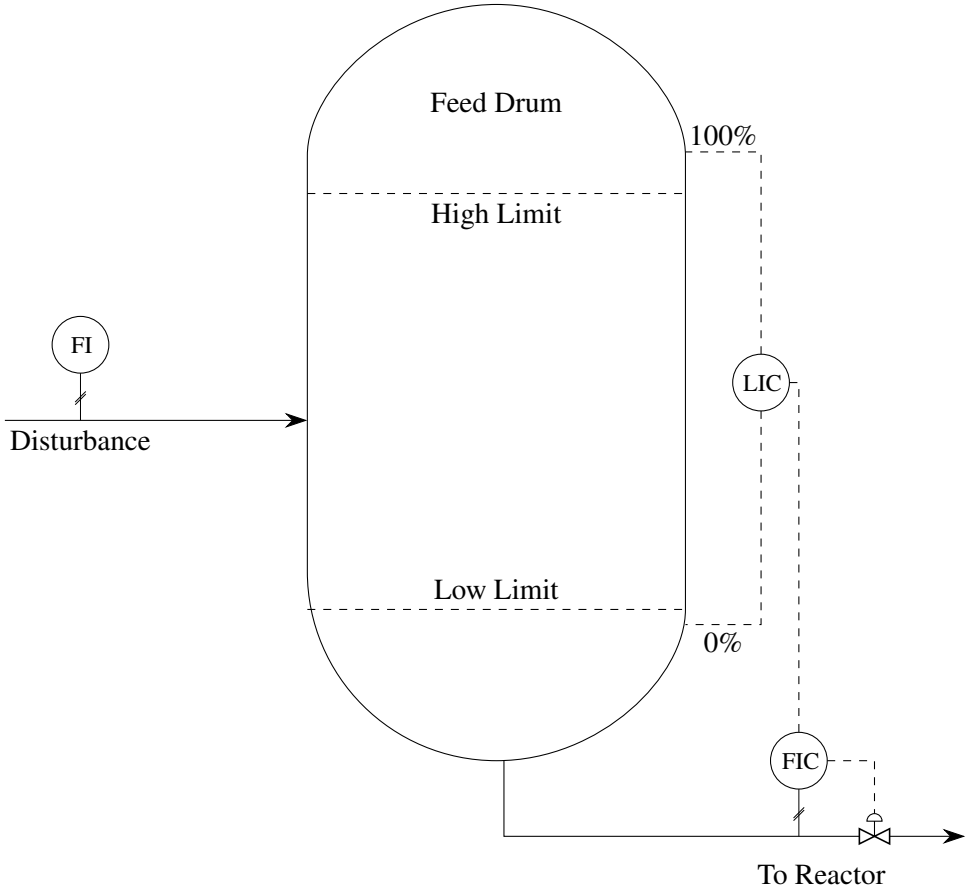

Figure 2.5. Feed drum with PI level control.

and the rate of change of y is zero. When this happens, d is the maximum distance that y should move away from y^{SP} and is usually less than the alarm or trip limits, shown as the high and low limits. Once implemented, the process response to these starting values can then be adjusted to better suit the process or the preferences of the operators. These starting values for K_c and τ_i will be denoted as $K_{c(PI)}$ and $\tau_{i(PI)}$.

For a PI controller the tuning constants for (2.1) are calculated (King, 2016) as:

$$K_{c(PI)} = \frac{80f}{Fd}, \tau_{i(PI)} = \frac{Vd}{12.5f} \tag{2.5}$$

where F is the range of the flow used as the manipulated variable, d is the maximum desired deviation between y and y^{SP}, with V as the volume of liquid between 0% and 100%, shown in Figure 2.5. This is an empirical set of equations that have been proven effective in industry (King, 2016).

The volume of the vessel (V) can be calculated from design drawings using the positions where the

level measurement is on the drum or tank, not the total volume of the vessel. This is the volume between 0% and 100% in Figure 2.5. If the area of the vessel remains relatively constant from the high measurement position to the low measurement position, the relationship between the vessel's measured volume and the flow can be assumed to be constant. If this assumption is reasonably accurate, a step test may be performed to calculate V.

This is done by getting the system to steady state, where the rate of change of the level is zero. A step change ΔF_{ST} is then made either in the flow into or the flow out of the vessel. If the flow is not measured, then a step change in u is made. The level will then either begin rising or dropping at a constant rate of change. This imbalance will be maintained for a set time t, until a substantial change in level is seen as shown in Figure 2.6 but will be stopped before the maximum deviation is reached. The volume can then be calculated as:

$$V = \frac{100\Delta F_{ST} t}{\Delta L} \tag{2.6}$$

Care should be taken to ensure that the engineering units, specifically to ensure that the units of time for t and ΔF are the same.

Should the relationship between the height of the level in the vessel and the volume not be linear, the linearisation should be done as explained in King (2016) to ensure proper functioning of the PI controller.

2.4.2 Model based control on integrating variables

Model based control can be used to control integrating variables by obtaining a model between u and y. This model can be inverted to determine what to do with u to get a desired response in y. Common examples of model based control include MPC (Qin and Badgwell, 2003), Dynamic Matrix Control (DMC) (Cutler and Ramaker, 1980) and optimal averaging level control (McDonald et al., 1986) as discussed in Section 2.4.2.1. Gupta (1998) showed that a steady-state offset can occur during sustained load changes when using DMC for controlling integrating processes. A simple modification in the DMC algorithm that eliminates this offset is presented.

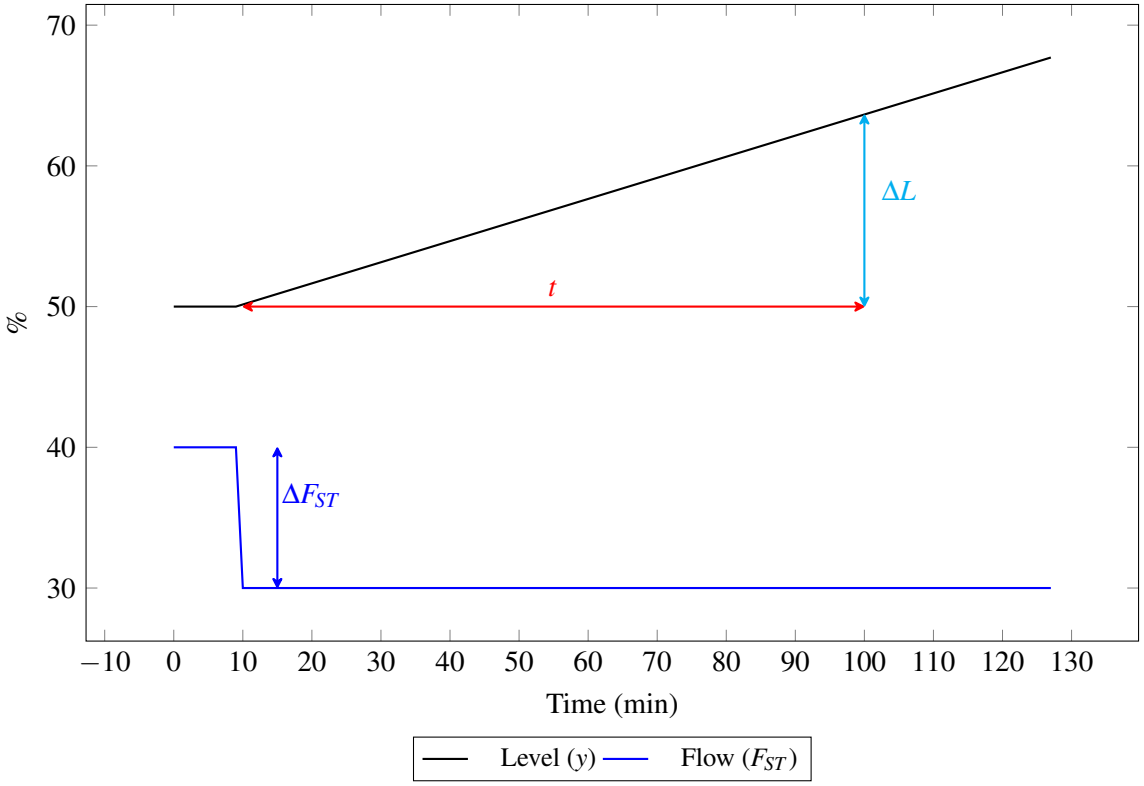

Figure 2.6. Level step test.

van der Burg and Djavdan (1995b) used MPC with the addition of disturbance prediction in order to improve averaging level control performance. Rosander et al. (2011) investigated the impact of several flow changes in the same direction on averaging level control and showed similar performance using a robust MPC and the optimal averaging level controller. Campo and Morari (1989) used the optimal averaging level controller but added integral action to the controller in order to have y return to y^{SP}. They also introduced a single parameter to balance the objectives of minimising Δu and a rapid settling time.

In order to implement model based control, additional computer hardware and model predictive software are required. This is typically installed alongside the DCS or PLC of the plant and connects using communications protocols such as OPC. Because of this additional complexity, model based control typically runs at slower execution intervals than ARC or PID controllers. It is also required that a good model for the level behaviour is available (Lakerveld et al., 2013) which is typically acquired through plant testing and model identification. These requirements lead to additional complexity and cost when comparing model based controllers with DCS or PLC based controllers. However, they can be more successful in keeping the level between limits as no assumed maximum disturbance is used

during tuning. Model based controllers typically do not use a setpoint and the engineer can set high and low limits independently.

2.4.2.1 Optimal averaging level control

Optimal averaging level control (McDonald et al., 1986) uses MPC to predict the trajectory of the level, and to calculate control moves that result in the smallest continuous moves in u that will keep the level between limits. These moves will be made until the level limit is reached and the level is balanced. Optimal averaging level control has been implemented and tested widely and it is accepted that it improves averaging level control performance.

Ye et al. (1995) applied the optimal averaging level controller successfully to the Tennessee Eastman problem. Optimal averaging level control is used together with the base layer control to improve product flow stability during random and step feed disturbances. Simulation results show that variation of the product flow rate is significantly improved by using the optimal averaging level controller.

Lakerveld et al. (2013) compared the performance of conventional PI level controllers with optimal averaging level controller. The aim is to reduce the production of off-spec material in continuous pharmaceutical processes by minimizing the variations in the outlet flow rate of upstream buffer tanks that send reagents to pharmaceutical reactors. The results show the optimal averaging level controllers strongly outperforming the PI controllers.

As most DCS based controllers make control moves that are a function of the error, the PID based controllers will initially make larger moves after a disturbance as the error will increase quickly. Therefore, PID controllers will initially make large moves that become smaller as time progresses. This gives optimal averaging level control a distinct advantage over PID controllers as it implements the same size move over the full control horizon. This is especially true if the largest move made over the control horizon is used as a measure of success.

Optimal averaging level control requires additional computer and network hardware and software, plant tests and model identification that increases cost and complexity.

2.4.2.2 Programmed imbalance ramp control

Commercially available MPC software uses an averaging level control feature known as programmed imbalance ramp control (Qin and Badgwell, 2003; Gupta, 1998; Brooks et al., 2017). The controller

uses model based prediction to predict the future trajectory of the level. The engineer sets a high and low limit that should not be exceeded as well as a tuning parameter called the ramp horizon. This is a set timespan into the future that the predicted trajectory of the level should be kept between the high and low limits.

If the level is predicted to cross a limit at a time further into the future than the ramp horizon, the controller will not act. As time passes, the level will move towards the limit. Eventually the level will get close enough to the limit that the current rate of change of the level will cause it to cross the limit in a shorter amount of time than the set ramp horizon.The controller will then make a small move that will decrease the rate of change of the level just enough so that the level will cross the limit at the ramp horizon.

As the level approaches the limit in this way, the controller will keep making control moves that will decrease the rate of change of the level until the rate of change becomes zero and the level is balanced as the limit is reached.

As with optimal averaging level control, this control method also requires additional computer and network hardware and software, plant tests and model identification that increases cost and complexity.

2.4.3 Advanced regulatory control on integrating variables

An alternate approach to model based averaging level control is to use ARC techniques which can be implemented using standard DCS logic blocks. Such techniques can yield results similar to model based averaging level control, without the requirement of additional computer hardware and model predictive software. ARC techniques are often used in industry, and are recently receiving increasing attention from the academic literature (Skogestad, 2023).

Taylor and La Grange (2002) compared the performance of a P-only, a non-linear and an MPC controller and concluded that not including the averaging control in the MPC, but in a stand-alone P-only controller has advantages. Cheung (1979) investigated using P-only and PI level controllers and provided design charts to ensure proper tuning to maintain levels within limits while minimising the movement of u. This work was followed up by examining the advantages and pitfalls of non-linear PI control (Cheung and Luyben, 1980).

Sausen, Sausen and de Campos (2014) proposed limits for the non-linear gain of an error-squared controller to ensure stability. Reyes-Lúa et al. (2018) proposed a PI-controller for normal operation that switch to one of two high gain P-only controllers to prevent the level going over limits. When comparing this approach to MPC, it shows similar performance.

Reyes-Lúa and Skogestad (2019) shows how economic optimisation can be done using ARC when active constraints change. This is done by switching CVs using selectors, CV-MV switching using the input saturation pairing rule and MV-MV switching using split range control. Reyes-Lúa and Skogestad (2020) also compared split range control using a new generalized control structure with MPC. MPC requires a detailed dynamic model and does not allow for using only one input at a time as the split-range ARC scheme does. Zotică et al. (2022) advocate the use of a decentralized control system that is able to maximize production when bottlenecks occur on multiple units in series.

2.4.3.1 Proportional gap control

Most modern DCS have proportional gap control as an option in their PID algorithm (2.1). Proportional gap uses lower controller proportional gain values if y is within a set range from y^{SP} (in the gap) and a higher value if y is outside the set range or outside the gap. This is typically implemented by defining a ratio:

$$K_r = \frac{K_{c(gap)}}{K_c}, \tag{2.7}$$

where K_r is the ratio between the gain used when y is inside the gap ($K_{c(gap)}$) and the gain used when it is outside the gap, i.e., the normal controller gain (K_c).

The proportional gap controller can be written as:

$$\Delta u_k = \begin{cases} K_{c(gap)}\left(e_k - e_{k-1}\right) & \text{if } \underline{y} < y < \bar{y}, \\ K_c\left(e_k - e_{k-1}\right) & \text{otherwise,} \end{cases} \tag{2.8}$$

where $\underline{y}$ and $\bar{y}$ are the lower and upper bounds of the gap respectively.

The controller will take less aggressive control action while y is in the gap and be more aggressive to prevent limit violations when y is closer to the limits. Tuning rules are available to determine the values of K_r, $K_{c(gap)}$ and K_c (King, 2016).

Gap control allows improved usage of the available volume by making smaller control moves in response to smaller disturbances, but also prevents limit violations when occasional bigger disturbances are experienced.

2.4.3.2 PI gap control

Adjusting the controller gain when applying PI gap control is done in the same way as with P-only gap control, as described in Section 2.4.3.1.

When applying a proportional gap to PI control, the proportional tuning constant is changed from $K_{c(gap)}$ to K_c and back, which affects both the proportional and integral action of the controller. Therefore, the PI controller will apply less proportional and integral action in the gap, while increasing both outside of the gap.

As the ratio of proportional and integral tuning remains the same inside and outside of the gap, the need to keep the integral action weaker than the proportional action is addressed. However, if the tuning inside and outside the gap differs too much, a situation can occur where little control action is taken when y is inside the gap and the controller will let y drift through the gap without balancing y at y^{SP}. Once y exits the gap, increased control action will balance and turn the level, which will then allow y to drift through the gap again, in the other direction (King, 2016).

If the ratio between proportional and integral is not correct, a cycling level will result as with P-only gap control. To avoid this situation, it is recommended that tuning rules in literature be adhered to (King, 2016; Friedman, 1994).

2.4.3.3 Non-linear P-only control

Another approach to increase the control action when the level gets closer to limits is non-linear P-only control. Here the proportional gain of the controller is changed to increase non-linearly as y moves farther away from y^{SP} (Kelly, 1998; King, 2016; Friedman, 1994; Sanchis et al., 2011). A commonly used non-linear P-only controller is the error squared controller (Kelly, 1998; King, 2016; Friedman, 1994). The error is not squared as the name implies but rather the error is multiplied by the absolute value of the error to retain directionality as shown in (2.9).

$$\delta u = K_c(e_n - e_{n-1})|e_n - e_{n-1}| \tag{2.9}$$

The advantage of non-linear P-only control is similar to P-only gap control. As with P-only gap control, typical disturbances will be rejected by moving u slowly and the occasional larger disturbance will be handled by increasing the control action as the error increases. Using non-linear P-only control will also allow the control engineer to make better use of the volume available by making smaller control moves during typical disturbances but also provides a smooth transition to larger control moves as the

error increases. Another approach to non-linear P-only control is to map u to the current position of y as with the P-only controller, but to use a non-linear mapping (Sanchis et al., 2011).

2.4.4 Disturbances used to evaluate different averaging level controllers

2.4.4.1 Single step

The simplest disturbance in literature is a single step change in either the flow into or out of a drum as shown in Figure 2.7 (Rosander et al., 2012; McDonald et al., 1986; Ogawa et al., 2002; Sidhu, 2003; Rosander et al., 2011; Lindholm, 2009; Lakerveld et al., 2013; Sanchis et al., 2011; Sbarbaro and Ortega, 2005; van der Burg and Djavdan, 1995b; Ye et al., 1995; Friedman, 1994). This is done with a process stream that is not used as manipulated variable by the control system. The disturbance will cause the level to increase or decrease until the rate of change remains constant, if no control action or further disturbances is introduced.The time that elapses from when the disturbance occurs until the level reaches a constant rate of change is referred to as the time to steady state. The step duration should be longer than the time to steady state in order to show the full dynamic and steady state response of the level.

2.4.4.2 Random walk

Many authors use a disturbance called a random walk (Ogawa et al., 2002; Lindholm, 2009; Sanchis et al., 2011; Horton et al., 2003; Sbarbaro and Ortega, 2005). This is a series of steps upwards and then downwards, with a sinusoidal signal with a smaller range superimposed on it as shown in Figure 2.7. This is a theoretical construct but can be used to test the control system's response to large and slow as well as small and fast disturbances simultaneously.

2.4.4.3 Continuous cycle

Some authors use a continuous cycle (McDonald et al., 1986; Sbarbaro and Ortega, 2005; Ye et al., 1995) as disturbance as shown in Figure 2.7. This cycle should have a wavelength double the time to steady state. Often cycling disturbances have to be rejected on process plants. Cycles frequently occur because of poorly tuned level or distillation column temperature controllers or because of valve stiction.

2.4.4.4 Plant data

Before implementing controllers, control engineers often simulate new control systems as well as the process they will be used on. In this case it is good practice to use actual plant data as a test disturbance signal to optimise controller tuning and performance before the controller is commissioned (Rosander et al., 2012; Lakerveld et al., 2013; Horton et al., 2003).

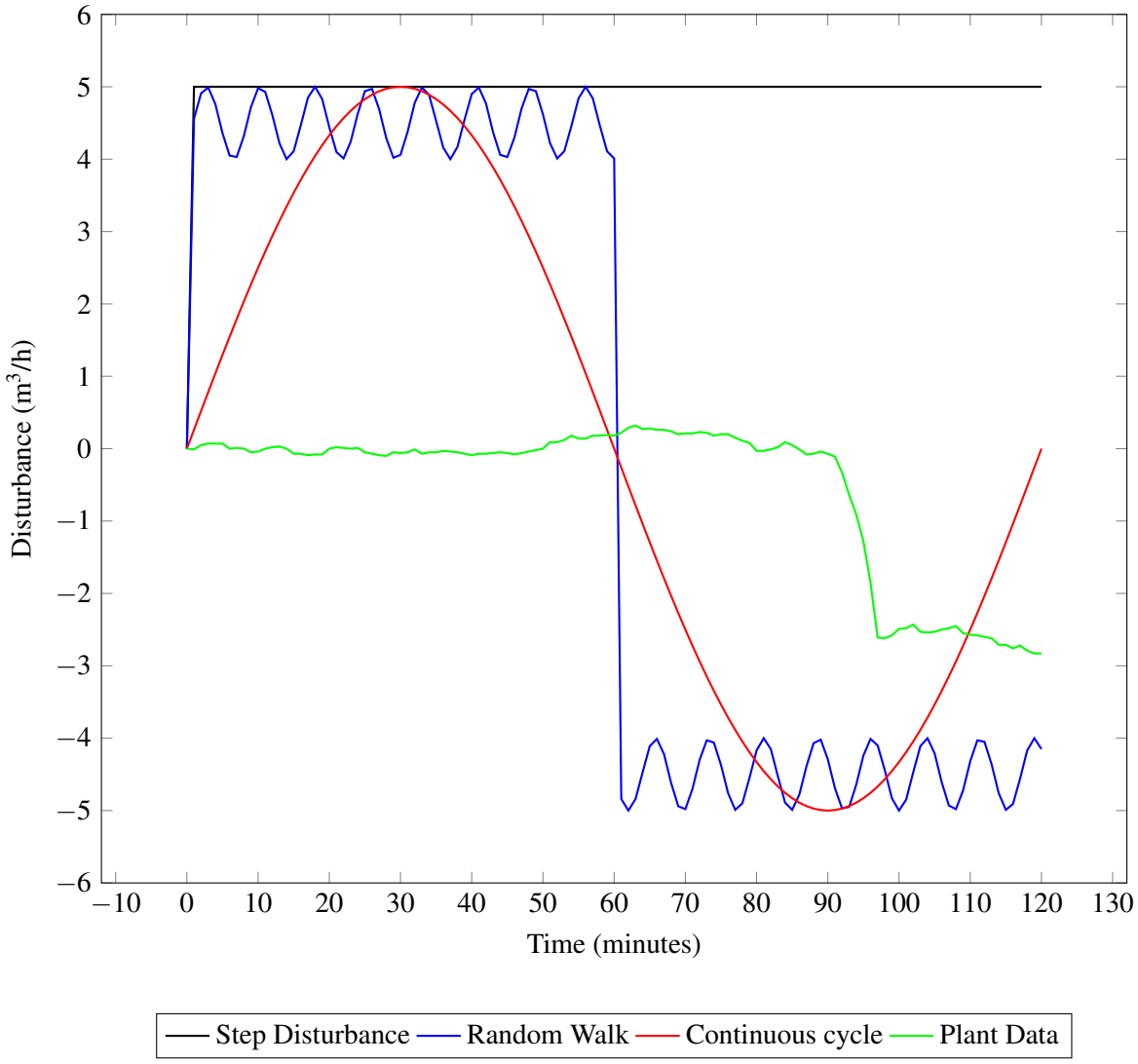

Figure 2.7. Disturbances.

