NTNU | Department of Chemical Engineering

TKP4580 - Specialization Project

Decentralized control structures for balancing supply and demand in gas networks

Author: Erik André Klepp Vik

Trondheim, December 18, 2020

Supervisor: Sigurd Skogestad, IKP

Co-supervisor: Cristina Zotica, IKP

Abstract

One of the main objectives within process control is optimal and stable process plant operation. This project aimed to ensure stable operation in a typical low pressure steam network by optimizing the supervisory control layer using decentralized control structures. The system model represented the relatively uncharted area of balancing supply and demand in the network, where control implementation has previously shown large economic potential for these type of systems.

For the project model, three consumers sharing one common supplier were used. The problem is recognized when a critical disturbance causes the consumers to require more input than the producer can supply. In this case, the control structure should prioritize the most economical consumer thus ensuring stable and predictable plant operation. A modified mass balance in terms of pressure was used as a basis for the implementation of different decentralized control structures. The pressure balance was implemented in MATLAB and Simulink, where the control structures were implemented on top of the model for the system pressure balance.

First, a default control structure were implemented and assessed for a critical supplier disturbance as a basis to other control structures. Then, parallel control using equal setpoints were implemented and analysed in the same way, followed by parallel control using different setpoints. In the end, droop control were also implemented and tested.

Other controller structures including split range control and valve position control were found as good, feasible options compared with simple or no control structures. However, because of their complex implementation compared to other solutions, they were discarded.

In the end it was found that controlling both the consumers and the supplier using droop control would be a simple and effective implementation to ensure supply to the prioritized consumer. Droop control also showed more advantageous results for setpoint control and stability than other solutions. To ensure the validity of the results, further research is recommended within centralized control, for example by comparison to a model predictive controller.

Table of Contents

List of Figures

List of Tables

Abbreviations

The list presented in this section describes all abbreviations and acronyms that could be used within the body of this specialization project report:

AC Alternating current

AE Algebraic equation

CV Controlled variable

DAE Differiental algebraic equations

DCS Decentralized control structure

DOF Degrees of freedom

DV Disturbance variable

HP High pressure

LP Low pressure

MC Holdup (mass) controller

MV Manipulated variable

NN Neural network

ODE Ordinary differential equations

PC Pressure controller

PID Proportional-integrating-Derivative

SIMC Skogestad internal model control

VPC Valve position control

Chapter 1

Introduction

Oxford English Dictionary refers to optimization as the action of making the best or most effective use of a situation or resource.[3] One optimization issue with large saving and carbon-reducing potential in the industry is balancing supply and demand in steam generation and distribution networks. Steam generation and distribution is usually a large part of a plant operating costs, and play a large part in the chemical industry. In 2012, the cellulose company Nipo-Brasiliera S.A optimized their steam supply and demand using advanced control structures and reduced the plant excess steam use by 90 %.[1] This reduction only shows the great potential in systematic implementation of advanced control structures.

Advanced control structures include multivariable and decentralized control. It is often a question what to use and when, as there exists advantages and disadvantages to both. Therefore, this project will systematically approach and explore the possibility of using decentralized control structures for the common but little studied supply and demand case in gas networks. The project will then utilize the results from the decentralized control structure analysis and compare the results to multivariable control in the continued work of a master thesis.

The basis for the analysis will assume that there is one low pressure steam supplier and three low pressure consumers. An optimization issue occurs when the consumers demand more input than the supplier can provide. The problem is mentioned in gas networks, electrical grids and possibly many other applications.[8] Thus, this project aim is two-fold: First, the optimal operation for a system with one supplier and three consumers needs to be defined through the use of a dynamic model. Then, the consumers need to be prioritized when the supply is not large enough. For the second part, we want to find the simplest way to implement optimal operation for critical disturbances using decentralized control structures on the dynamic system defined in part one.

Achieving the project goals requires identification of the relevant manipulated variables (MVs), controlled variables (CVs), main disturbances (DVs) and constraints. Then, a mathematical model representing the dynamic behaviour of the system should be modeled. After this, controller implementation based on analysis of the most promising decentralized control structures can be done. The most promising controller structures should then be compared in performance, constraint handling and disturbance rejection to achieve optimal operation.

Chapter 2

Theory

2.1 Process Control

Chemical processes often require precise control of measured process data, such as for example flow rates, pressures or valves, in order to keep the system at steady state conditions. Process control is defined as the methods applied to control such processes in a process plant. There are three main goals of process control, namely: Setpoint tracking, disturbance rejection and ensuring safe operation. [5]

Setpoint tracking ensures that the process follows a given trajectory for a given process. This is usually the optimal operating point given process constraints and limitations. Following the optimal operating point will yield optimal operation which again maximize process efficiency. [5]

Disturbance rejection involves that compensating for disturbances affecting process output. This is important to achieve optimal setpoint tracking, as well as consistency, predictability and process safety. Disturbance rejection also includes reduction of process variability. [5]

Ensuring safety is a result of setpoint tracking and disturbance rejection. Process equipment may have physical limitations such as a maximum temperature or pressure. If these limitations or constraints are violated, it may pose a risk to the surrounding environment. Thus, good setpoint tracking and disturbance rejection will help avoid creating dangerous situations, ensuring safety. [5]

The control of a chemical plant is usually divided into a hierarchy based like the on seen in Figure 2.1. For direct process control, which this project embrace, one usually consider the three lower control layers. That is, the control layer, which is used to control the plant layer, and the plant itself, including the physical equipment in the plant. The plant layer is not actually in charge of controlling anything itself, as by definition it is controlled by the control layer. It is the physical process units used for system control and consists merely of the system manipulated variables (MVs), also referred to as the system inputs u. In this project, the possible MVs will all be valve actuators. However, MVs are not limited to being valve actuators. What defines the MVs are that they can be manipulated in order to affect the process, and they can be controlled by the control layer. By this definition, power inputs, pump speed and many other process variables could be used as MVs.

The control layer changes the system MVs in order to keep the systems controlled variables (CVs) or outputs y at given setpoints. That is, the control layer is unable to change the system CVs itself,

instead it uses the system inputs for CV setpoint tracking. Usually the control layer is divided into a supervisory and regulatory control layer. The main regulatory control layer action is stabilizing the process, thus keeping operation at steady state conditions.

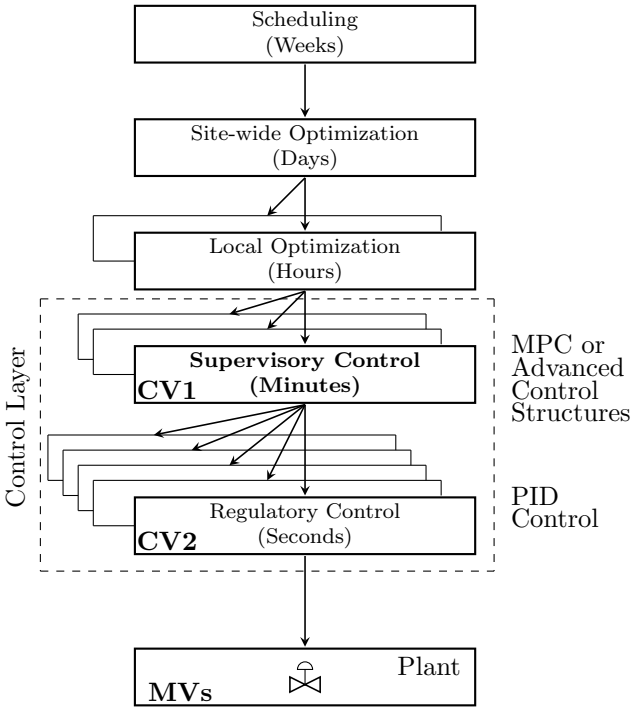

Figure 2.1: Typical control hierarchy in a process plant. [9]

The regulatory control structures are usually divided into open- and closed-loop systems. Open-loop systems are the simplest form of control. A block-diagram of the basic open-loop control structure is shown in Figure 2.2. The open-loop systems only use the system model and the current desired state y^{sp} of the system to calculate system input u. Thus, no rejection for disturbances d based on process output y is possible: Contrary to what the definition of process control implies. However, open-loop responses say a lot about how disturbances and process variability affect the plant. This is because the disturbances will have a direct influence on the output y as there are no measurements for impacts on the system, unless this is considered in the model itself, which is not possible under real conditions. Knowledge about open-loop responses is therefore important to study in order to gain knowledge about the system and development of process control structures. [5]

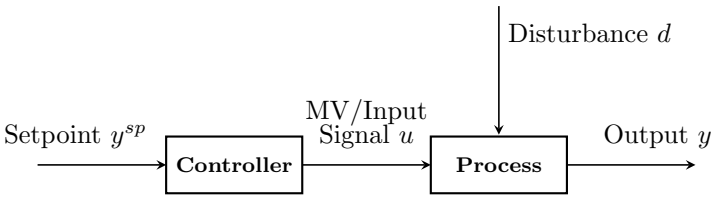

Figure 2.2: Open-loop control structure.