Plant data is collected and used as a disturbance signal to test the performance of different controllers and tuning values. In order to start the simulation at steady state, the data set is first adapted to start at 0 by subtracting the value of the first data point from the whole series. The data used as disturbance to test the different control methods is shown in Figure 2.7.

As shown in Figure 2.7, a sequence of data with a maximum range of a little less than 5 m^3/h was used. An engineer examining the data shown in Figure 2.7 would typically assume that the maximum disturbance that should be tuned for is 5.

2.4.5 Performance assessment of averaging level controllers

Performance metrics described in this section are used to compare the success of different averaging level control techniques and the relevant tuning parameters used. Performance metrics found in literature typically only penalize the movement of u, while assuming that the level does not violate the pre-set limits. These include the standard deviation of u (σ_u) (Rosander et al., 2011; Horton et al., 2003; Lee and Shin, 2009; Brooks et al., 2017) and the total variance of u (TV) (Skogestad, 2003).

If the total variance of a dataset is divided by the number of datapoints, the result can be compared to another dataset with more or less entries, therefore TV/N is used.

$$\sigma_u = \sqrt{\frac{1}{N-1}\Sigma_{k=2}^{N}\left[(u_k - u_{k-1}) - \mu\right]^2}, \tag{2.10}$$

$$\mathrm{TV} = \Sigma_{k=1}^{N}|u_{k+1} - u_k| \tag{2.11}$$

For (2.10) and (2.11), N is the number of datapoints in the dataset and μ is the mean of u over the dataset.

When using σ_u, a control system that results in a lower standard deviation while keeping the level between the predetermined limits is desired. When using TV, the smaller the maximum rate of change of u, the better the controller is at making smaller rather than larger moves. If TV is divided by the number of samples, i.e., TV/N, a comparison between two or more datasets with differing numbers of samples can be done.

Horton et al. (2003) introduces a flow smoothing performance index for the assessment of surge tank level loops against PI controller that is used as a standard. Horton et al. (2003) not only measures the performance of the installed controller, but also specifies new settings for the PI tuning parameters. The performance metric is normalised by dividing σ_u of the controller being tested by σ_u of the benchmark PI controller. This normalises the metric to one, with a result smaller than one being better than the benchmark.

Another popular metric measures the maximum rate of change of u, which is equivalent to the maximum move of u over the dataset (McDonald et al., 1986; Rosander et al., 2012; Ye et al., 1995; Lindholm, 2009). The controller that makes the smallest maximum move in u while maintaining the level between limits is deemed most successful. This metric greatly favours the optimal averaging level controller,

discussed in Section 2.4.2.1, as it is designed to make the same size moves over the control horizon, while most other methods will show a first-order response.

2.4.6 Research in averaging level control

As shown in Section 2.4 there are 3 common methods of implementing averaging level control. The first is by using PID control as P-only or PI control, tuned to minimise Δu. The second method is by using model based control methods such as MPC or Optimal Averaging Level Control. The third method is using ARC methods such as Proportional Gap Control, PI Gap Control and Non-linear Gap Control.

Using P-only or PI control methods are simple and quick to implement as well as robust when maintaining the control solution. This is because these methods are typically implemented on a DCS or PLC using standard control building blocks. For the same reason it is also possible to run these solutions at an execution cycle of 1 second or less. The PID based solutions are not as robust when attempting to maintain the level between limits when compared to model based methods and therefore variations like gap and non-linear controllers were developed to increase aggression of control as limits are approached (Reyes-Lúa et al., 2018; Rosander et al., 2012; Sanchis et al., 2011).

Model based methods of averaging level control outperform P-only and PI methods as shown by McDonald et al. (1986). Ye et al. (1995) showed that the optimal averaging level outperforms P-only and PI controllers using the Tennessee Eastman problem. Lakerveld et al. (2013) compared the performance of conventional PI level controllers with optimal averaging level controller on continuous pharmaceutical processes. By by minimizing the variations in the outlet flow rate of buffer tanks that send reagents to pharmaceutical reactors, the averaging level controllers are able to reduce the production of off-spec material. The results show the optimal averaging level controllers strongly outperforming the PI controllers. van der Burg and Djavdan (1995b) used MPC with the addition of disturbance prediction in order to improve averaging level control performance.

However, model based controllers need separate computer hardware and software for deployment. They are typically deployed on servers that communicate with the plant DCS or PLC and therefore need network hardware and protocols as well as watchdog algorithms to monitor the health of the server and network connection. They are therefore more difficult and costly to deploy and as well as to maintain after implementation.

Many authors worked on comparing different variants of P-only or PI control with model based control like optimal averaging control. Attempts are made to improve the robustness of PID control (keeping the level between the set limits) (Reyes-Lúa et al., 2018; Rosander et al., 2012) as well as the performance of averaging control (moving u less). This is done in order to create a solution that combines the best features of both methods by being:

- robust and simple to deploy,
- high performing in terms of minimizing movement u,
- able to run at a faster execution cycle, and
- robust in terms of maintenance of the solution.

2.4.7 Stiction compensation

Stiction (Huba et al., 2011) usually applies to a valve stem that will not start moving when the control system changes the position of u. Because of the valve stiction, the controller output to the valve, will not always be the same as the actual valve position (u'). The control system will keep moving u in the same direction until the change in u is large enough to overcome the static friction and the valve moves to a new position. This is usually due to either a rough valve stem or the packing that prevents process material from escaping past the valve stem. An electronic signal u is normally sent to a valve positioner where the electronic signal is translated to a pressure signal that pushes a diaphragm in the desired direction. Once enough pressure is built up, the static friction will be overcome and the valve stem will move. The amount that u must change before the valve stem will move is referred to as the stiction band. To determine the stiction band, the control loop is taken out of automatic mode into manual mode. The engineer will start making cumulative, but small moves in u, in the opposite direction to which the valve had been moving. Once the cumulative change in u is larger than the stiction band the valve will move to a new position and the direction in which the level was moving will change. The initial value of u at the beginning of the test is taken as the actual valve position u' initially, and the value of u when the process response is seen in y is taken as u' at the end of the test. The absolute value of the difference is accepted as the stiction band.The stiction band (SB) is defined as:

$$SB = |u_i - u_f| \tag{2.12}$$

where u_i is the value of u at the beginning of the test and u_f is the value of u when the process response is seen in y.

When the valve moves, it will typically have moved too far because the integral action of the controller

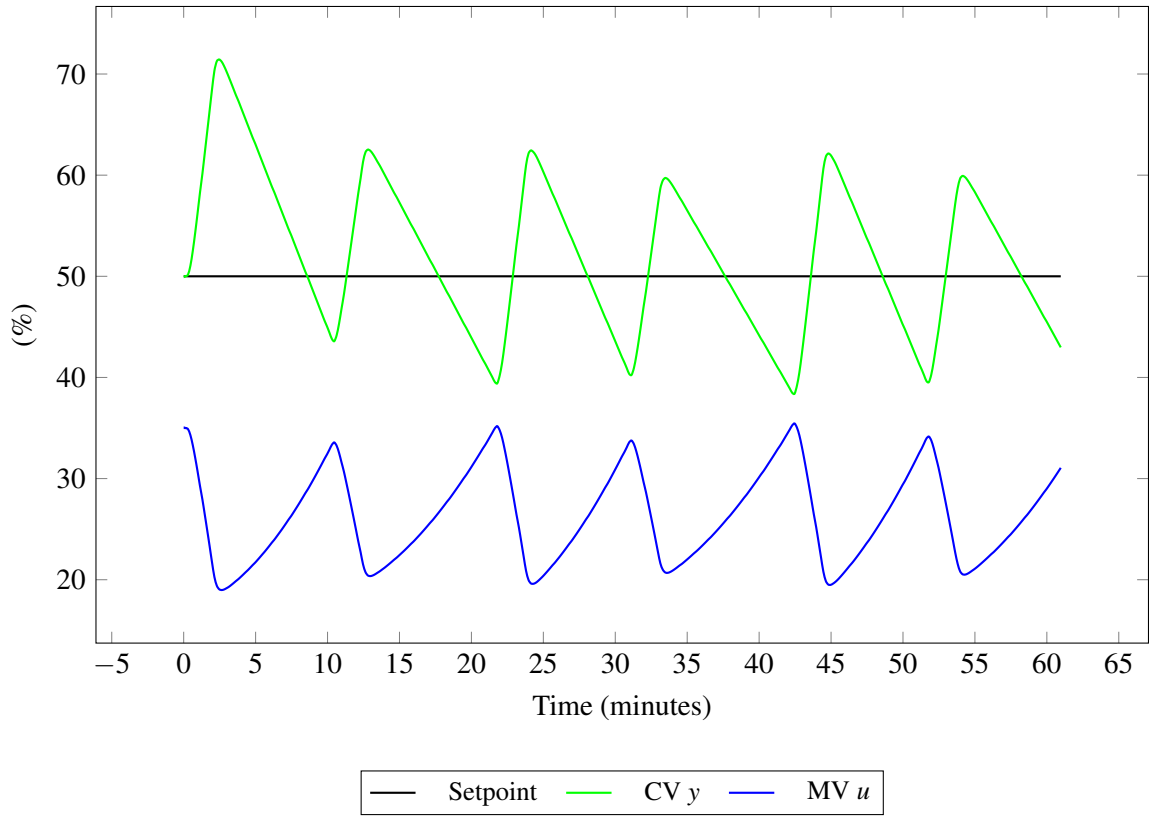

Figure 2.8. Typical stiction pattern.

would have kept adjusting u while the valve position did not change. This will cause a reversal of y which will lead to a cycle. Often this leads to cycles with sharp reversals in y as shown in Figure 2.8, which can be taken as an indication of stiction, although this is not always the case. Stiction decreases control performance in level control because levels integrate while the valve is stuck.

Valve stiction remains a problem in industry, and as a result has received significant attention in the literature. Daiguji and Yamashita (2022) developed a simulation model and a stiction model that searches for optimal PID tuning values in order to minimise the typical cycle that results when stiction occurs. The knocker method (Hägglund, 2002; Srinivasan and Rengaswamy, 2005) of stiction compensation is designed to address stiction by applying short, high-amplitude pulse signals to the actuator to create a similar effect as mechanical knocking. These pulses are designed to overcome the static friction without significantly affecting the overall control signal.

A well known method in industry is the two-moves method (Srinivasan and Rengaswamy, 2008), where, instead of moving the valve towards a new value as determined by the controller, the valve is

moved twice. The valve is first moved in one direction, in a step change that is larger than the stiction band, and then a step in the opposite direction, also greater than the stiction band, that will bring the valve opening to its desired value. This method of compensation will severely increase stress on the valve as well as introduce short but large disturbances to the process. Di Capaci et al. (2016) reduced the valve movement by revising the two-step method by temporarily changing to a PI controller and using the estimated controller output at steady-state.

De Souza L. Cuadros et al. (2012) introduced a method to only superimpose compensating pulses when required to reduce process control error. Decreasing the amount of compensating pulses will delay further valve stem degradation. The basic principle involves monitoring the control signal sent to the valve actuator and identifying when this signal is within a certain threshold or deadband range where movements are likely to be ineffective or counter-productive due to stiction. When small signal changes are detected that do not exceed this predefined threshold, the control system temporarily freezes u in its current position, preventing it from making minor adjustments that would not overcome the stiction. The valve is only moved when the control signal change is significant enough to ensure that the actuator's force will be sufficient to overcome the static friction and will effect a desired change in y.

Some authors attempt to minimise the effects of stiction by changing PID tuning (Mohammad and Huang, 2012). Control engineers attempt to solve the effects of valve stiction in two opposite ways. The first is to tune the PID controller extremely aggressively which will start a cycle. The advantage of this method is that the process may filter out the rapid changes and that, on average, y should be at the desired position. The disadvantage is that strain is added to an already failing system, which could hasten its demise, causing a plant shutdown. The second method is to substantially decrease the control action. The premise behind this method is that when stiction is overcome, the valve position will jump to a new value, and hopefully this new value will be close to where u should be. This method was shown to be ineffective (Silva and Garcia, 2014).

Stiction should not be confused with backlash (Desoer and Shahruz, 1986; Daiguji and Yamashita, 2022), or hysteresis. Backlash is a form of deadband that results from a temporary discontinuity between the input and the output of a device when the input of the device changes direction. Hysteresis is the maximum difference in output value for any single input value during a calibration cycle, excluding errors due to deadband.

Backlash occurs when the control system changes the direction of u and no effect on y is seen initially. Further changes in the same direction does not cause the same effect. A good example of backlash was found on a boiler plant where a sprocket and chain opened and closed dampers that supply air to the furnace of the boilers (Rambalee et al., 2010). The chain was not stretched tightly around the sprockets and every time the control system changed the direction of u, a significant amount of slack had to be overcome before the sprocket on the damper side would start moving. The problem was solved by shortening the chains.

The obvious solution to stiction is to replace or repair the faulty final control element. This is often not an option that is available if the plant has to be shut down in order to repair or replace the valve. Therefore, several attempts have been made at modifying or extending control algorithms by adding a compensation method in order to decrease the impact of stiction.

Another way to decrease the effects of stiction is to use a valve positioner. This is a controller that measures the valve position, compares it to u and takes control action when there is a difference. This does not solve the issue of stiction, the stiction remains. The valve positioner can however, take much faster control action as the valve response to changes in u should be much faster than say the response of a level in a feed drum. This also then adds more strain to an already broken system.

2.5 CHAPTER SUMMARY

In this chapter existing ARC methods were discussed to establish a baseline in terms of complexity. Existing averaging level controllers were discussed, both ARC and MPC versions. Disturbances that are typically used to measure control performance were discussed, as well as calculations that allow engineers to compare the performance of different averaging level control methods. Current stiction compensation methods were discussed. This was done in order to compare the new more complex methods with existing methods in terms of complexity as well as performance.

CHAPTER 3 PROPOSED ADVANCED REGULATORY TECHNIQUES

3.1 CHAPTER OVERVIEW

In this chapter four new averaging level controllers, and two stiction compensation algorithms are proposed and explained. The algorithms are explained, as well as how to deploy and tune them. These six new techniques will be used to show how more complex ARC schemes can be deployed using only standard DCS functionality and should afford the control engineer improvements in control performance.

3.2 AVERAGING LEVEL CONTROL

Four averaging level controllers are proposed. RHC, SOALC, IGC and RHIGC can all be implemented using standard DCS functionality. Therefore, they can run at a faster execution cycle which enables them to detect and reject disturbances much sooner and faster. This means that they can be implemented successfully where slower executing controllers will fail.

The proposed averaging level controllers also do not need much engineering effort to implement and tune. To tune these controllers, only a process gain of the level is required and the upper and lower limits need to be specified. Because of the faster execution cycle, errors in the process gain are less critical than in a controller executing at a slower pace. No estimate of a maximum disturbance is required for RHC, SOALC and RHIGC. They also do not require any additional hardware or software or communication protocols, reducing effort and cost of implementation.

RHC, SOAL and RHIGC are all able to ensure that a level measurement is kept within the desired upper and lower limits, regardless of the size of the disturbance. They are also better at minimising movement

of u than traditional PID-based averaging level controllers. This has been shown via simulations, using different disturbance signals of different magnitudes (Gous et al., 2021, 2023).

The simulation results show that these controllers can keep y within limits when traditional DCS-based controllers will let the limits be breached when disturbances are larger than what was assumed during tuning.

More often in practice, because assuming too small a maximum disturbance during tuning will lead to PID-based controllers letting y go outside the desired limits, the maximum disturbance is exaggerated. The simulation results in Chapter 4 show how this leads to ineffective averaging level control by the PID-based controllers while RHC, SOAL and RHIGC controllers make use of the full allowed range between limits to minimise the movement of u more effectively.

Because RHC is more effective at minimising movement in u than the SOALC, it was implemented on a process plant where it significantly decreased the movement of u as discussed in Chapter 5.

3.2.1 Ramp horizon controller

RHC (Gous et al., 2021, 2023) is a simplification of programmed imbalance control widely used in industry (Qin and Badgwell, 2003; Gupta, 1998; Brooks et al., 2017). RHC can be seen as a 'wait and see' control strategy, where control moves, Δu, are only made if there is imminent danger of the level violating a limit. It can be implemented using standard DCS functions. A number of execution cycles is selected during which the high and low limits imposed on the level may not be exceeded. This is called the ramp horizon (T_{RH}).

The trajectory of the level is predicted, based on the current position and rate of change. Control moves are only made if the predicted trajectory will violate a limit within the ramp horizon. If no violation is predicted, no control moves are made.

The value of y at T_{RH} is calculated as:

$$y_{k+T_{RH}} = y_k + T_{RH}\left(\frac{dy}{dt}\right)_k, \tag{3.1}$$

where $y_{k+T_{RH}}$ is the predicted value of the level at T_{RH}, the ramp horizon in minutes, and y_k is the current value of the level. If $y_{k+T_{RH}}$ is inside limits, no control moves are implemented. If $y_{k+T_{RH}}$ is outside limits, a control move will be calculated and implemented.

A process slope gain k' is calculated by either using step test data or calculating the volume of liquid between the high and low limits on the drum (King, 2016). The process slope gain is the change in the rate-of-change of the level that will result when the RHC output u_{RH} is moved up by one engineering unit:

$$k' = \frac{\Delta y}{\Delta u_{RH} \times \Delta t}, \tag{3.2}$$

where Δy is the change in the output y from $t = 0$ (initial steady state) to $t = \Delta t$ (a selected time after the change in u was made).

By rearranging (3.2), the size of the move in u that prevents the level limit being exceeded within the ramp horizon T_{RH}, can be calculated using the process slope gain:

$$\Delta u_{RH} = \frac{\Delta y}{k' \times T_{RH}}, \tag{3.3}$$

where Δy here is the difference between the value of the limit towards which the level is moving (also called the active limit) and the current value of y.

Consider an example of a drum where u_{RH} is a flow controller on the product line exiting the drum. The level is approaching the upper limit and will exceed the limit within the ramp horizon. Moving u up continuously per cycle will prevent the level from exceeding the limit when the prediction reaches the ramp horizon. However, as the level will still be approaching the limit, during the next execution cycle the controller will once again predict that the limit will be exceeded. It will implement another controller move to ensure that at the ramp horizon the level will still be inside the limit. If no other process influences the trajectory of the level, this process will be repeated until the rate-of-change of the level reaches zero as it reaches the high limit.

When a relatively small disturbance causes a change on a level that is controlled using RHC, the prediction will initially not show that a limit will be exceeded within the ramp horizon. Initially no control moves will be made. As time passes, the level will approach the limit and at some stage a small violation will be predicted. A control move will be calculated to change the rate of change of the level such that the level will be on the limit at the ramp horizon.