Feed-forward control is a form of open-loop control based on measuring and reacting on disturbances before they affect the system, illustrated in Figure 2.3. This implies that the disturbances must be accurately measured and then future predicted. Often this involves a very accurate model considering the relationship between the process disturbance and output. This makes the feed-forward control very vulnerable to unmeasured and unexpected disturbances. It is usually favourable to use feed-forward

control when the measurement of the output y has a long delay, while the process itself has a short delay. If the process itself involves a large delay, feed-forward control will not be better compared to feedback control. Feed-forward is usually used in combination with a closed loop controller structure. [5]

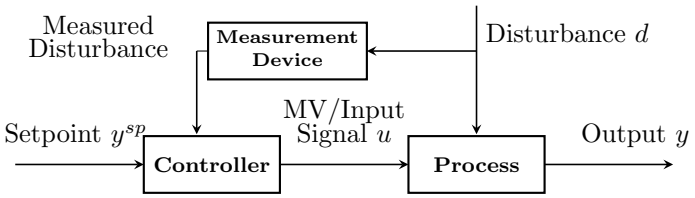

Figure 2.3: Open-loop feed-forward control structure.

Closed loop systems, feedback systems, are based on measuring the process output y. Then the process error e is calculated from the output compared to a reference state y^{sp}, yielding $e = y^{sp} - y$. The system then tries to compensate to make the output equal to the reference input state. That is done using a controller, which sends the input u to the system. Using feedback control yields the advantage that the controller structure is error driven, and no disturbance model is required. For processes where there are unknown disturbances, feedback would therefore be the obvious choice. The obvious disadvantage for this control structure would be that the process only can take corrective measures once the disturbance is detected in the process. For processes with significant dead-time between the measured output variable and the input or manipulated variable, this controller structure could be very ineffective. That is, a process with a large delay in the output. Still, because no actual disturbance model is required, some form of the feedback control system structure is the most commonly used control structure in process plants. Also for this reason, a feedback system will be used to control the model presented later in this project. [5]

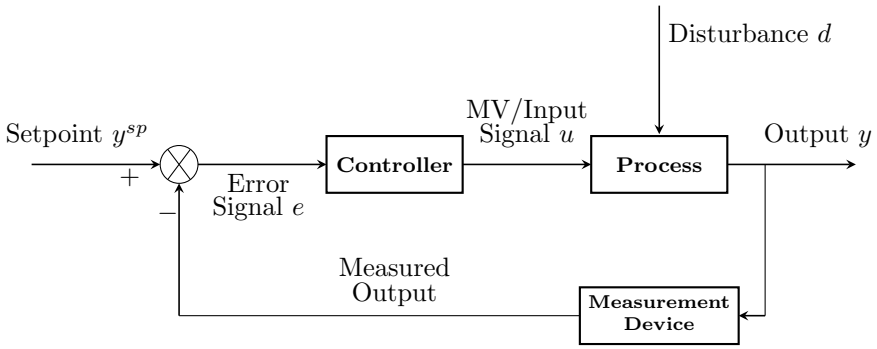

Figure 2.4: Closed-loop feedback control structure.

2.1.1 PID Algorithm

Now that different control loop logic's have been introduced, we can look at actual controller logic. The controller part of a controller structure usually consist of the proportional-integral-derivative (PID) algorithm which has been around since the 1930s. While many vendors have tried to introduce more effective algorithms, the PID-controller algorithm remains one of the most commonly used controller algorithms in the industry today. The PID-controller actually consists of three individual controller concepts, that is a proportional controller, an integral controller and a derivative controller. [5]

The algorithm idea is to add proportional, integrative and derivative action in one controller as a sum, yielding the PID-controller. The combined controller concept on continuous form is illustrated

in Figure 2.5, with the proportional action on top, the integral action in the middle, and the derivative action at the bottom. Each parallel part consists of a tuning part as well as proportional, integral or derivative action to the input error e. The gain decided to which extent the different controller actions should be amplified. A PID-controller could also be represented in a discrete form, but in this project, all controller action will be on continuous form. Continuous PID-control is the most common for feedback systems. The difference in continuous and discrete controllers are how the error signal is processed, and discrete controllers are only recommended if the modelled system is discrete.