RHC will only ensure that the limits are not exceeded, and make no attempt to move the level away from the limit. The risk remains that continuous or consecutive disturbances in the same direction may occur once the level has reached the limit. When this happens, the controller will not be able to keep

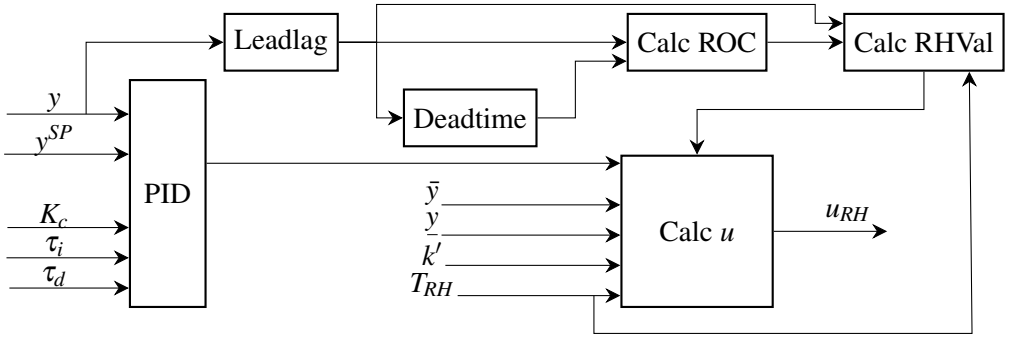

Figure 3.1. Implementation RHC on a DCS.

the level at or between the imposed limits. However, if this occurs and the level moves outside a limit, RHC will return the level to the limit that was exceeded using the same algorithm.

RHC simplifies the calculated response of the level by using a gain only. An imprecise control action might result as dynamic behaviour is ignored, measurement noise and unmeasured disturbances might be present, and the controller gain might be incorrect. However, in a short execution cycle, the level measurement that is taken as input and the rate-of-change of the level that is calculated at every execution cycle continuously updates the controller error. This compensates for the possible inaccuracies in measurement and control.

3.2.1.1 Implementing RHC

The implementation of RHC on a Honeywell Experion DCS (Gous et al., 2023) is shown in Figure 3.1. Several auxiliary calculation blocks are used. These auxiliary calculation blocks have up to 8 calculation slots that can be coded to do logic and mathematical operations on input signals as well as the results of previous calculation blocks. The input signals are denoted P1 to P8 and the calculation results are denoted C1 to C8. In Figure 3.1 these calculations are shown in the order that the calculations are done.

A standard PID block is also used- on a plant implementation this would be the current controller that controls the level. If the PID block's mode is set to manual and the mode attribute is set to Program, the block will accept values from auxiliary calculation block Calc *u* shown in Figure 3.1. The added benefit of using the existing PID block is that changing the mode to "Auto" and the mode attribute to "Operator" will revert to the previous control scheme, which is a huge advantage during commissioning and tuning of the RHC.

Initially y is filtered using a Leadlag block in order to prevent noise from introducing instability. Next a Deadtime block is used to delay the filtered y by 20 seconds. Depending on the magnitude of the noise, this delay may be increased or decreased in order to allow proper filtering of the signal. In the Calc ROC auxiliary calculation block the filtered and delayed value from the deadtime block is subtracted from the current filtered value of y from the LeadLag block. This result is multiplied by 3 to obtain the predicted rate of change of y in minutes as:

$$ROC(y) = \mathcal{L}^{-1}\{3Y(s)H(s)(1-exp(-20s)\} \tag{3.4}$$

where $H(s) = (s+z)/(s+p)$ is the leadlag filter and $\mathcal{L}^{-1}$ is the inverse Laplace transform. If a different value for the delay was chosen, that value should also be used in (3.4).

Auxiliary calculation block Calc RHVal calculates $y_{k+T_{RH}}$, as shown in (3.1). Auxiliary calculation block Calc u calculates the new value for Δu_{RH} as shown in (3.3). This value is added to the current value of u_{RH} if $y_{k+T_{RH}}$ is outside the limits. If $y_{k+T_{RH}}$ is between the limits, the previous value of u_{RH} is maintained. If $y_{k+T_{RH}}$ is outside the limits, the new value for u_{RH} will change the slope of y just enough to prevent $y_{k+T_{RH}}$ going over the active limit.

3.2.1.2 Tuning RHC

There are two tuning parameters for RHC. The first is the process gain k' as explained in Section 3.2.1. Although this value may be changed and have an impact on controller performance, it is not recommended, as it represents the response of the process to changes in u. Should the process or process conditions change so as to influence the process gain, then only should the new gain be determined by calculation or step test.

This leaves only T_{RH} as a tuning handle. If a large value for T_{RH} is chosen, the time during which the level my not exceed a limit is extended further into the future. A small predicted rate of change for y will result in the calculations indicating that a limit will be exceeded within the ramp horizon. If a smaller value for T_{RH} is chosen, the same rate of change for y will not exceed the limits within the ramp horizon, and therefore no moves will be made. This may lead to the deduction that a smaller value of T_{RH} is better as less control moves will be made. However, using a smaller value of T_{RH} will force the controller to make larger control moves once a limit violation is predicted. Because RHC will now start moving later, it will need to make larger moves in order to prevent a limit violation.

Tuning RHC therefore comes down to playing off not moving at all with making moves that are larger.

Most engineers will assume a starting value for T_{RH} based on a visual inspection of the data. Factors that need consideration is the magnitude of disturbances, the capacity of the vessel and the range of the MV. A visual inspection of the rate of change of y during upsets can be used to determine what value of T_{RH} will allow the same rate of change in y initially. It is better to start with a larger value of T_{RH} in order to let the controller start making moves earlier as this is more conservative. As the engineer and operations personnel get more comfortable with the RHC performance, the value can be decreased so that RHC will move less, improving the controller performance.

3.2.2 Integral gap control

The tuning of PID controllers in most modern DCS can be changed in real time using function blocks or code based on process or control states. Similar to commonly used proportional gap control (King, 2016), IGC (Gous and de Vaal, 2021) uses this facility to change the integral control action. A gap is defined around y^{SP} of a PID controller where the integral action is decreased when y is inside the gap. The gap is defined as:

$$Gap = \bar{y} - \underline{y} \tag{3.5}$$

where $\bar{y}$ and $\underline{y}$ are the user defined upper and lower limits. The gap is usually of the same magnitude on both sides of y^{SP}, but it does not have to be. The PID controller then uses less integral action when y is close to y^{SP} (inside the gap) and more integral action when y is further away from y^{SP} (outside the gap). As with PI gap control, this can be implemented by defining a ratio:

$$\tau_{i(r)} = \frac{\tau_{i(in\ gap)}}{\tau_{i(outside\ gap)}}, \tag{3.6}$$

where $\tau_{i(r)}$ the ratio between the integral tuning constant used when y is inside the gap ($\tau_{i(in\ gap)}$) and the integral tuning constant used when y is outside the gap ($\tau_{i(outside\ gap)}$).

IGC can be defined as:

$$\Delta u_k = \begin{cases} K_c \left[(e_k - e_{k-1}) + \frac{t_s}{\tau_{i(in\ gap)}} e_k \right] & \text{if } \underline{y} < y < \bar{y}, \\ K_c \left[(e_k - e_{k-1}) + \frac{t_s}{\tau_{i(outside\ gap)}} e_k \right] & \text{otherwise.} \end{cases} \tag{3.7}$$

IGC enables more aggressive tuning when y is close to the limits while allowing the controller to slow down the approach to y^{SP} enough to prevent overshoot. Reducing the integral control action when y is close to y^{SP}, while taking more aggressive integral action when y is farther from y^{SP}, mitigates the potential overshoot when y is returning to y^{SP}.

When y is moving towards y^{SP} from outside of the gap, proportional control will work against the

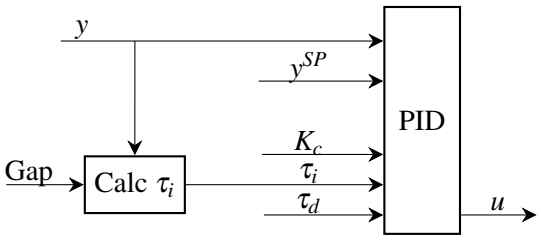

Figure 3.2. Implementation IGC on a DCS.

integral action, slowing down the return to y^{SP}. As the integral action is still tuned aggressively it should overshadow the proportional action, causing y to quickly return to y^{SP}.

Once y moves back into the gap, decreased integral action will ensure that y will not overshoot y^{SP}. Only small controller moves will be made while y is inside the gap.

By setting the proportional gain, the integral tuning and the gap correctly, IGC will be able to:

- avoid overshoot,
- more aggressively return y to y^{SP} while the error is large, and
- slow down the controller response when the error is small.

3.2.2.1 IGC implementation

The implementation of the IGC on a Honeywell Experion DCS as shown in Figure 3.2 shows how simple it is to deploy on a modern DCS. Using an auxiliary calculation block (Calc τ_i), with an If statement, the auxiliary calculation block determines if the absolute value of y minus y^{SP} is smaller than the gap. If so, the level is inside the gap and $\tau_{i(gap)}$ is sent as integral tuning parameter to the PID controller, otherwise, τ_i is used.

3.2.2.2 Tuning IGC

The tuning rules for using a PI controller for averaging level control are discussed in Section 2.4.1.2. These are empirical rules that have been proven effective by use in industry (King, 2016), as well as being shown to be effective in simulations. Using these rules as is to tune IGC will cause it to make larger than necessary moves on u, and it will not allow y to move all the way to the maximum desired deviation.

After a disturbance, the error is small, as the disturbance occurred recently and y has only started moving away from y_{SP}. During this stage, the impact of integral action is also small, as it is based on

$1/\tau_i$ and as y should still be in the gap, $\tau_{i(in\ gap)}$ will have a large value. Later, y will move to outside of the gap and at this stage, increased integral action, coupled with a larger error, will make the control much more aggressive. The increased total controller action while the error is large has the effect that using K_c as calculated in (2.5) will result in y not reaching the desired limit. Depending on $\tau_{i(outside\ gap)}$ and $\tau_{i(in\ gap)}$ values chosen, the increased control action when the error is large should turn the level before it reaches the limit, providing a control solution that will not let y move over the full range between limits.

The choice of gap size, the desired maximum deviation, $\tau_{i(in\ gap)}$, and $\tau_{i(outside\ gap)}$ will determine the duration for which the less aggressive $\tau_{i(in\ gap)}$ and the more aggressive $\tau_{i(outside\ gap)}$ are applied. Consequently, all these variables influence the maximum deviation. In order to determine which values should be used for the tuning variables, it is possible to simulate the system to find the maximum deviation that will result from the choice of tuning variables. Because (2.1) takes proportional action, based on the error, a multiplicative factor can then be calculated to determine by how much K_c should be decreased to allow y to reach the limit. A relationship then has to be found between the tuning values chosen and the amount by which K_c has to be adjusted.

Three differently sized drums were simulated using the simulator shown in Figure 4.2 and data collected to find reasonable tuning values for IGC. Different values for the desired deviation, gap and estimated maximum disturbances were used in subsequent simulations. When using the simulation to obtain tuning parameters, only one step disturbance was introduced at the beginning of the simulation in order to capture the response when the assumed maximum disturbance occurs. The simulation is then allowed to run until the rate of change of y is zero, at which time the actual maximum deviation is captured.

PI tuning values are obtained by using the tuning values for a normal PI controller as shown in Section 2.4.1.2. An integral factor τ_f is defined to determine the difference between $\tau_{i(in\ gap)}$ and $\tau_{i(outside\ gap)}$. τ_f is used as follows: $\tau_{i(in\ gap)} = \tau_{PI}\tau_f$ and $\tau_{i(outside\ gap)} = \tau_{PI}/\tau_f$. A gap to deviation ratio is defined as $R_{gap} = Gap/d$.

The impact of the estimated maximum disturbances was simulated to determine the actual measured deviation of y from y^{SP}, which was compared to the desired maximum deviation (d). Because the action of the controller is proportional over the range of y, all that remains is to find a factor that can

be applied to $K_{c(PI)}$ in order to allow y to move closer to the desired maximum deviation. This factor will have to be less than 1 to move u less and to allow y to move farther away from y^{SP}.

A fraction of actual deviation divided by desired deviation is calculated. If the value of $K_{c(PI)}$ is then multiplied by this fraction, the controller will let y move to the desired limit. In order to find an equation that will provide the change required in K_c, simulations were run with different values of τ_f and R_{gap}. The simulations were done using values of 2, 3 and 4 for τ_f, different size disturbances and varying gap/desired maximum deviation values.

The data from the simulations are shown in Table 3.1 and in Figure 3.3. This figure shows three data series that represent τ_f equal to 2, 3 and 4. From the trends it is clear that a second order polynomial relationship between R_{gap} and actual deviation / desired deviation exists.

Trail and error showed that a linear relationship between τ_f and actual deviation / desired deviation provides satisfactory results.

K_c for IGC can then be calculated as:

$$K_c = \frac{K_{c(PI)}}{(0.9+0.04\tau_f)(1.09+0.13R_{gap}-0.26R_{gap}^2)} \tag{3.8}$$

In order to tune IGC:

- use data to determine what the maximum expected disturbance is,
- use the alarm and/or trip limits as well as discussions with operators to determine the maximum deviation,
- use the tuning rules in (2.5) to calculate $K_{c(PI)}$ and $T_{i(PI)}$,
- choose a value for τ_f, based on how large the difference between $\tau_{i(in\ gap)}$ and $\tau_{i(outside\ gap)}$ should be,
- choose the gap, base the decision on what fraction of the maximum deviation should be less aggressive and what fraction should be more aggressive,
- calculate $\tau_{i(in\ gap)} = \tau_{PI}\tau_f$ and $\tau_i = \tau_{PI}/\tau_f$, and
- calculate K_c using (3.8).

Table 3.1. Data used to develop (3.8).

V (m^3)	τ_f	$K_{c(PI)}$	$\tau_{i(PI)}$	Disturbance (m^3/h)	Gap (%)	Desired	Actual
6.82	2	0.13333	196.4	5	5	30	26.86
6.82	2	0.13333	196.4	5	10	30	27.23
6.82	2	0.13333	196.4	5	15	30	27.45
6.82	2	0.13333	196.4	5	20	30	28.13
3.45	2	0.16	82.8	6	5	30	27.12
3.45	2	0.16	82.8	6	10	30	27.32
3.45	2	0.16	82.8	6	15	30	27.72
3.45	2	0.16	82.8	6	20	30	28.37
3.45	3	0.16	82.8	6	5	30	25.22
3.45	3	0.16	82.8	6	10	30	25.54
3.45	3	0.16	82.8	6	15	30	26.16
3.45	3	0.16	82.8	6	20	30	27.22
3.45	3	0.16	82.8	6	25	30	28.8
3.45	4	0.16	82.8	6	5	30	23.8
3.45	4	0.16	82.8	6	10	30	24.23
3.45	4	0.16	82.8	6	15	30	25
3.45	4	0.16	82.8	6	20	30	26.35
3.45	4	0.16	82.8	6	25	30	27.7
13.866	2	0.224	237.7	7	5	25	22.44
13.866	2	0.224	237.7	7	10	25	22.7
13.866	2	0.224	237.7	7	15	25	23.22
13.866	2	0.224	237.7	7	20	25	24.16
13.866	3	0.224	237.7	7	5	25	20.84
13.866	3	0.224	237.7	7	10	25	21.25
13.866	3	0.224	237.7	7	15	25	22.16
13.866	3	0.224	237.7	7	20	25	23.52

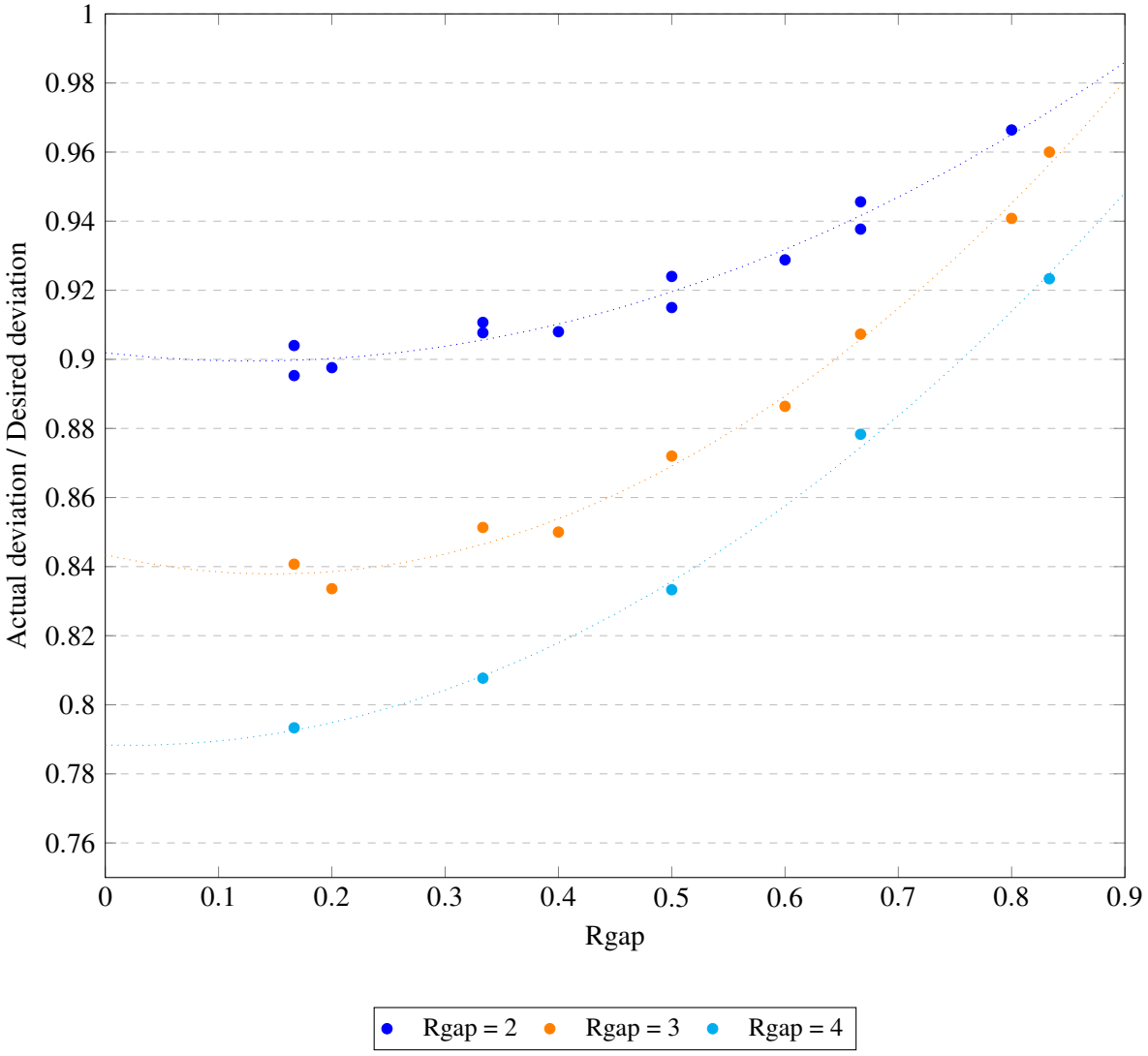

Figure 3.3. Impact of gap ratio on deviation.

In order to minimise incidents, when tuning IGC the typical controller tuning guidelines should be adhered to:

- Before changing tuning parameters, make a note of y^{SP}, u, K_c, τ_i, t_s. This is a precautionary measure, as tuning often takes an unexpected turn. If the controller becomes unstable, switch the control loop to manual, change u to the previous value, revert to the previous values of K_c and τ_i. Allow the process to stabilise before attempting tuning changes again.
- When increasing K_c, do not increase or decrease the value of K_c by more than 10 % at a time. After making these changes allow ample time to evaluate the impact of the change before increasing or decreasing the value again. This is because errors are sometimes made when calculating tuning values, and increasing the value of K_c to the suggested value in one step may

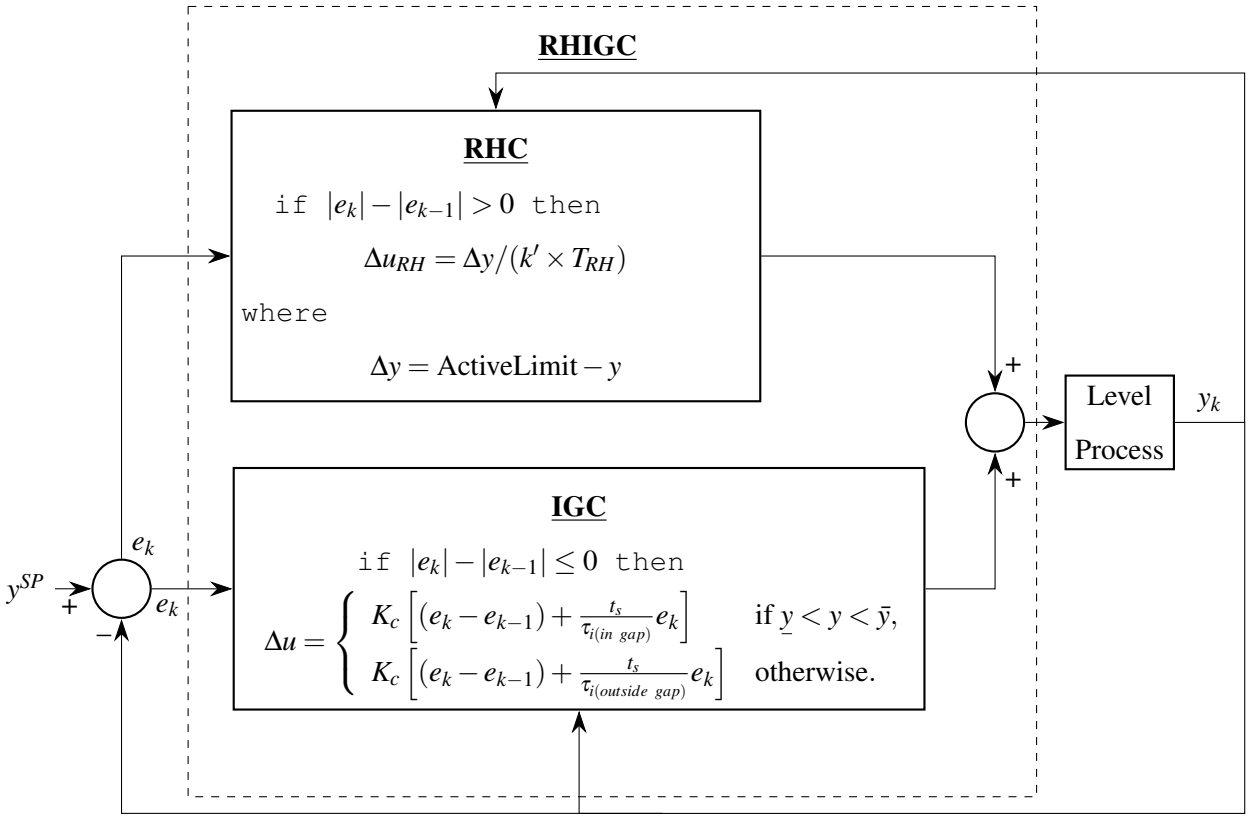

Figure 3.4. RHIGC.

end badly.