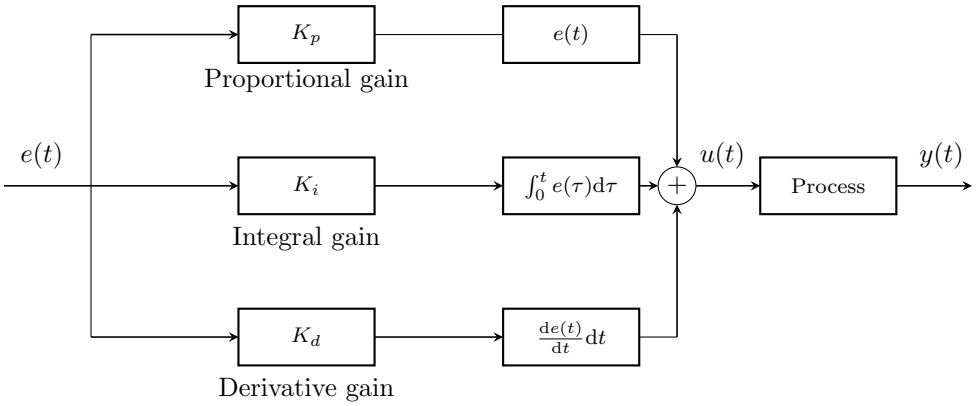

Figure 2.5: Conceptual drawing of a PID-controller structure on continuous form.

Another way to look at the PID-controller is as a sum of independent P-, I-, and D-controllers. The sum of this controller output will be the process input, u. For a continuous controller this would yield the controller expression 2.1. [17]

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\mathrm{d}\tau + K_d \frac{\mathrm{d}e(t)}{\mathrm{d}t} \tag{2.1}$$

In the Equation, e is the controller error from the measured output y subtracted from the reference input y^{sp}. and K_p, K_i, K_d the gains for each of the controller parts. They are tuning parameters, determining how much amplification each controller action part should receive. Often the the gains are rewritten as a function of a joint controller proportional gain K_c, which simplifies tuning. The expression written using the proportional gain K_c is shown in Equation 2.2:

$$u(t) = K_c\left(e(t) + \frac{1}{\tau_i}\int_0^t e(\tau)\mathrm{d}\tau + \tau_d \frac{\mathrm{d}e(t)}{\mathrm{d}t}\right) \tag{2.2}$$

This expression of the PID algorithm gives some new parameters, that is the integral time τ_i and derivative time τ_d. They could be determined experimentally, or using a tuning method. Using a tuning method, they usually represent first and second order time delays in process open-loop responses. Having presented a basic PID-algorithm knowledge, the next sections will go depth for each of them.

2.1.2 Proportional Controller

The proportional controller, the P-controller, is the simplest controller structure algorithm. It involves a linear gain, K_p, amplifying the controller input, the error signal e. The proportional controller output

signal u is therefore proportional to the error signal. The proportional gain K_p is a tuning parameter that must be selected. K_p can have a negative or positive value depending on the process gain. If the process gain is positive, an increase in the value of the controller or measured variable y requires a decrease in the control signal u. This is because we want to counteract the change in the controlled variable y, which moves away from the setpoint y^{SP}. Therefore the controller gain K would have to be positive in this case, because of the way the controller error is defined. This is also called reverse-acting control. Likewise, if there is a negative process gain, y would increase if not a counteracting measure increasing u were to be implemented. Thus, for a negative process gain, the controller gain should be negative. A controller with negative gain is called direct-acting controller. [5]

There are also other important measures to be noted about the proportional controller. An important drawback of proportional control is that it there will always be a residual error between the output and input in processes with compensation. This is as the controller requires an error to generate a proportional output. To overcome this, the idea of adding an integrator to the controller structure was proposed. This resulted in the integral controller.

Another obvious P-controller weakness is that a high proportional gain result in a large change in the controller output for a given current value of the error. High proportional gains should therefore often be avoided, as this could lead to instabilities in the system. [5]

2.1.3 Integral Controller

The output u from the integral controller, the I-controller, is proportional to the integral of the error e with respect to time. This is because the idea of integral action is to eliminate offset caused by the P-controller. The integral ontroller will therefore continue calculation changing the output y as long as error exists. The integral controller is therefore advantageous to use to decrease or eliminate steady-state offset. This determines the speed the measured variables will have towards the setpoint value. The tuning parameter for the I-controller is the integral gain, denoted as K_i. [5]