- When changing the value of τ_i the value may be changed at bigger increments, as this parameter has a smaller impact on the system. The rule of thumb is to not decrease or increase τ_i more than 25 % at one time. As with changes in K_c, allow enough time to evaluate the impact of the change.
- Make constant notes regarding changes made, as well as the time at which these changes have been made. This is to ensure that changes are not made too quickly, as well as knowing to which values to revert, should this be required.

3.2.3 Combining RHC and IGC

This section details the design of the RHIGC (Gous et al., 2023) which combines RHC and IGC. In particular, RHIGC benefits from the fact that RHC make no assumption regarding the maximum disturbance during tuning, and that it does not make any changes to u if there is little threat of y going over limits. The IGC part of RHIGC compensates for the main disadvantage of RHC, which does not return the level to y^{SP} after a disturbance.

An advantage of RHIGC over advanced control methods such as MPC, is that it can be implemented on a DCS or PLC without the need for additional computer and network hardware and software, plant tests and model identification.

RHIGC works by switching between RHC and IGC depending on which controller is best suited to current process conditions. Switching is dependent on whether the error is increasing or decreasing. RHC is used when the error is increasing, i.e., when the current value of the error has the same sign as the rate of change of the level. RHIGC is switched to the IGC part when the error is zero or decreases, i.e., the rate of change of the level becomes zero or while y is moving back towards y^{SP}.

RHC makes the smallest moves possible while the level is moving away from y^{SP} and will prevent y from going outside its limits. IGC will bring y back towards y^{SP} after RHC has rejected the disturbance and balanced y. IGC will move u more aggressively when y is close to the limits, where there is a larger risk that RHC will not be able to keep the level within limits should another disturbance occur. Later, IGC will make smaller moves when y is inside the gap, improving the averaging level control performance and ensuring that y does not overshoot y^{SP}.

It is possible to switch between different inputs and outputs in DCS based controllers (Reyes-Lúa and Skogestad, 2019). When y is moving away from y^{SP}, the error increases and standard DCS functions are used to change the controller to manual mode in order to implement the control moves calculated by the RHC part of RHIGC. This is done by using standard DCS functions to switch the IGC controller to manual and writing the RHC u values directly to the output of the controller. When y is moving towards y^{SP} or stays constant, the error correspondingly decreases or remains constant and standard DCS functions switch the IGC controller from manual to auto mode, enabling IGC to output its own calculated u value. This is done to enable seamless switching between the two parts of the controller.

When determining whether y is approaching or moving away from y^{SP}, a first-order filtered value of y is used to prevent frequent switching between the IGC and RHC parts of the controller.

3.2.4 Simplified optimal averaging level controller

While optimal averaging level control (McDonald et al., 1986) requires MPC software running on a dedicated server, it may be simplified using the same approach as used with the RHC.

If the dynamics of the level is ignored and only the absolute value of the level and its current rate of change (*ROC*) is considered, a simplified version of the Optimal Averaging Level Controller, called the Simplified Optimal Averaging Level Controller (SOALC) may be developed (Gous et al., 2021). If the current rate of change of the level is extrapolated from the current value of the level, limit violations can be predicted. A controller move size u, that must be implemented during every execution cycle until the level is balanced at the limit, can then be calculated. The sum of the controller moves until the level is balanced at the limit will then be $u.k$, where k is the number of execution cycles until the limit is reached.

Because dynamics are ignored, this approach will not be as accurate as a model based controller but the calculations are simple enough to implement on a standard DCS while feedback can be used to ensure adherence to limits.

If a level is currently at y_0, with a positive rate of change of $(\frac{dy}{dt})_0$, and a high limit $\bar{y}$ that will be violated based on extrapolation of $(\frac{dy}{dt})_0$, then a sequence of k moves $(u_1 ... u_k)$ must be calculated to balance the level at $\bar{y}$. The total movement of u over the control horizon must be $\delta u = u.k$. If a controller gain (K_G) is defined as the change per execution cycle in the rate of change of the level brought about by increasing u by 1% then $\delta u = (\frac{dy}{dt})_0 / K_G$.

Combining these equations yields $u = (\frac{dy}{dt})_0 / (K_G k)$

After a disturbance the initial conditions for SOALC will be $y = y_0$ and $\frac{dy}{dt} = (\frac{dy}{dt})_0$. The final values are reached when the level is at the limit and the rate of change is zero, $y = \bar{y}$ and $\frac{dy}{dt} = 0$.

If a step of size u is made during every execution cycle then $\frac{dy}{dt}$ will be decreased by $u.K_G$ during every execution cycle. The level will also increase by the current rate of change per execution cycle. Therefore Table 3.2 can be calculated and used to calculate the value of y ans well as the rate of change of y at execution cycle n.

In Table 3.2, the first column shows the execution cycle. The second column shows the level measurement, calculated as the previous value of the level plus the change in the level. The change in the level will be the rate of change in the level during the previous execution cycle as shown in column 3.

Table 3.2. Change in level and rate of change over the control horizon.

Execution cycle	Level	Rate of change of level
0	y_0	$(\frac{dy}{dt})_0$
1	$y_0+(\frac{dy}{dt})_0$	$(\frac{dy}{dt})_0-u.K_G$
2	$y_0+2(\frac{dy}{dt})_0-u.K_G$	$(\frac{dy}{dt})_0-2.u.K_G$
3	$y_0+3(\frac{dy}{dt})_0-3.u.K_G$	$(\frac{dy}{dt})_0-3.u.K_G$
4	$y_0+4(\frac{dy}{dt})_0-6.u.K_G$	$(\frac{dy}{dt})_0-4.u.K_G$
5	$y_0+5(\frac{dy}{dt})_0-10.u.K_G$	$(\frac{dy}{dt})_0-5.u.K_G$
n	$y_0+n(\frac{dy}{dt})_0-(0.5.n^2-0.5n)u.K_G$	$(\frac{dy}{dt})_0-n.u.K_G$

As example, during execution cycle 3, the previous value of the level was calculated as $y_0+(\frac{dy}{dt})_0+(\frac{dy}{dt})_0-u.K_G$. If the change in level, seen in column 3 as $(\frac{dy}{dt})_0-2.u.K_G$ is added, the value for the level at cycle 3 is calculated as $y_0+(\frac{dy}{dt})_0+(\frac{dy}{dt})_0-u.K_G+(\frac{dy}{dt})_0-2.u.K_G$ or $y_0+3.(\frac{dy}{dt})_0-3.u.K_G$.

At execution cycle k the controller has made the last move of size u, which will cause the rate of change of the level to become 0, at $\bar{y}$. Therefore, at row k in Table 3.2, the value of the level is $\bar{y}$ and the rate of change is 0.

Because we want to balance the level at the limit, the value of y at execution interval k must be $\bar{y}$. In calculating the equation in column 2, row n of Table 3.2, it is clear that the coefficient of y_0 should be 1 and the coefficient of $(\frac{dy}{dt})_0$ should be n. The series of coefficients for the term $u.K_G$ (0,1,3,6,10,....) shown in column 3 can be calculated as $0.5n^2-0.5n$.

Using $\bar{y}=y_0+k.(\frac{dy}{dt})_0-(0.5k^2-0.5k)uK_G$ yields

$$u=0.5(\frac{dy}{dt})_0^2/(K_G(\bar{y}-y_0-0.5(\frac{dy}{dt})_0)) \tag{3.9}$$

(3.9) calculates by how much u must be changed during every execution cycle to balance the level at the high limit. As stated before, dynamics on the level model may cause inaccuracies but as the value of the level and the rate of change of the level is measured every execution cycle, feedback is introduced.

Because SOALC will let the level go up to limits while making the smallest control moves to ensure the limits are not breached, the risk remains that disturbances that push the level in the same direction as before may occur once the level has reached the limit. f this occurs, SOALC will not restore the level to the breached limit using the same algorithm. Instead, the algorithm that regulates the level between limits will adjust u in the wrong direction, exacerbating the disturbance that caused the level to exceed its limits. When this happens, the error will grow and SOALC will accelerate u in the wrong direction.

Just multiplying the calculated change in u by -1 if y is outside a limit sounds like a simple solution to SOALC moving u in the wrong direction. However, if this is done, once the movement of the level away from the limit is halted, the rate of change will be zero and SOALC will stop making changes. This will let the level remain outside the desired limits until another disturbance occurs. Therefore another algorithm like PI or RHC needs to be applied if the level is outside its limits.

3.2.5 Double step disturbance

Certain controllers, such as the P-only and RHC will allow the level to move to its set limits after a disturbance, without attempting to return the level to setpoint. This leaves the system vulnerable to breaching limits, should a consecutive disturbance in the same direction occur. This effect is not shown clearly by the disturbances typically used in literature such as the single step, the random walk or a cycle.

3.3 STICTION COMPENSATION

3.3.1 Time-based stiction compensation

The aim of time-based stiction compensation (Gous et al., In Press) is to keep the valve at the desired position u (expressed as a percentage) on average for a period of time instead of getting the valve to the exact desired position at each time instance.

A stiction compensation interval is defined as:

$$t_{CI} = t_1 + t_2. \tag{3.10}$$

The valve is set at a high value u_{Hi} for a period of time t_1 and to a low value u_{Lo} for a period of time t_2. The desired value of u over t_{CI} is the average:

$$\bar{u} = \frac{t_1 u_{Hi} + t_2 u_{Lo}}{t_{CI}}. \tag{3.11}$$

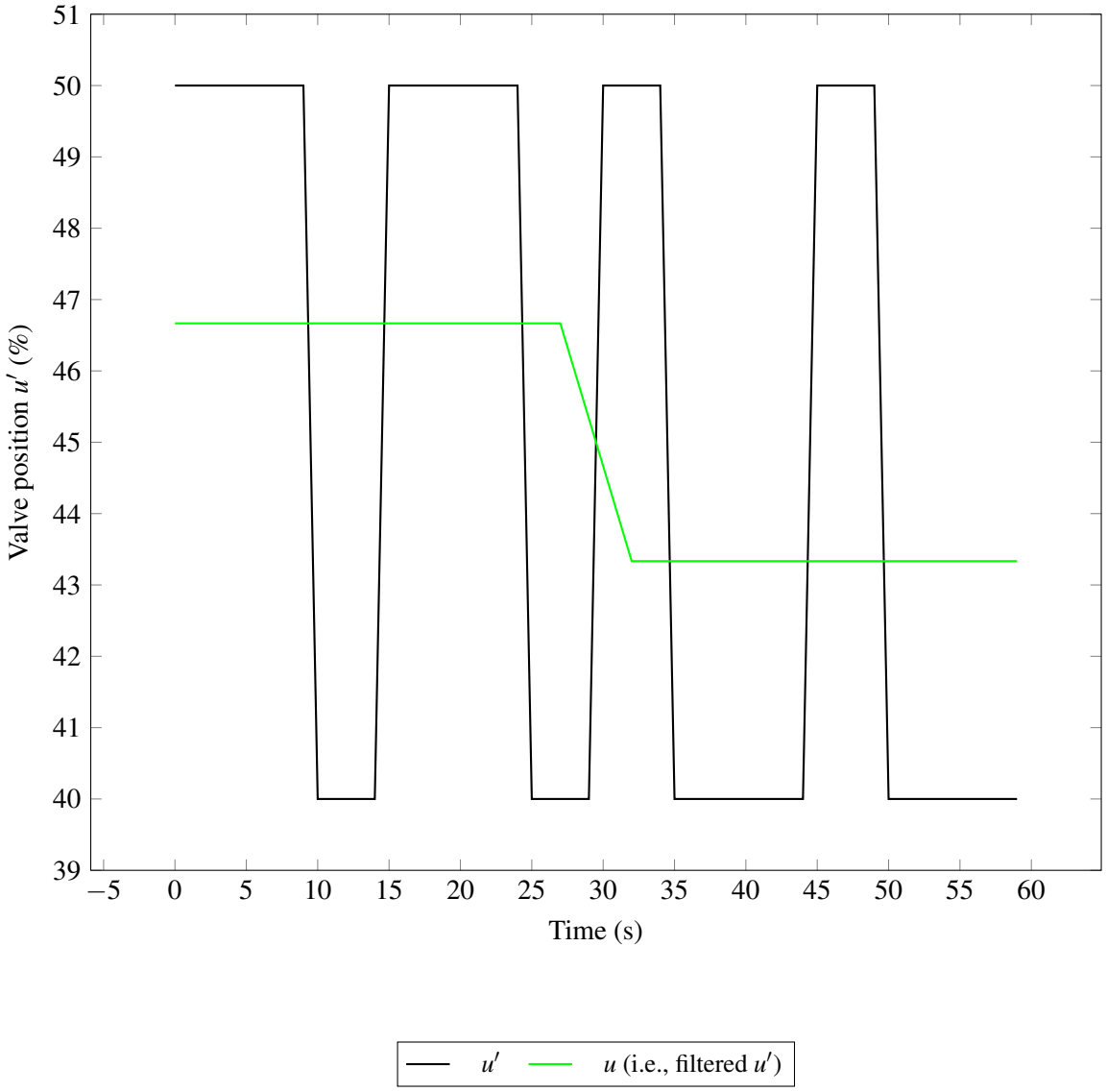

Figure 3.5. Time-based valve position compensation.

This approach is illustrated graphically in Figure 3.5, showing how to overcome $SB \leq 10$ % to step u from 46.7% to 43.3%. The valve is set to $u_{Hi} = 50$ % for $t_1 = 10$ s and $u_{Lo} = 40$ % for $t_2 = 5$ s to maintain the valve on average at $\bar{u} = 46.7$ % over the period $t_{CI} = 15$ s. At $t = 30$ s, the valve is set to $u_{Hi} = 50$ % for $t_1 = 5$ s and $u_{Lo} = 40$ % for $t_2 = 10$ s to maintain the valve on average at $\bar{u} = 43.3$ %. The green line in Figure 3.5 is the result of a central averaging filter with a span of 15 s.

Consider the following example. A control system attempts to control the level of a drum where a PI level controller writes directly to a valve on the outlet from the drum. The desired value of u is at 45%, the valve is currently at 40% and $SB = 20$ %. If the value of u is written to the valve, nothing

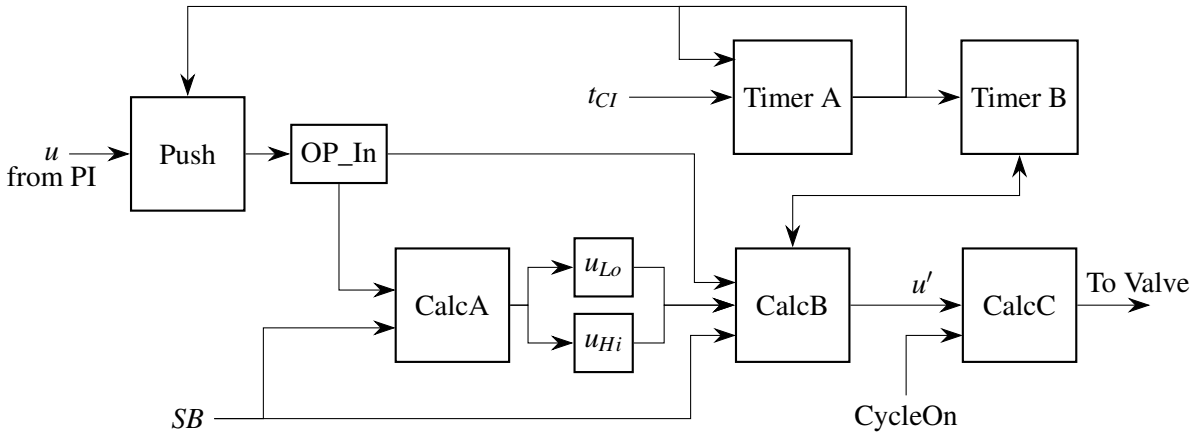

Figure 3.6. Implementation of time-based stiction compensation on a DCS.

will happen because the 5% that the valve is required to move is less than SB. Instead the valve can be moved to $u_{Hi} = 60$ %, which will overcome the stiction. If the valve is kept at $u_{Hi} = 60$ % for $t_A = 0.25t_{CI}$, and at $u_{Lo} = 40$ % for $t_B = 0.75t_{CI}$, the average value for t_{CI} will be $\bar{u} = 45$ %.

For a reasonably long t_{CI}, the average valve position $\bar{u} = 45$ % should keep the average of the level at setpoint y_{SP}. However, if t_{CI} is long, the level will may move further away from y_{SP} during the interval t_{CI} than what operators are comfortable with. Should a shorter t_{CI} be chosen, the stiction compensation will move the valve too frequently and increase stress on an already broken system. Operators will also not be comfortable with a valve that moves too frequently. It is recommended that t_{CI} be chosen such that operators are comfortable with the difference between the highest and lowest values of the level during each interval t_{CI}.

In the case of a short t_{CI}, if the t_{CI} approaches the execution interval of the DCS, granularity is introduced. For instance, if the PI controller as well as the stiction compensation executes at 1 s, and a t_{CI} of 5 s is chosen, the minimum time that can be spent at $\bar{u}$ will be one fifth of t_{CI}, and the resolution of u' becomes an issue.

3.3.2 Implementation of time-based stiction compensation on a DCS

The deployment of the time-based compensation algorithm on a DCS is shown in Figure 3.6. The algorithm in Figure 3.6 uses two timers. The first timer, Timer A, starts the compensation interval by counting down t_{CI} chosen by the engineer. When the timer starts, the desired value of u is read from the PI controller. This value of u will be used for the entirety of the current cycle, and will only be

updated at the next compensation interval. This is done by the block called 'Push', which pushes the desired value of u to numeric block 'OP_In' when Timer A has counted down.

Calculation block A determines the necessary high and low u' values based on the desired average u':

$$u_{Hi} = SB\left(\left\lfloor \frac{u+SB}{SB} \right\rfloor\right) \tag{3.12}$$

$$u_{Lo} = u_{Hi} - SB. \tag{3.13}$$

Calculation block B determines the time to keep u' at u_{Hi}:

$$t_1 = \left\lfloor \frac{u_{Hi-u}}{SB} t_{CI} \right\rfloor. \tag{3.14}$$

The value t_1 is sent to Timer B. While Timer B is counting down, calculation block B will pass u_{Hi} to calculation block C. When timer B stops, calculation block B passes u_{Lo} to calculation block C. Calculation block C passes the value it receives to the valve. When Timer A has completed, it resets itself and Timer B to start a new control interval.

CycleOn is used as a switch to enable or disable the stiction compensation. If it is set to 1, calculation block C will pass its most recently received value (either u_{Lo} or u_{Hi}) to the valve, otherwise it will send the unaltered value of u from the PI controller to the valve.

3.3.3 Time-based stiction compensation with move suppression

De Souza L. Cuadros et al. (2012) show that the effect of stiction can be reduced by only implementing changes in u when absolutely necessary. Time-based stiction compensation, which moves the control valve up and down with a magnitude greater than SB every cycle, can be combined with the part of ramp horizon control (RHC) that determines whether an adjustment in u is required in order to prevent y moving over a limit before T_{RH} minutes (Gous et al., 2021, 2023). A high and low limit is chosen, similar to RHC, that must not be violated before T_{RH} minutes into the future. Because y should stay equally close to y^{SP} on either side, the low and high limits should be equal distances from y^{SP}, therefore, it is easier to implement this as a gap around y^{SP}. If y is predicted to be outside the gap at time T_{RH}, the changes in u from the stiction compensation algorithm will be implemented, otherwise, u is held constant.