An obvious disadvantage with integral action is that long integral times will lower the integration effect, thus slowing the controller movement towards the setpoint. Also, integral action is unstable for oscillatory responses. This is as integral action essentially calculates the area of the error from zero, and in an oscillatory response the controller output could therefore increase the oscillatory effect. Another important note is that integral controllers should not be used on their own. Using them alone will possibly cause transient overshoot, saturating the controller output. An integral controller should therefore be used with a P-controller in series, or in a combination as a proportional-integrating controller (PI). [5]

2.1.4 Derivative Controller

The derivative controller, the D-controller, is proportional to the derivative of error e, that is the rate of change of the error. Derivative action is thus intended to be anticipatory. Because the rate of change in the error is what the derivative controller measures, processes with rapid change will have the most to earn from using D-control action. [5] Since most processes at process plants are large processes with slow dynamics, that is one reason D-action is often not required. D-control should be tuned using the tuning parameter K_d.

A major issue with the derivative controller, is that it amplifies any kind of noise signal, even small ones. The noise signals could be disturbances or measurement noise, and inputs as error to the controller. In process plants there are often small measurement noises. These will cause a large effect on the derivative controller, and the controller could end up amplifying the small disturbance, instead of compensating it. This is yet another reason derivative controller action is rather uncommon to be seen in process plants. For plants actually using D-control, mathematical filter action is often applied to the measurements to ensure noise is filtered out, such as Kalman filter action. [5]

Another issue with the D-controllers is that they cannot be used alone. This is due to the fact that if the error is constant, the controllers will produce no output, and no actual error reduction will occur. Thus, the controller should be used as the I-controller, in series with a P-controller, or in a combined proportional-derivative controller (PD). [5]

2.2 SIMC Tuning Method

In many processes, the art of controller tuning is left to trial and error. This remains even due to the fact that over the years, many tuning methods proven for their efficiency has appeared. Examples of some of the more successful tuning methods existing today are the Ziegler Nichols method, the Cohen Coon method or the IMC-method. However, a more recent, simpler and possibly even better tuning method is the Skogestad Internal Model Control (SIMC). Therefore, this project will use the SIMC tuning algorithm. [5] [14]

The SIMC tuning method is a model-based, analytically derived algorithm, and involves only one actual tuning parameter. That is, the desired first order closed-loop time constant, τ_c. The other parameters are decided by approximating to the system process to a first or second order model based on the open-loop step responses for the CVs, to changes in the MVs. Alternatively, a P-controller with a proportional gain of 1 in a closed loop could be used, as this corresponds to the controller essentially giving an open-loop response. By using one of these or other methods, information about the plant gain, k, the dominant time constant τ_1, and the effective time delay θ, can be obtained. These are tuning parameters for the proportional controller gain K_c. The SIMC tuning rule for K_c for a first order process is found in Equation 2.3. [14]

$$K_c = \frac{1}{k} \cdot \frac{\tau_1}{\tau_c + \theta} \tag{2.3}$$

The variables used in the steady state plant gain for a first order process can be found using Figure 2.6. The effective time delay θ is the time from we make a step change on the input until the actual process response. The time constant τ_1 is defined in SIMC as the time it takes for the response to reach 63 % of its new steady state value. Plant gain k is defined as the change in the open-loop response $\Delta y(\infty)$ divided by the input step change Δu.

When tuning a controller for a first order process we may want to include integral action. To do this, the integral time constant must be decided. The SIMC rules states that integral time for tight control should be chosen such that τ_i is chosen as the minimum between the first order time constant, τ_1, and

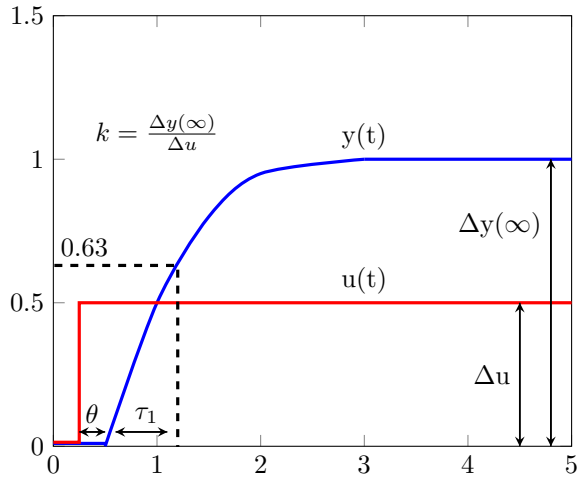

Figure 2.6: SIMC tuning parameters from open-loop response for a first order process.