Figure 3.7 indicates the DCS implementation of RHC to augment Figure 3.6 to achieve time-based stiction compensation with move suppression. T_{RH} and Gap in Figure 3.7 are inputs given by the

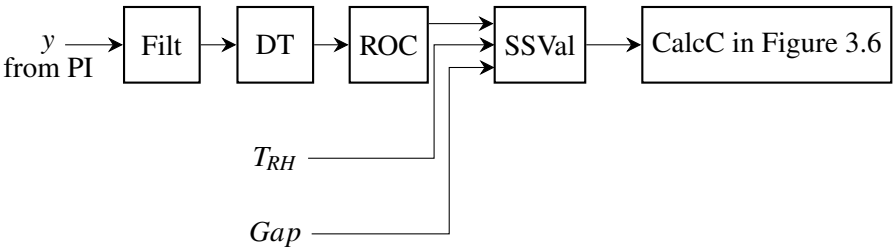

Figure 3.7. Implementation of move suppression on a DCS.

control engineer, with T_{RH} indicating the Ramp Horizon, during which time the extrapolation of y's trajectory may not move outside of the gap.

To suppress moves, the value of y is filtered to minimise the impact of noise using a standard filter block (Filt in Figure 3.7) . This value is passed through a deadtime block (DT) of 1 minute and calculation block ROC then calculates the current rate of change of y (δy). Block SSVal calculates the extrapolated value of y at time T_{RH} as $y + \delta y T_{RH}$. If this calculation determines that y will be outside the allowed gap at time T_{RH}, then the output of SSVal will instruct calculation block C in Figure 3.6 to implement the value of u', otherwise, u' is not changed.

Time-based stiction compensation with move suppression can be seen as a 'close enough' strategy, where moves are only implemented when it is indicated that y will go out of bounds before the ramp horizon. In this way not only is y kept between limits, but valve movement is also minimised.

3.4 CHAPTER SUMMARY

In order to test if controllers that run natively on a standard DCS are able to compete in terms of averaging level control performance four novel control techniques for averaging level control were developed. RHC, IGC, SOALC and RHIGC execute natively on a standard DCS at execution cycles that are much faster than typical model based controllers. The techniques are described in Gous et al. (2021) and Gous et al. (2023).

Two stiction compensation techniques were developed, called time-based stiction compensation and time-based stiction compensation with move suppression. The objective of time-based stiction compensation is to get a system with stiction to closely resemble the same system without stiction, while time-based stiction compensation will attempt to keep y close enough to y^{SP} while minimising

valve movement. Both techniques can run on a standard modern DCS without need for additional hardware or software. The techniques are described in (Gous et al., In Press).

CHAPTER 4 SIMULATION STUDIES

4.1 CHAPTER OVERIEW

In this chapter the performance of of the four level averaging controllers discussed in Chapter 3, i.e. RHC, IGC, SOALC and RHIGC are compared in simulation with a standard PI controller. They were all deployed on a standard DCS and used to control a level on a feed drum. The response of the level of the feed drum is also simulated using standard DCS functionality. The different ways that the four controllers respond to two consecutive step disturbances in the same direction is discussed to illustrate the difference in the algorithms. RHIGC is then compared to MPC in a 24-hour real-time simulation using plant data as disturbance. The metrics show that RHIGC and MPC performance are similar. The difference of the cost of deployment of the controllers, as well as the need for additional hardware, software and communication protocols for MPC also needs to be considered.

The two stiction compensation methods are simulated and the results compared to the original system with and without stiction, as well as an attempt to overcome stiction by using more aggressive tuning.

4.2 AVERAGING LEVEL CONTROL

4.2.1 Description of simulation

The difference between PI, RHC, IGC, SOALC and RHIGC is demonstrated in a simulation example of the control of a drum level as shown in Figure 4.1. The process simulation is based on the industrial plant example where RHIGC is implemented, as discussed in Chapter 5. The level controller (LIC) cascades to the flow controller (FIC) which manipulates the flow exiting the vessel. In other words, the level (y) is controlled by manipulating y^{SP} of FIC. The flow into the vessel is treated as a disturbance and is measured by FI.

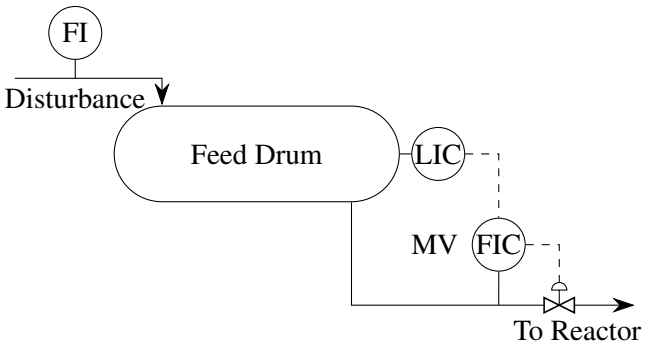

Figure 4.1. Feed drum process flow diagram.

The simulation is conducted on a Honeywell Experion DCS using a typical PI controller as well as IGC, RHC, SOALC and RHIGC controllers to control the feed drum level. The level was simulated in the DCS using standard DCS function blocks as shown in Figure 4.2.

A standard PID block is used and is run in automatic to simulate IGC. To simulate RHC the PID block is run in manual mode and u is set by the function blocks of RHC. The simulation starts with y at y^{SP}, i.e., the level of the tank is at 50%. At the start of the simulation u is also at 50%. Whether u is set by the PID block or the RHC code, the changes in u is subtracted from the initial 50%. This value is then scaled according to the size of the drum by the multiplier block. The multiplier block is actually also a summer block that has a scaling function. The output of the multiplier block is sent to an accumulator block, also called a totaliser block. The output of the accumulator block is passed through a deadtime block and a filter block, which simulates the process dynamics of the level, and then added to the current value of y.

In essence, the simulation scales the value of u to simulate the drum size and uses the deadtime and filter blocks to simulate the dynamics of the level. The accumulator block is used to change the model from self-regulating to an integrator.

At the start of the simulation, the disturbance is at 0. At this time, the output of the summer block will be 0, therefore the output of the multiplier block will also be 0. The totaliser block's output will maintain the previous value of y until the reset flag is set to true. The reset value for the accumulator is set to 50 in order to start the calculation for y at 50% when the simulation is initialised. Initialisation is complete when the output of the deadtime and lag block reaches 50%, at which time the start flag is set to true in order to start the simulation. After initialisation both the level and controller output will

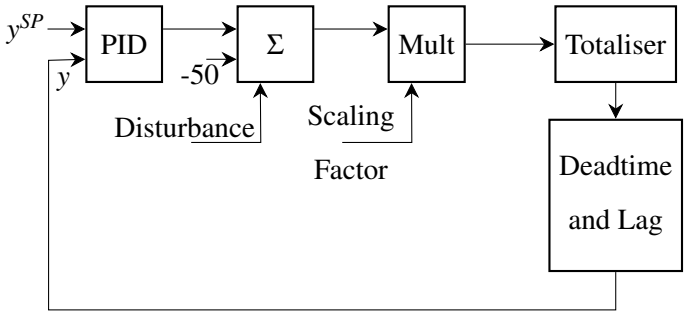

Figure 4.2. PID and drum level simulation.

be steady at 50%. After 3 minutes, the disturbance is introduced and the simulation continues in real time.

The upper hard constraint is a level of 70%. Two consecutive steps of 5 m^3/h each are made, with an interval of 120 minutes between them. The duration of the interval between the two steps is chosen to allow enough time for the controllers to either reduce the rate of change of y to zero, or to start returning y to y^{SP}.

The model of the process in Figure 4.1 as used in the simulation represents the response of the flow setpoint to the drum level:

$$G(s) = -1.43\frac{0.19s+1}{0.54s^2+s}e^{-0s} \tag{4.1}$$

where the units for $G(s)$ are $\frac{level\%}{m^3/h}$.

4.2.2 Simulation results

The simulation was done to show typical responses of the different controllers rather than provide a quantitative analysis. It shows how the PI controller has a typical response, how the IGC can move u faster when the error is large, as well as how it can move u slower when the error is small. It also shows how RHC can prevent the level going over the limit during the first disturbance only and how RHIGC could prevent it during the second disturbance as well, because the IGC part of the controller moves the level away from the limit.

Figure 4.3 shows how the PI controller, IGC, SOALC, RHC and RHIGC responded to the two step changes in the disturbance. The first disturbance in FI of 5 m^3/h upwards occurs at 4 minutes and the second step upwards of 5 m^3/h occurs at 115 minutes.

The PI controller is able to reject the first step disturbance successfully, but fails to keep the level below the high limit of 70% when the second disturbance occurs. This happens because the combined disturbance of 10 m^3/h exceeds the maximum assumed disturbance of 5 m^3/h.

IGC increases its integral control action when the level moves higher than the gap, which is why it goes over the limit by a smaller margin than the PI controller.

RHC is able to keep the deviation over the high limit smaller than both the PI and IGC controllers. However, it does make very large moves during the period that the level is over the limit, and once the level turns, no attempt is made to stop the level from drifting through the range between the high and low limits.

After the first disturbance, SOALC initially makes smaller moves than the other controllers, and as its move size remains constant, its moves are larger than the other controllers further on. Figure 4.3 also shows how SOALC moves u in the wrong direction after the high limit is breached, as well as how this quickly causes a runaway reaction. The reason why SOALC moves u in the wrong direction when y is outside of limits, can be seen in Table 3.2. This can be countered by changing the sign of Δu. Even if this is done, Table 3.2 shows that SOALC will then only balance the level outside of the breached limit, it will not bring it back to between limits. A better solution would be to switch to PI control while the level is in violation of a limit.

Another consideration with SOALC is that it is designed to make the same size adjustments to u after a disturbance. When comparing this to a typical first order response obtained from most controllers, it is clearly not a good solution. A first order response in u will initially make large moves after a disturbance. If there is any error regarding the response, the error will become evident early on in the response, and feedback will be able to address it. If the same move size in u is implemented, smaller initial moves and smaller subsequent error will leave much of the feedback required for later on in the control response, which will lead to poor control. Upon consideration of these issues, it was decided not to test SOALC on a plant.

RHIGC is able to keep the level from going over the high limit. This is possible because the IGC component of the controller moves the level away from the limit once the RHC component of the combined controller brings the rate of change of y to zero. Figure 4.3 also shows how RHIGC starts

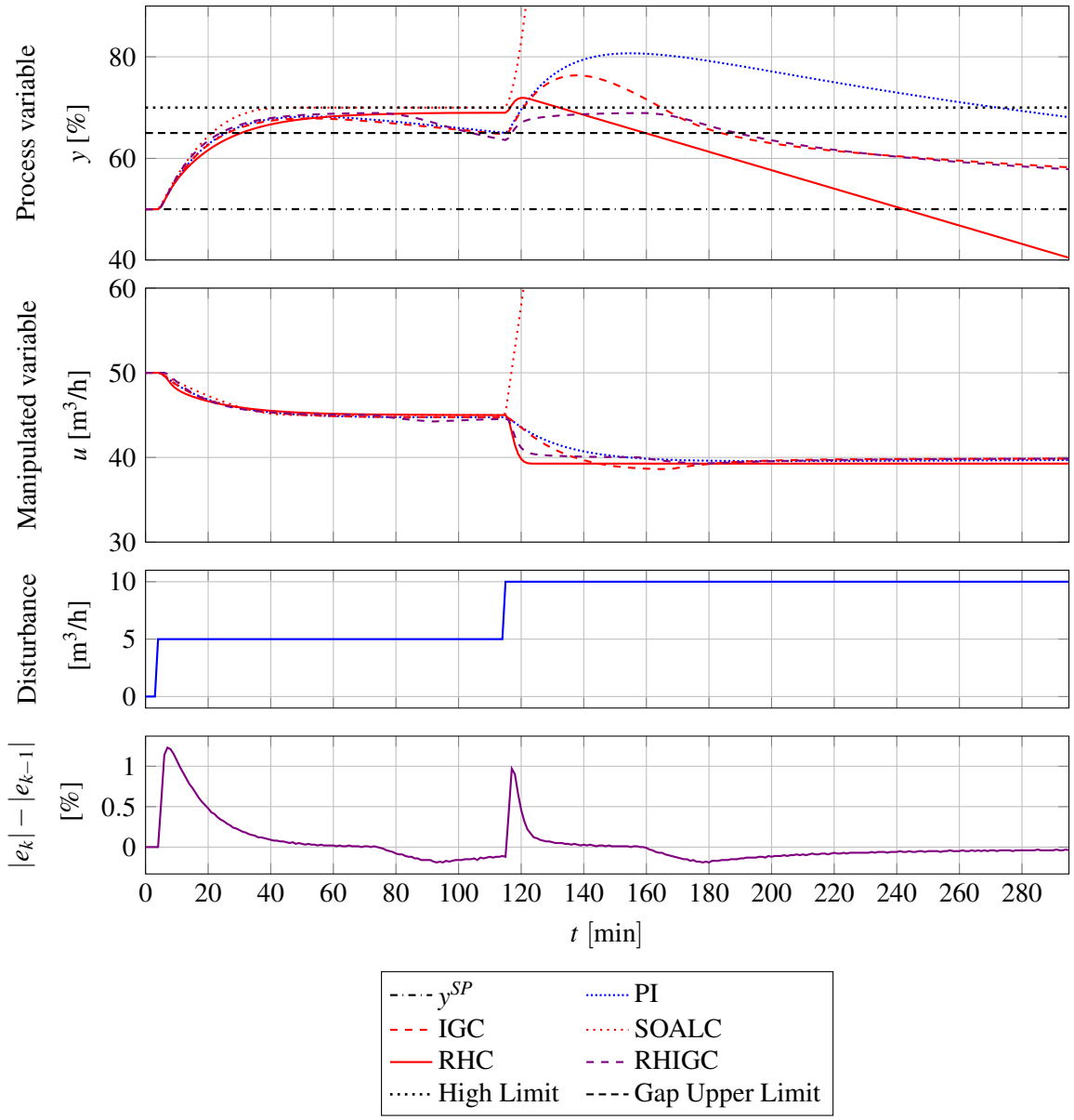

Figure 4.3. Simulated response of the PI, IGC, SOALC, RHC and RHIGC controllers to two step disturbances.

moving u again after the RHC part of to system brought the rate of change of y to zero and the IGC part of RHIGC takes over.

The simulation results clearly illustrate the principle of RHIGC. As can be seen from Figure 4.3, the RHC part of the controller, which is active from $t = 5$ min to $t = 74$ min, as well as from $t = 116$ min to $t = 157$ min where $|e_k| - |e_{k-1}| > 0$, prevents the level from exceeding limits. The IGC part of the controller, which is active from $t = 75$ min to $t = 115$ min, as well as from $t = 158$ min to $t = 295$ min

where $|e_k| - |e_{k-1}| \leq 0$, moves the level away from the limits to prevent consecutive disturbances from forcing the level over a limit.

4.2.3 Comparison with MPC

The simulated plant shown in Figure 4.1 was used to compare the performance of RHIGC against a traditional MPC algorithm. Typical plant data of a stream flowing into a drum on an actual process were used as the disturbance for a 24-hour simulation. The low and high hard level constraints are chosen as 30% and 70% respectively.

The MPC can be represented as:

$$\min_{u_k, u_{k+1}, \ldots, u_{k+N_C-1}} \sum_{j=1}^{N_P} \left(e_{k+j}^2 Q_R + S_{k+j} Q_S\right) + \sum_{j=0}^{N_C-1} \left(\Delta u_{k+j}\right)^2 R \tag{4.2}$$

subject to:

$$x_{k+j} = Ax_{k+j-1} + Bu_{k+j-1} \qquad \forall j = 1, \ldots, N_P \tag{4.3a}$$

$$y_{k+j} = Cx_{k+j} + Du_{k+j} \qquad \forall j = 1, \ldots, N_P \tag{4.3b}$$

$$\underline{y} \leq y_{k+j} \leq \overline{y} \qquad \forall j = 1, \ldots, N_P, \tag{4.3c}$$

$$\underline{u} \leq u_{k+j} \leq \overline{u} \qquad \forall j = 0, \ldots, N_C - 1, \tag{4.3d}$$

$$\Delta\underline{u} \leq \Delta u_{k+j} \leq \Delta\bar{u} \qquad \forall j = 0, \ldots, N_C - 1. \tag{4.3e}$$

N_P is the prediction horizon and N_C is the control horizon. The objective is to keep the level between soft limits with the slack variable weight Q_S and to slowly move the level towards y^{SP} with the penalty Q_R so that sequential disturbances in the same direction can be rejected. Flow stability is influenced through the input move size weight R.

The tank level is modelled as $\Delta y/\Delta t = (Q_{in} - Q_{out})/a$ where a is the area of the drum, Q_{in} is the flow into the drum and Q_{out} is the flow out of the drum. Therefore, using the state-space representation as in (4.3a) and (4.3b), A=0, $B = 1/a$, C=1 and D=0.

Table 4.1. Tuning parameters for comparison of RHIC to MPC.

Controller	Parameter	Value	Units
MPC	N_P	40	min
	N_C	15	min
	Q_R	9	$\%^{-1}$
	Q_S	3	$\%^{-1}$
	R	0.1	$\%^{-1}$
	t_s	60	s
RHIGC	K_c	0.19	$\%^{-1}$
	τ_i	96	min
	$\tau_{i(gap)}$	256	min
	K_G	18.75	$\%^{-1}$
	T_{RH}	900	s
	t_s	1	s

S_{k+j} is a slack variable used to penalise level limit violations before hard constraints are reached and is defined as:

$$S_{k+j} = \begin{cases} y_{sh} - y; & y > y_{sh}, \\ y - y_{sl}; & y < y_{sl}, \\ 0; & y_{sl} \leq y \leq y_{sh}, \end{cases} \tag{4.4}$$

where y_{sl} and y_{sh} are the soft low and high level limits respectively. The soft limits are chosen to be 3% away from the hard limits. $\underline{y}$ and $\bar{y}$ are y low and high limits, $\underline{u}$ and $\bar{u}$ are u low and high limits and $\Delta\underline{u}$ and $\Delta\bar{u}$ are u move size low and high limits respectively. The controller calculates a vector of future control moves up to N_C and the first control move in the vector is implemented at every execution cycle.

Setpoint tracking, soft limits for y and minimizing u move sizes are included as optimisation objectives in (4.3). Process constraints are met using hard constraints for the range of y (4.3c), the range of u (4.3d), and the maximum u move size (4.3e).

The MPC and RHIGC tuning parameters for the simulation are given in Table 4.1. The process model used for the simulation as well as the model used for control by the MPC is shown in (4.1).

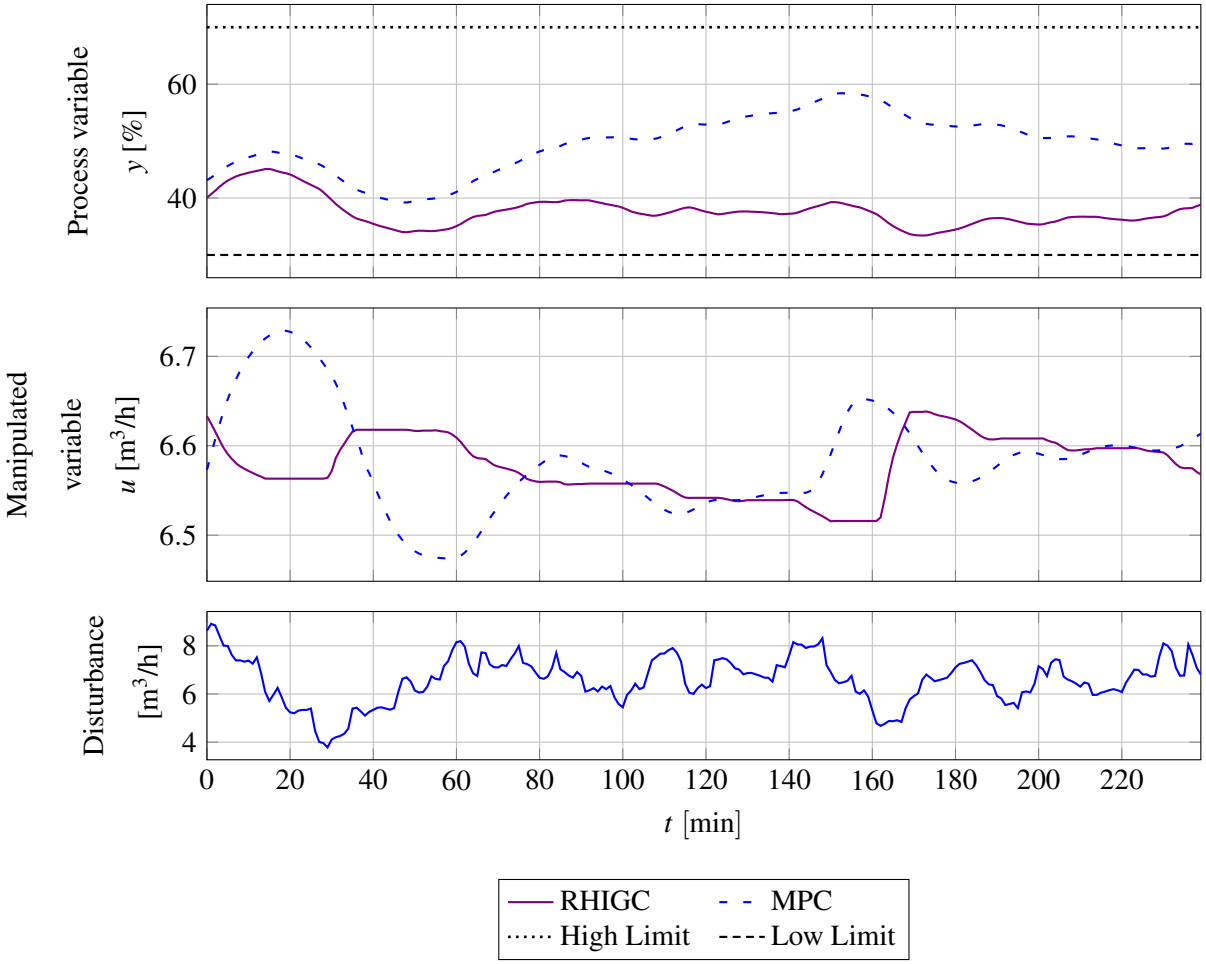

Figure 4.4. Typical RHIGC and MPC behaviour with flow disturbance.

To enable a fair performance comparison, data was sampled at one minute intervals for both controllers, even though t_s is 1 s for RHIGC and t_s is one minute for MPC.

4.2.4 Results of comparison with MPC

The simulated plant shown in Figure 4.1 was used to compare the performance of RHIGC against a traditional MPC algorithm. Typical plant data of a stream flowing into a drum on an actual process were used as the disturbance for a 24-hour simulation. The low and high hard level constraints are chosen as 30% and 70% respectively. The highest value of y in the dataset (y_{high}) as well as the lowest value of y in the dataset (y_{low}) is shown together with σ_u and TV/N in Table 4.2.

The performance comparison of the controllers is shown in Table 4.2 and the typical behaviour of the controllers are shown by displaying data for the last four hours of the 24-hour simulation of each controller in Figure 4.4. The inlet flow disturbance is typical of what can be expected on a real

Table 4.2. Comparison between RHIGC and MPC for the 24-hour simulation.

	y_{high}	y_{low}	σ_u	TV/N
RHIGC	68.0	31.4	0.079	0.039
MPC	68.3	31.4	0.085	0.039

process.

As shown in Table 4.2 the performance of RHIGC and the MPC is similar. The σ_u performance metric indicates that RHIGC has the advantage as it requires less movement in u. The RHIGC has a faster execution interval and a simplified process model running natively on a DCS. The MPC has a slower execution frequency and a higher fidelity model that includes the dynamic model (4.1) running on additional computer hardware and communicates via OPC. Typical MPC project methodology includes step testing, model identification, tuning and commissioning which would also be much more costly than deploying and tuning an RHIGC.

4.3 STICTION COMPENSATION

4.3.1 Simulation results of time-based stiction compensation

A bespoke process simulation was developed to show the impact of valve stiction on a level controller. The feed drum shown in Figure 4.1 was simulated in Honeywell's Experion DCS. The DCS was used to simulate the control action required by using a standard PI controller, as well as the time-based stiction compensation algorithm. The DCS was linked via OPC to the simulator that simulated a valve with stiction. The simulation compared the effect of different stiction bands on the valve as shown in Figure 4.5.

Figure 4.5 shows how the simulated CV y and MV u reacts to a step disturbance to the flow into the drum at time 10 s. The simulation starts with all variables at steady state. The same magnitude step disturbance of 10 m^3/h is used for all simulations. The CV y in the figure shows typical responses of the level to changes in the MV u. Stiction bands of 0%, 5% and 10% are used. The simulated values for y are very similar to what is seen in industry when a valve sticks. Because the valve jumps from a low to a high position and back, typically a sawtooth pattern will emerge.

In industry the turning points of y will normally be smoother due to process and signal filtering. Typical

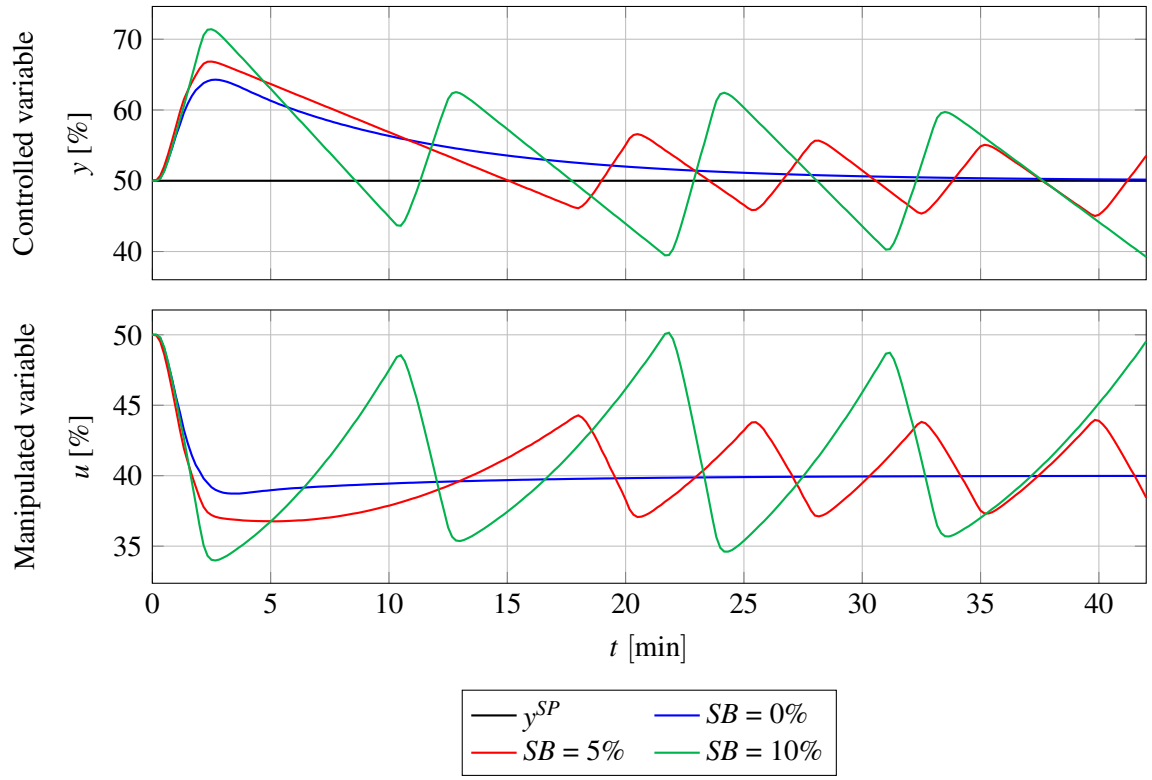

Figure 4.5. Impact of different stiction bands.

cycles in industry will not cycle symmetrically around y^{SP}, but there may be a large offset to either side depending on initial conditions. Because the simulation started with a large disturbance that the PI controller reacted to, initially the average value for the cycles seem to be close to y^{SP} in the simulations.

The different trends in Figure 4.5 shows that the PI controller can cope with smaller stiction bands, but the larger the stiction band, the more excessive the valve travel becomes. Therefore the error between y and y^{SP} becomes larger, and y moves closer to alarm and trip limits while increased movement of u may exacerbate existing valve damage.

The technique of using faster or slower PI tuning in order to keep the average of y close to y^{SP} was simulated. The results are compared with moderate tuning as shown in Figure 4.6. The PI tuning was changed from fast tuning, $K_c = 3.0$, $\tau_i = 2.5$ min, to moderate tuning, $K_c = 1.5$, $\tau_i = 5$ min, to slow tuning, $K_c = 0.75$, $\tau_i = 15$ min. A disturbance signal of 10 m^3/h is introduced at t = 1 min and stiction of 10 % is simulated for all instances.

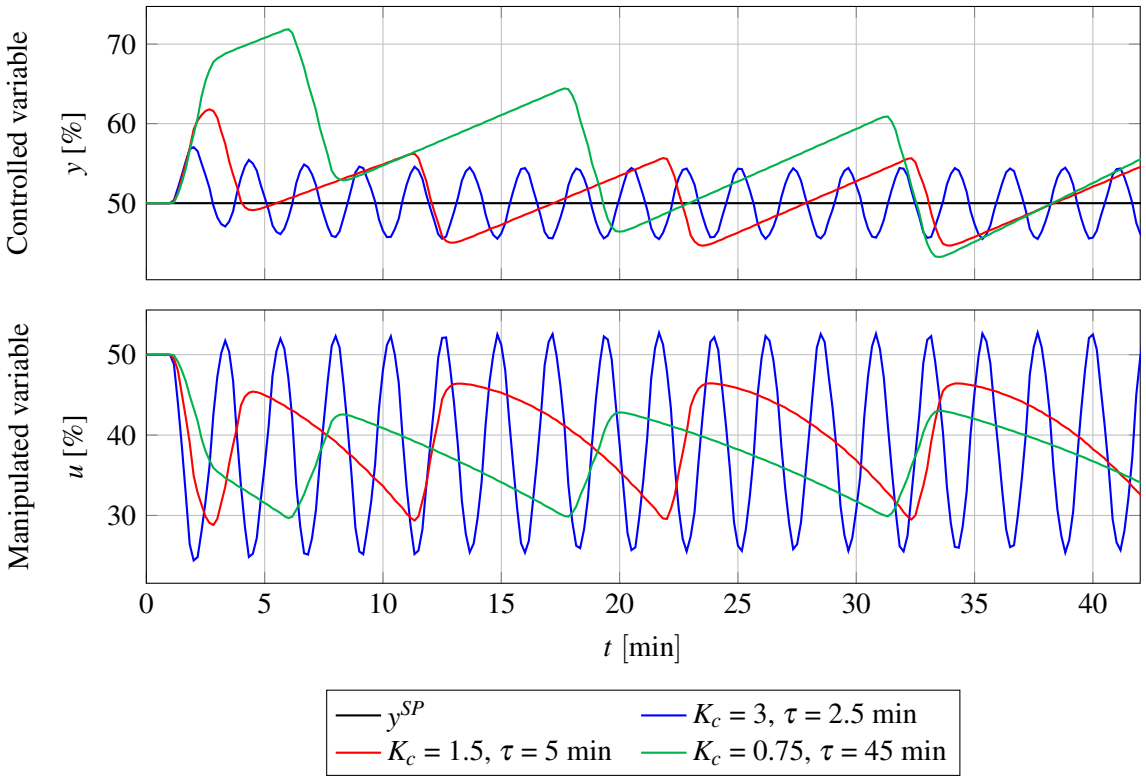

Figure 4.6. Impact of PI controller tuning on stiction.

Using more aggressive tuning causes the controller to adjust u by larger amounts per execution cycle, which causes u to move past the stiction band sooner, causing a faster cycle in both y and u. Using fast tuning enables the PI controller to keep y closer to y^{SP}, but movement in u is excessive. Using slower PID tuning also does not provide better results, as the controller moves u by smaller amounts, but the cycle caused by the stiction remains and y moves further away from y^{SP}. In all cases a cycle remains as there were no additional disturbances. Comparing the PI controller's performance using different tuning values agrees with Silva and Garcia (2014), that faster PI tuning is not a good stiction compensation technique. The PI controller using slower tuning shows the risk of breaching alarm or trip limits increases as tuning is made less aggressive.

Figure 4.7 shows how the time-based stiction compensation with a cycle time of 15 s is able to closely approximate the original control response when no stiction is present. Though y and u both cycle, the step disturbance is rejected in a way that is quite close to the system without stiction and y is brought back to y^{SP} in the same amount of time as the original system.

Figure 4.8 shows that when a cycle time of 60 s is chosen, the control response shows marked

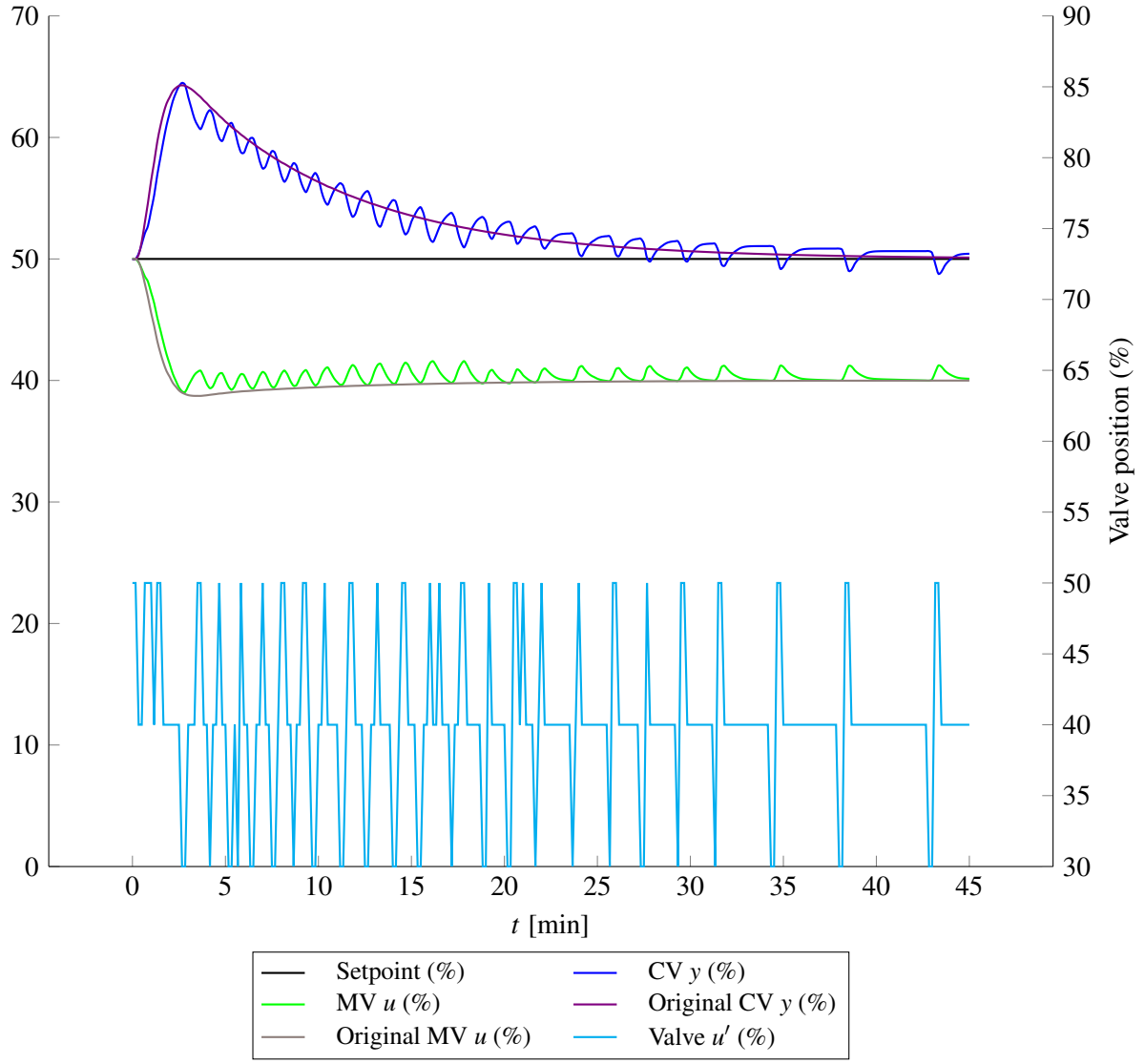

Figure 4.7. Impact of time-based stiction compensation.

deterioration. Again, the same disturbance of 10 m^3/h and a stiction band of 10% was used as in all simulations.

4.3.2 Simulation results of time-based stiction compensation with move suppression

Time-based stiction with move suppression, as discussed in Section 3.3, can minimise valve travel at the expense of increased error. As shown in Figure 4.9, simulating time-based compensation with move suppression, with an *SB* of 10% and a cycle time of 15 s, will bring y back to y^{SP} after a single disturbance of 10%, but not as well as when using only time base stiction compensation. The advantage is that less control moves are required to control the process. Therefore this method was implemented on a process plant, the results can be seen in Section 5.3.

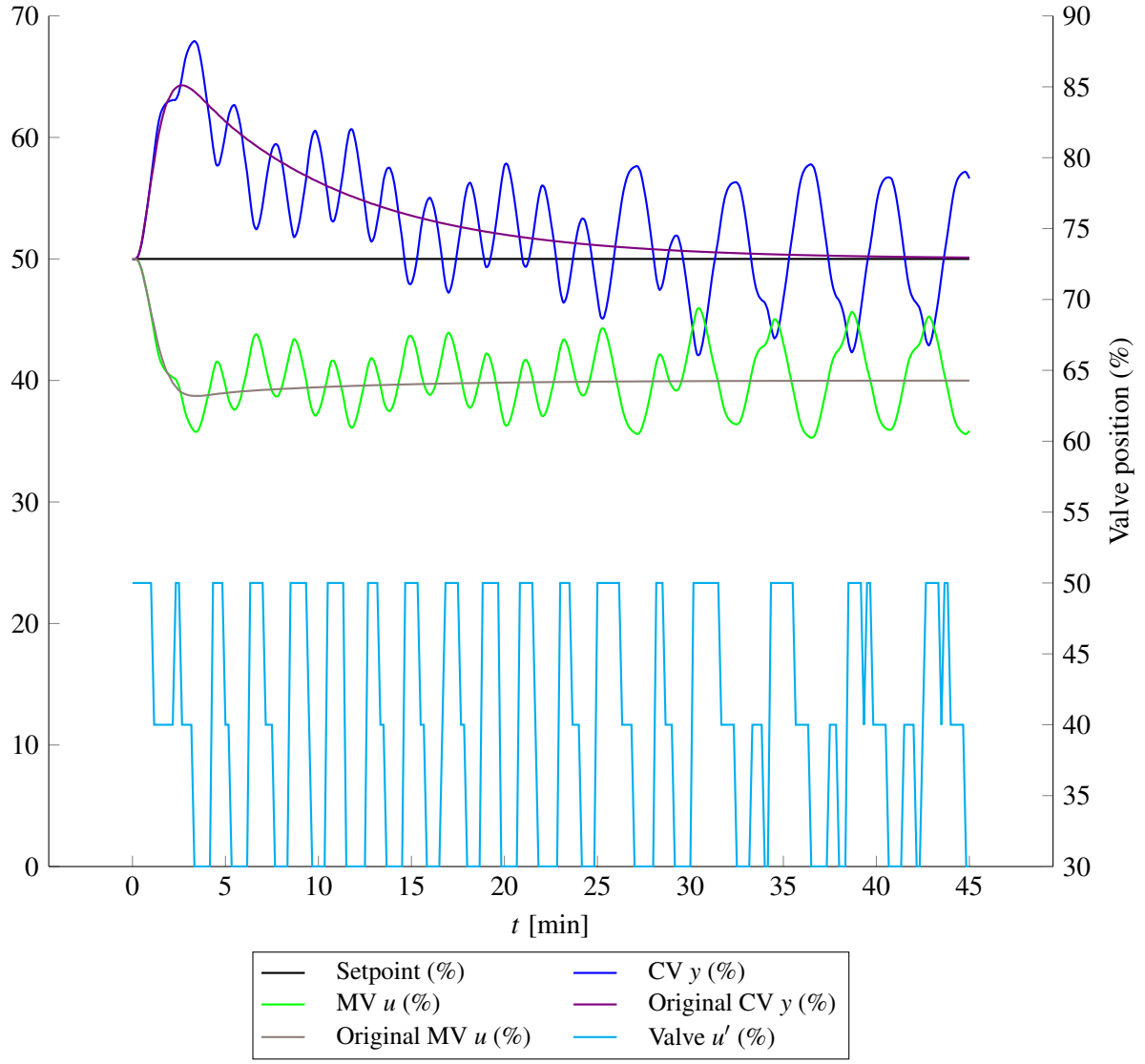

Figure 4.8. Impact of a slower cycle time on stiction compensation.

4.4 CHAPTER SUMMARY

In this chapter simulations using the IGC, RHC, SOALC and RHIGC to control a typical feed drum level while minimising movement of u are shown and discussed. The purpose of the simulations is not to compare the performance of the different controllers, but to show how their responses differ in how well they are able to keep y between desired limits while minimising δu.

The performance of RHIGC is compared to MPC in a 24-hour real-time simulation using typical plant data as a disturbance signal. The metrics show that performance was similar but the difference of the cost of deployment of the controllers, as well as the need for additional hardware, software and

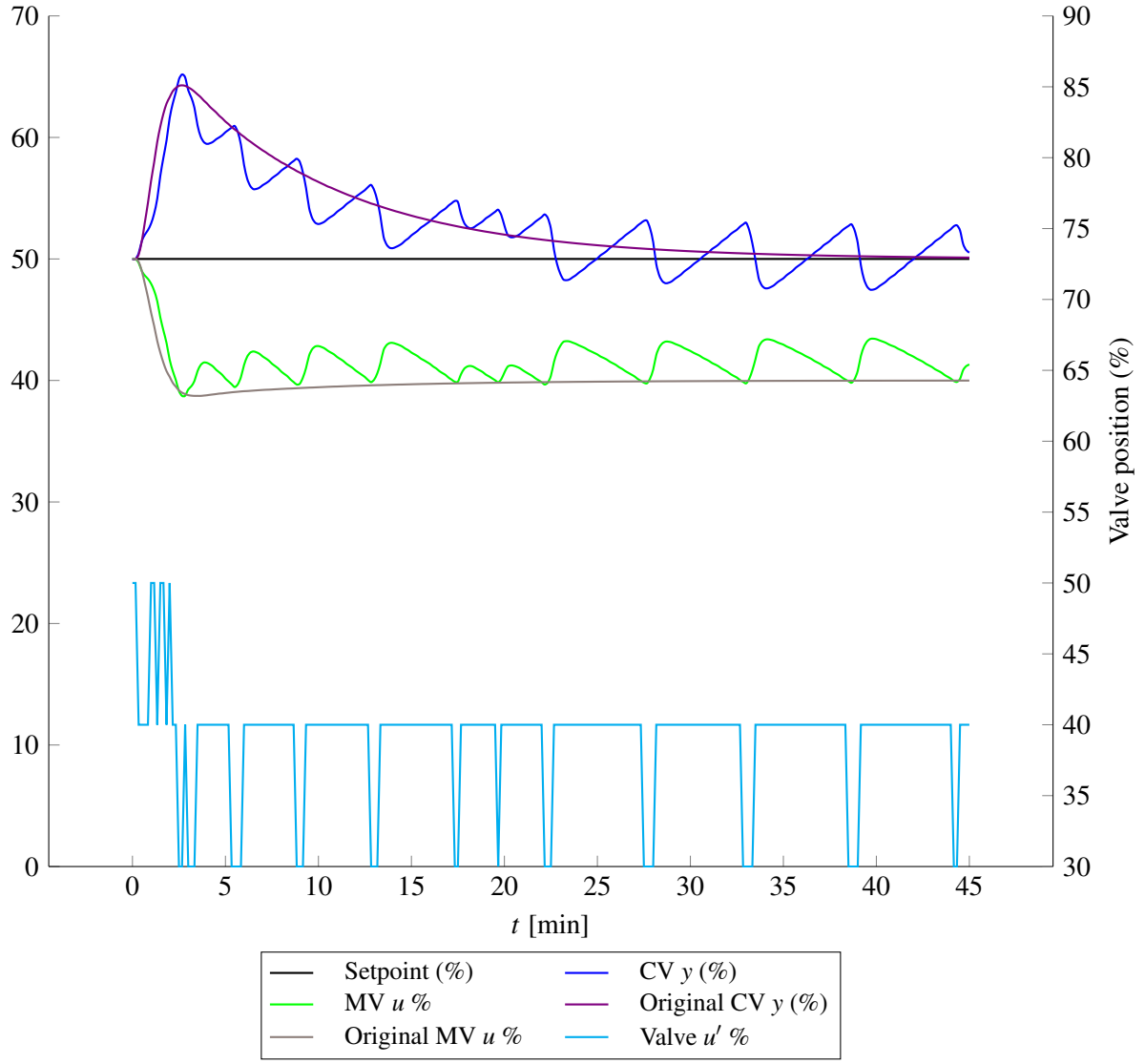

Figure 4.9. Impact of move suppression.

communication protocols for MPC also needs to be considered.

The two stiction compensation methods are simulated and the results compared to the original system with and without stiction, as well as an attempt to overcome stiction by using more aggressive tuning. The simulations show how time-based stiction compensation can bring the controller response back to where it resembles the original system without stiction, while time-based stiction compensation with move suppression is able to keep y within a reasonable distance from y^{SP} while minimising movement of the valve. Here it is also clear that the ability of the stiction compensation algorithms to minimise control error and/or minimise valve movement is worth the added complexity.

CHAPTER 5 INDUSTRIAL IMPLEMENTATION

5.1 CHAPTER OVERVIEW

This chapter discusses the results of industrial implementation of two averaging level control strategies described in Chapter 3, specifically RHC and RHIGC. The implementation was done on a typical industrial feed drum and the results compared with that of an existing PI gap original PI gap controller. The results of implementing time-based stiction compensation with move suppression on a level controller on a typical industrial tank are compared to the original controller as well as a controller that was tuned much faster in an attempt to compensate for the valve stiction.

5.2 AVERAGING LEVEL CONTROL

If averaging level control is implemented on a plant where multiple disturbances that move the level in the same direction can occur, RHIGC will bring the level back to y^{SP} after a disturbance. This will ensure that the control system is in a better position to reject consecutive disturbances. This ability was demonstrated on a typical process plant with a significant improvement in performance as shown in Figure 5.1 and Table 5.2.

RHIGC was implemented on a standard DCS on a real process plant in the petrochemical industry where a varying feedrate into a feed drum causes the level to vary continuously, similar to the process flow diagram shown in Figure 4.1. If the low level is violated, the reactor feed pumps will cavitate and if the high level is violated, a high-level protection controller will send reactor feed material to another process where the feed would be converted to a lower value product.

There was an existing PI gap controller implemented on the process. RHC was implemented on the process to improve the control by replacing the existing PI gap controller. The control was subsequently

Table 5.1. Tuning parameters for plant test.

Controller	Parameter	Value	Units
PI gap	K_c	0.95	$\%^{-1}$
	τ_i	180	min
	K_r	0.625	-
RHC	K_G	20	$\%^{-1}$
	T_{RH}	100	min
RHIGC	K_c	0.6	$\%^{-1}$
	τ_i	180	min
	$\tau_{i(gap)}$	247	min
	K_G	20	$\%^{-1}$
	T_{RH}	100	min

further improved by replacing RHC with RHIGC. The PI gap controller, RHC and RHIGC were all executing at t_s = 1 s, and their tuning parameters are given in Table 5.1.

The RHC part of RHIGC was tuned to keep the level between 30% and 80%. The IGC part of RHIGC was tuned by inspection of plant data to identify a typical and maximum feed disturbance magnitude. Averaging level tuning rules proposed by King (2016) were used to calculate K_c and τ_i for the IGC part of RHIGC. The RHC part of the controller was given a horizon of 100 minutes based on trial and error.

Figure 5.1 shows the level and outlet flowrate set by u, each for a different 24-hour timespan of typical operation, using the original PI gap controller, RHC and RHIGC respectively. As can be seen, all controllers are well tuned and reasonably successful in keeping the level between the range of 30% to 80%, while making minimum control moves.

The comparison of the controllers according to the performance metrics standard deviation (2.10) and TV/N (2.11) are shown in Table 5.2, along with the highest value (y_{high}) and the lowest value (y_{low}) of y during the time each controller was used. The PI gap controller was tuned well as it kept the level between limits most of the time. It did let the level go outside the high limit of 80%. RHC was able to successfully minimize u movement, but left y close to limits, increasing the risk that subsequent

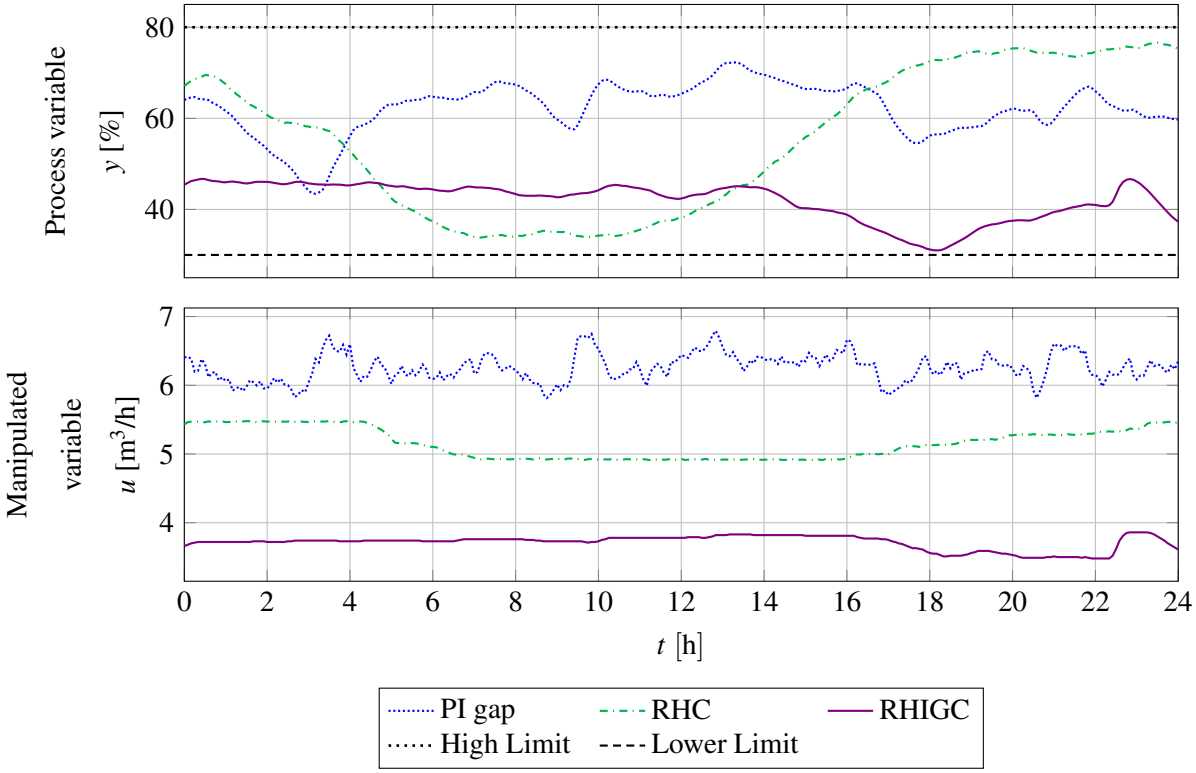

Figure 5.1. Behaviour of the PI gap, RHC and RHIGC controllers on the plant.

Table 5.2. Comparison of RHC, RHIGC and PI gap control.

	y_{high}	y_{low}	σ_u	TV/N
PI gap controller	84.7	36.7	0.33	0.15
RHC	77.6	30.6	0.057	0.043
RHIGC	78.2	29.4	0.052	0.013

disturbances may push the level over the limits. Both σ_u and TV/N show that both RHC and RHIGC moved u much less than the PI gap controller, with associated process improvements downstream, due to increased stability.

5.3 STICTION COMPENSATION

The method of time-based stiction compensation with move suppression was implemented on a tank in industry that had stiction in excess of 7% in a make-up water valve. The result can be compared to the initial tuning as shown in Figure 5.3, as well as an attempt to use faster tuning to compensate for the existing stiction as shown in Figure 5.2.The initial tuning constants of the PI controller was $K_c = 0.75$ and $\tau = 30$ min. To test the impact of faster tuning variables, the value of τ was decreased to 10 min,

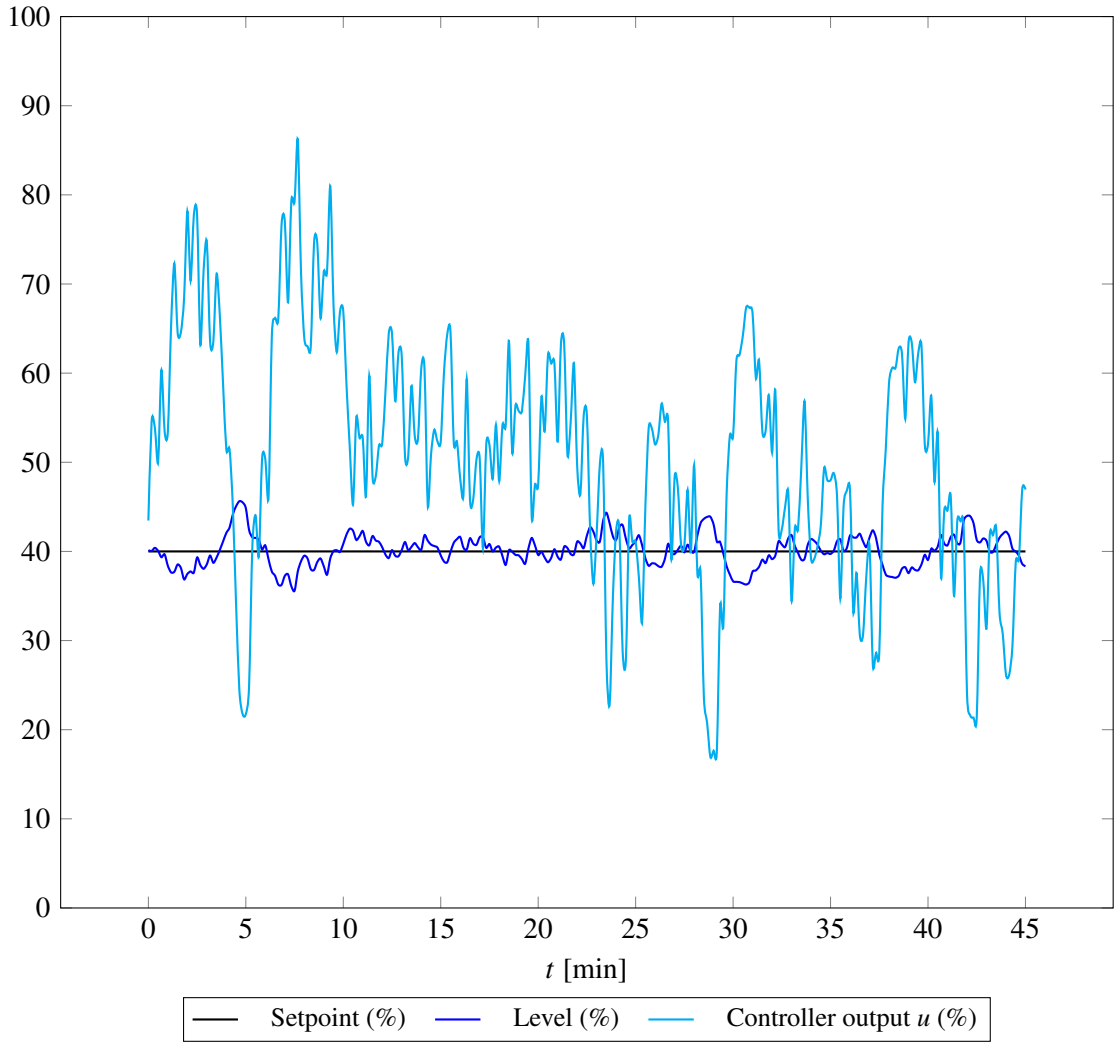

Figure 5.2. Plant results using fast tuning.

while K_c was kept at 0.75. To test time-based stiction compensation with move suppression, τ was changed back to a value of 30 min.

Figure 5.4 shows the results of time-based stiction compensation with move suppression. A gap of 4% on both sides of the y^{SP} of 35% was used with a maximum rate of change of 4% per minute and a minimum rate of change of 0.5% per minute. Unfortunately, the data historian on the plant stored the output of the compensation algorithm and not the controller output, which is an issue when comparing the result with the previous graphs. It also brings the comparison of the standard deviation in Table 5.3 into question, as the variability of u is compared to the variability in u'. It is also clear that the average value of u' in Figure 5.4 is lower than in Figures 5.2 and 5.3. This indicated that the make-up water flow was lower during the test of the time-based stiction compensation. Unfortunately

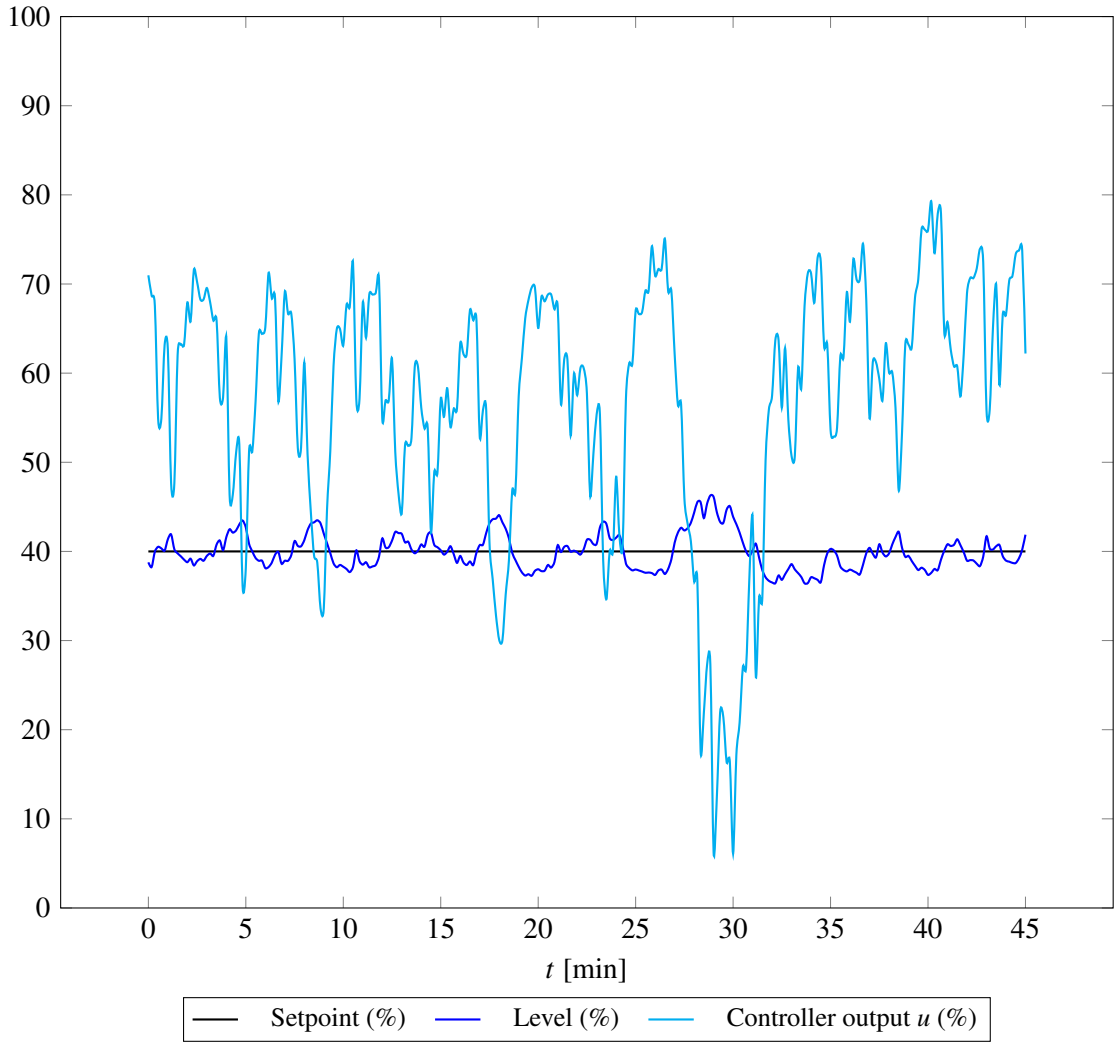

Figure 5.3. Plant results using initial tuning.

the valve was replaced and the test could not be repeated. However, the decrease in valve movement was substantial.

A comparison of the level with original tuning, fast tuning and time-based stiction compensation with move suppression is shown in Table 5.3. This shows that the time-based compensation with move suppression kept y close to y^{SP} while decreasing valve position variability.It is clear that time-based compensation with move suppression outperforms both original and faster tuned PI controllers in terms of decreasing movement of u and decreasing the range of y. To compare the reduction in valve travel the standard deviation of u (2.10) and the total variance of u TV/N (2.11) is used.

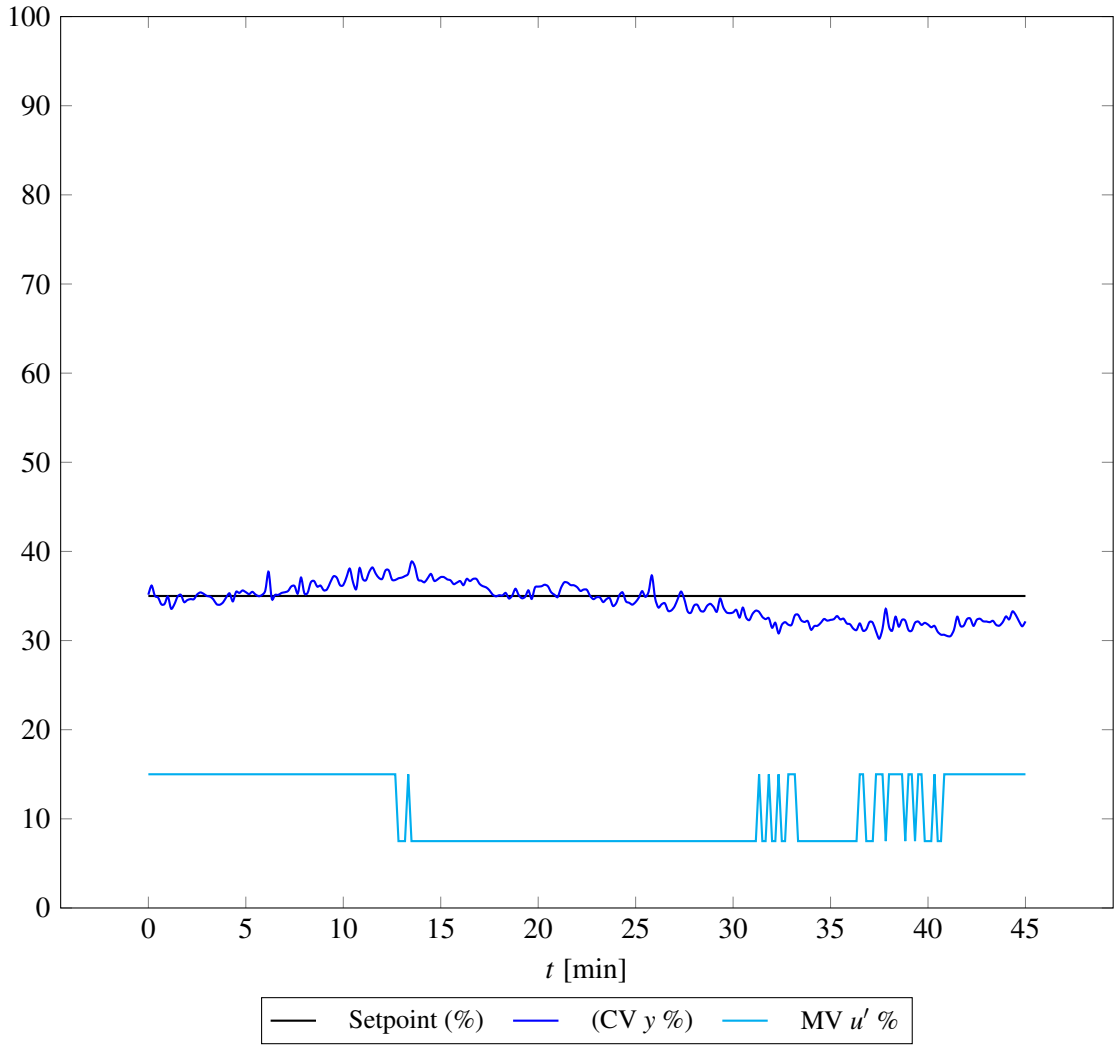

Figure 5.4. Results of time-based stiction compensation with move suppression on an industrial plant.

Table 5.3. Comparison of plant results.

	y_{high}	y_{low}	σ_u	TV/N
Initial tuning	46.3	34.8	13.7	4.2
Fast tuning	45.6	35.6	10.0	111
Time-based stiction compensation	42.2	26.2	4.4	0.65

5.4 CHAPTER SUMMARY

This chapter discusses the results of implementing RHC and RHIGC on a typical industrial feed drum and compares the results with the original PI gap controller. The comparison of the controllers according to the performance metrics show that both RHC and RHIGC were able to move u much less

than the original PI gap controller. These results show that the increase in performance is worth the increased complexity.

The results of implementing time-based stiction compensation with move suppression on a level controller on a typical industrial tank were compared to the original PI controller as well as a PI controller that was tuned much faster. The decrease in valve movement was substantial even though there were issues with comparing the different trials.

CHAPTER 6 DISCUSSION AND CONCLUSION

6.1 CHAPTER OVERVIEW

This chapter summarises the simulations and plant implementations of more complex ARC that lead to improved control performance, showing that modern DCS offer much more functionality than what most control engineers use.

6.2 CONCLUSION

6.2.1 Averaging level control

This thesis described four ARC techniques for averaging level control namely integral gap control (IGC), ramp horizon control (RHC), simplified optimal averaging level control (SOALC), and a combination of RHC and IGC called ramp horizon integral gap control (RHIGC).

IGC decreases movement in u when y is in the defined gap, therefore reducing variability in u when there is little risk of y breaching a limit. It is easy to implement and tune and has the added advantage that it provides an easy solution to cycling level controllers as found on many industrial plants.

SOALC was not implemented and tested on a plant because it will keep manipulating u when there is no imminent risk of y going over a limit. SOALC will also not move u to return y to y^{SP} once y has reached but not exceeded a limit. SOALC will make the closed-loop unstable once a limit has been breached. This can be countered by changing the sign of Δu. Even if this is done, SOALC will then only balance the level outside of the breached limit, and will not move it back to between limits. A better solution is to switch to PI control while the level is in violation of a limit. Upon consideration of these issues, it was decided not to test SOALC on a plant.

RHC will decrease variability in u and will employ a "wait and see" approach by making no control

moves unless there is reasonable certainty that a limit will be breached. A shortcoming of RHC and SOALC is that if the level is already at or close to a limit because of an earlier disturbance, another disturbance in the same direction may cause a violation of limits. This is independent of the size of the initial disturbance. This may be overcome by setting the limits conservatively and thereby sacrificing potential improved stability in u.

RHIGC combines the strengths of RHC and IGC to improve the averaging level control in a process plant. RHIGC is able to keep y within limits in the presence of large disturbances due to the RHC component. RHIGC is able to return y to y^{SP} in a manageable time frame after frequent disturbances due to the IGC component. Additionally, RHIGC aims to move u as little as possible in light of frequent and large disturbances.

RHIGC is able to achieve similar results when compared to MPC in a simulation environment and has the advantage of being implementable using standard DCS blocks. RHIGC have advantages over MPC in that it can be executed at high sampling frequencies and does not require specialised software and hardware.

RHIGC was implemented on a real process plant and showed good improvements when compared with an existing PI gap controller and RHC. The control techniques were compared using the standard deviation and total variance as performance metrics.

RHIGC showed good results in preventing the level from going outside of set high and low limits while simultaneously decreasing variability of the manipulated variable. In process plants, keeping the level between set limits may avoid alarm or trip limits being exceeded, while the decreased variability of the manipulated variable leads to increased stability of the downstream process equipment.

All four averaging level control techniques were implemented on a standard Honeywell DCS, with no additional hardware, software or communications protocols required, showing that modern DCS are able to provide much more than the typical control systems found in industry. Implementations of the controllers on simulators and industrial processes showed how control performance can be enhanced by making use of more of the capabilities of modern DCS. The additional complexity in the controllers is substantial, but the improvement in control performance and lower cost of implementation makes it worthwhile.

6.2.2 Stiction compensation

It was shown in simulation that time-based stiction compensation can keep the average valve position during a cycle at the output of the controller, which leads to improved controller performance as indicated by y staying closer to y^{SP}. While the valve will only implement moves larger than the stiction band, keeping the average value of the valve position at the controller output ensures that a result that is very close to the original system before stiction occurred.

Because valve stiction is indicative of control hardware that is under duress, it is desirable to move the valve as little as possible. When adding the ability of the algorithm to suppress making changes in u when y is not at risk of going too far from y^{SP} in a set period, the method of time-based stiction compensation with move suppression will let y move over a larger range, but it is still able to control y successfully by keeping it in a reasonable range from y^{SP}. This can be seen as a "close enough" approach, where valve position moves are withheld if there is no imminent danger of triggering alarms or trip limits. Time-based stiction compensation with move suppression was demonstrated to be successful in minimising variance in u' while still keeping y close enough to y^{SP}, in both simulation and an industrial implementation on a drum level.

Both stiction compensation techniques were implemented on a standard Honeywell DCS, with no additional hardware, software or communications protocols required. Implementing the stiction compensation on simulated and industrial processes showed that the impact of stiction can be decreased, showing that modern DCS are capable of improving control performance by making use of new functionality.

REFERENCES

Bennett, S. (1996). A brief history of automatic control, *IEEE Control Systems Magazine* **16**(3): 17–25.

Brooks, K., Burchell, J. and Pieterse, F. (2017). Model predictive control of an exothermic pressure leach circuit, *IFAC-PapersOnLine* **50**(1): 10232–10237.

Campo, P. J. and Morari, M. (1989). Model predictive optimal averaging level control, *AIChE Journal* **35**(4): 579–591.

Chen, D. and Seborg, D. E. (2002). Relative gain array analysis for uncertain process models, *AIChE Journal* **48**(2): 302–310.

Cheung, T.-F. (1979). Liquid-level control in single tanks and cascades of tanks with proportional-only and proportional-integral feedback controllers, *Industrial & Engineering Chemistry Fundamentals* **18**(1): 15–21.

Cheung, T.-F. and Luyben, W. L. (1980). Nonlinear and nonconventional liquid level controllers, *Industrial & Engineering Chemistry Fundamentals* **19**(1): 93–98.

Corripio, A. B. (1982). Digital control techniques, *AIChE, Washington, Series A* **3**: 12.

Cutler, C. R. and Ramaker, B. L. (1980). Dynamic matrix control - a computer control algorithm, *Proceedings of the 1980 Joint Automatic Control Conference, 13-15 August 1980, San Francisco, USA*, Vol. 4, pp. 3020–3025.

Daiguji, M. and Yamashita, Y. (2022). An approach for stiction compensation in industrial process control valves, *Computers & Chemical Engineering* **158**: 107641.

De Souza L. Cuadros, M. A., Munaro, C. J. and Munareto, S. (2012). Novel model-free approach for stiction compensation in control valves, *Industrial & Engineering Chemistry Research* **51**(25): 8465–8476.

Desoer, C. A. and Shahruz, S. M. (1986). Stability of dithered non-linear systems with backlash or hysteresis, *International Journal of Control* **43**(4): 1045–1060.

Di Capaci, R. B., Scali, C. and Huang, B. (2016). A revised technique of stiction compensation for control valves, *IFAC-PapersOnLine* **49**(7): 1038–1043.

Djavdan, P. (1993). Design of an on-line scheduling strategy for a combined batch/continuous plant using simulation, *Computers & Chemical Engineering* **17**(5-6): 561–567.

Downs, J. J. and Vogel, E. F. (1993). A plant-wide industrial process control problem, *Computers & Chemical Engineering* **17**(3): 245–255.

Dzedzemane, R., le Roux, J., Muller, C. and Craig, I. (2018). Steam header state-space model development and validation, *IFAC-PapersOnLine* **51**(21): 207–212.

Faanes, A. and Skogestad, S. (2004). Feedforward control under the presence of uncertainty, *European Journal of Control* **10**(1): 30–46.

Friedman, Y. Z. (1994). Tuning averaging level controllers, *Hydrocarbon Processing (International ed.)* **73**(12): 101–104.

Gous, G. (2000). Control level under fouling conditions, *Hydrocarbon Processing* **79**(11): 71–71.

Gous, G. and de Vaal, P. (2021). Integral vs. proportional gap for averaging level control. *Hydrocarbon Processing*.

https://www.hydrocarbonprocessing.com/magazine/2021/july-2021/bonus-report-the-digital-plan/integral-vs-proportional-gap-for-averaging-level-control Last accessed on 24 July 2023.

Gous, G., de Vaal, P. and van Coller, G. (2021). Implementation of averaging level control using native DCS functions. *Hydrocarbon Processing*. https://www.hydrocarbonprocessing.com/magazine/2021/may-2021/process-controls-instrumentation-and-automation/implementation-of-averaging-level-control-using-native-dcs-functions Last accessed on 24 July 2023.

Gous, G. Z. (1993). Personal communication, *National Petroleum Refiners of SA (Pty) Ltd (NATREF)* .

Gous, G. Z. (2021). Personal communication, *Analyte and Anglo American* .

Gous, G. Z., le Roux, J. D. and Craig, I. K. (In Press). Time based stiction compensation compensation, *IFACPapersOnline* .

Gous, G. Z., Wiid, A. J., le Roux, J. D. and Craig, I. K. (2023). Advanced regulatory control techniques for improved averaging level control performance, *Industrial & Engineering Chemistry Research* **62**(38): 15578–15587.

Grimholt, C. and Skogestad, S. (2018). Should we forget the Smith Predictor?, *IFAC-PapersOnLine* **51**(4): 769–774.

Gupta, Y. (1998). Control of integrating processes using dynamic matrix control, *Chemical Engineering Research and Design* **76**(4): 465–470.

Gustafsson, T. K. and Waller, K. V. (1992). Nonlinear and adaptive control of pH, *Industrial & Engineering Chemistry Research* **31**(12): 2681–2693.

Hägglund, T. (2002). A friction compensator for pneumatic control valves, *Journal of Process Control* **12**(8): 897–904.

Henson, M. A. and Seborg, D. E. (1994). Adaptive nonlinear control of a pH neutralization process, *IEEE Transactions on Control Systems Technology* **2**(3): 169–182.

Horton, E., Foley, M. and Kwok, K. (2003). Performance assessment of level controllers, *International Journal of Adaptive Control and Signal Processing* **17**(7-9): 663–684.

Huba, M., Skogestad, S., Fikar, M., Hovd, M., Johansen, T. and Rohal'-Ilkiv, B. (2011). *Selected Topics on Constrained and Nonlinear Control*, STU Bratislava – NTNU Trondheim.

Kelly, J. D. (1998). Tuning digital PI controllers for minimal variance in manipulated input moves applied to imbalanced systems with delay, *The Canadian Journal of Chemical Engineering* **76**(5): 967–974.

King, M. (2016). *Process Control: A Practical Approach*, John Wiley & Sons, United Kingdom.

Lakerveld, R., Benyahia, B., Heider, P. L., Zhang, H., Braatz, R. D. and Barton, P. I. (2013). Averaging level control to reduce off-spec material in a continuous pharmaceutical pilot plant, *Processes* **1**(3): 330–348.

Lee, M. and Shin, J. (2009). Constrained optimal control of liquid level loop using a conventional proportional-integral controller, *Chemical Engineering Communications* **196**(6): 729–745.

Lindholm, A. (2009). *Buffer management strategies for improving plant availability*, Master's thesis, Lund University.

Liu, L., Tian, S., Xue, D., Zhang, T. and Chen, Y. (2019). Industrial feedforward control technology: a review, *Journal of Intelligent Manufacturing* **30**: 2819–2833.

Luyben, W. L. (2022). Control of distillation columns with dual steam and hot-oil reboilers, *Chemical Engineering and Processing-Process Intensification* **179**: 109068.

Majanne, Y. (2005). Model predictive pressure control of steam networks, *Control Engineering Practice* **13**(12): 1499–1505.

McDonald, K. A. and McAvoy, T. J. (1987). Application of dynamic matrix control to moderate-and high-purity distillation towers, *Industrial & Engineering Chemistry Research* **26**(5): 1011–1018.

McDonald, K. A., McAvoy, T. and Tits, A. (1986). Optimal averaging level control, *AIChE Journal* **32**(1): 75–86.

Miller, M. (2018). Thickener design, control and development, *ALTA, 18-26 May 2018, Perth, Australia*.

Mitra, S. (2015). Design considerations of hot oil system, *New Castle, England* .

Mohammad, M. A. and Huang, B. (2012). Compensation of control valve stiction through controller tuning, *Journal of Process Control* **22**(9): 1800–1819.

Morari, M. (1987). Robust process control, *Chemical Engineering Research & Design* **65**(6): 462–479.

Ogawa, S., Allison, B., Dumont, G. and Davies, M. (2002). A new approach to optimal averaging level control with state constraints, *Proceedings of the 41st IEEE Conference on Decision and Control, 10-13 December 2002, Las Vegas, NV, USA.*, Vol. 2, pp. 1952–1957.

Qin, S. J. and Badgwell, T. A. (2003). A survey of industrial model predictive control technology, *Control Engineering Practice* **11**(7): 733–764.

Rambalee, P., Gous, G., De Villiers, P., McCulloch, N. and Humphries, G. (2010). Control and stabilization of a multiple boiler plant: An APC approach, *IFAC Proceedings Volumes* **43**(9): 109–114.

Reyes-Lúa, A., Backi, C. J. and Skogestad, S. (2018). Improved PI control for a surge tank satisfying level constraints, *IFAC-PapersOnLine* **51**(4): 835–840.

Reyes-Lúa, A. and Skogestad, S. (2019). Systematic design of active constraint switching using classical advanced control structures, *Industrial & Engineering Chemistry Research* **59**(6): 2229–2241.

Reyes-Lúa, A. and Skogestad, S. (2020). Multi-input single-output control for extending the operating range: Generalized split range control using the baton strategy, *Journal of Process Control* **91**: 1–11.

Rosander, P., Isaksson, A. J., Löfberg, J. and Forsman, K. (2012). Practical control of surge tanks suffering from frequent inlet flow upsets, *IFAC Proceedings Volumes* **45**(3): 258–263.

Rosander, P., Isaksson, A., Löfberg, J. and Forsman, K. (2011). *Performance Analysis of Robust Averaging Level Control*, Linköping University Electronic Press.

Sanchis, R., Romero, J. and Martín, J. (2011). A new approach to averaging level control, *Control Engineering Practice* **19**(9): 1037–1043.

Sausen, A., Sausen, P. S. and de Campos, M. (2014). The error-squared controller: a proposed for computation of nonlinear gain through Lyapunov stability analysis, *Acta Scientiarum. Technology* **36**(3): 497–504.

Sausen, A., Sausen, P. S., Reimbold, M. and de Campos, M. (2012). Application and comparison of level control strategies in the slug flow problem using a mathematical model of the process, *Acta Scientiarum. Technology* **34**(4): 441–449.

Sausen, A., Sausenand, P. S. and de Campos, M. (2014). The error-squared controller: a proposed for computation of nonlinear gain through lyapunov stability analysis, *Acta Scientiarum. Technology* **36**(3): 497–504.

Sbarbaro, D. and Ortega, R. (2005). Averaging level control of multiple tanks: A passivity based approach, *Proceedings of the 44th IEEE Conference on Decision and Control*, IEEE, pp. 7384–7389.

Seborg, D. E., Edgar, T. F., Mellichamp, D. A. and Doyle III, F. J. (2016). *Process Dynamics and Control*, John Wiley & Sons, United Kingdom.

Shinskey, F. G. (2010). Special rules for tuning level controllers (part-1). https://www.controlglobal.com/measure/level/article/11383379/

process-automation-special-rules-for-tuning-level-controllers-part-1-control-global. Last accessed on 24 July 2023.

Sidhu, M. (2003). *Multivariable averaging level control*, Master's thesis, University of British Columbia.

Silva, B. C. and Garcia, C. (2014). Comparison of stiction compensation methods applied to control valves, *Industrial & Engineering Chemistry Research* **53**(10): 3974–3984.

Singh, A., De Villiers, P., Rambalee, P., Gous, G., De Klerk, J. and Humphries, G. (2010). A holistic approach to the application of model predictive control to batch reactors, *IFAC Proceedings Volumes* **43**(9): 127–132.

Skogestad, S. (2003). Simple analytic rules for model reduction and PID controller tuning, *Journal of Process Control* **13**(4): 291–309.

Skogestad, S. (2023). Advanced control using decomposition and simple elements, *Annual Reviews in Control* **56**: 100903.

Skogestad, S. and Morari, M. (1987). Control configuration selection for distillation columns, *AIChE Journal* **33**(10): 1620–1635.

Smith, O. (1957). Closer control of loops with dead time, *Chemical Engineering Progress* **53**: 217–219.

Srinivasan, R. and Rengaswamy, R. (2005). Stiction compensation in process control loops: A framework for integrating stiction measure and compensation, *Industrial & Engineering Chemistry Research* **44**(24): 9164–9174.

Srinivasan, R. and Rengaswamy, R. (2008). Approaches for efficient stiction compensation in process control valves, *Computers & Chemical Engineering* **32**(1-2): 218–229.

Taylor, A. J. and La Grange, T. G. (2002). Optimize surge vessel control, *Hydrocarbon Processing* **81**(5): 49–52.

Van der Burg, M. and Djavdan, P. (1995a). Model predictive averaging level control using disturbance prediction, *IFAC Proceedings Volumes* **28**(12): 219–224.

van der Burg, M. and Djavdan, P. (1995b). Model predictive averaging level control using disturbance prediction, *IFAC Proceedings Volumes* **28**(12): 219–224.

Wade, H. L. (2005). Tuning level control loops, *Instrument Engineers' Handbook: Process Control and Optimization* **2**: 432–441.

Wang, S.-J. and Wong, D. S. (2006). Control of reactive distillation production of high-purity isopropanol, *Journal of Process Control* **16**(4): 385–394.

Williams, A. (2023). Fuels, furnaces, and fired equipment, *Industrial Energy Systems Handbook*, River Publishers, pp. 195–213.

Wolff, E. A. and Skogestad, S. (1996). Temperature cascade control of distillation columns, *Industrial & Engineering Chemistry Research* **35**(2): 475–484.

Wu, K.-L., Yu, C.-C. and Cheng, Y.-C. (2001). A two degree of freedom level control, *Journal of Process Control* **11**(3): 311–319.

Ye, N., Mcavoy, T. J., Kosanovich, K. A. and Piovoso, M. J. (1995). Optimal averaging level control for the Tennessee Eastman problem, *The Canadian Journal of Chemical Engineering* **73**(2): 234–240.

Zotică, C., Forsman, K. and Skogestad, S. (2022). Bidirectional inventory control with optimal use of intermediate storage, *Computers & Chemical Engineering* **159**: 107677.