$4 \cdot (\tau_c + \theta)$. This is expressed as a minimum function, shown in Equation 2.4. [12]

$$\tau_i = min[\tau_1, 4 \cdot (\tau_c + \theta)] \tag{2.4}$$

In addition to first order processes, there exist also higher order processes. For example, including derivative action using a derivative gain requires at least a second order response, where the derivative time constant τ_d will be equal to the second order time constant τ_2 in the process. For higher than second order processes, first or second order approximations should be made in order to use the SIMC rules. This can be done using for example the half rule approximation. There are also processes which are not any of the mentioned processes. Examples of other open-loop responses are integrating processes, static processes, derivative processes and more. [17]

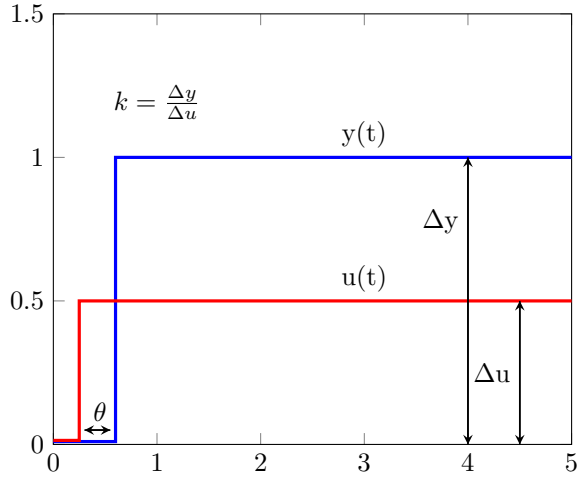

Figure 2.7: SIMC tuning parameters from open-loop response for a static process.

For a static process, as shown in Figure 2.7, the response is equal in shape to the one of $u(t)$, only varying in time delay θ and magnitude. Thus, τ_1 from the proportional gain K_c will approach zero,

making the recommended SIMC parameter for tuning K_c zero, shown in Equation 2.5:

$$K_c = \frac{1}{k} \cdot \frac{\tau_1}{\tau_c + \theta} \xrightarrow{\tau_1 \to 0} 0 \tag{2.5}$$

Having the proportional gain K_c approach zero, means we would need to derive a new relation for the integral gain K_i independent of the proportional gain K_c as the integral gain will not approach zero. The integral gain for a static process is derived in Equation 2.6:

$$K_i = \frac{K_c}{\tau_i} = \frac{1}{k} \cdot \frac{\tau_1}{\tau_c + \theta} \cdot \frac{1}{\tau_1} = \frac{1}{k} \cdot \frac{1}{\tau_c + \theta} \tag{2.6}$$

The parameters for the integral gain K_i is obtained as shown in Figure 2.7. For k, we want to use the relationship between the magnitudes of $\Delta y(t)$ and $\Delta u(t)$. θ is the effective time delay while the parameter τ_c is still a tuning parameter which must be chosen.

For choosing the tuning parameter τ_c in processes requiring tight control, τ_c is often set equal to θ. For more robust control, τ_c should be given a larger value than θ.

2.3 PID Performance

Knowing the important basics of PID-controller structures, all should be good? Or? Actually, this is most often not the case. PID-controller performance is often not perfectly satisfactory, and we must settle with acceptable performance. The meaning behind this is that perfect control, keeping perfect track of our desired setpoint y^{SP} is not possible. When using feedback control structures is not possible, especially those with significant dead-time, disturbance is not detected before the process output. Thus, some error will have to occur before the controllers can take correcting measures, and perfect setpoint tracking can thus never be achieved. It must therefore be determined what is satisfactory performance. If the controller performance turns out to be unsatisfactory, this is a product of the process equipment itself, the plant part. Considerations must therefore be made to the plant itself if controllers does not perform to a sufficient satisfaction, such as relocating measurements, valves, re-configuring flow tubes etc. This could help dampen disturbances and yield better control.

If it turns out the plant is input-output controllable using existing control configuration and the performance is still insufficiently good, this could be controller related. This involves both tunings or the controller structure itself, which will be discussed later. Say, for a P-controller that the P-gain is too low, the system response could be sluggish. If the gain is to high, the controller could make the system overshoot and oscillate, making the system unstable.

Another important controller related issue to address is saturation. When a MV such as a valve is either fully open or fully closed, it can no longer change to reach the setpoint for its CV. Thus, further control is no longer possible. Saturation limits also involves windup problems for controller structures with integration action in the controller loop.

2.3.1 Windup and Anti-Windup

Windup is the action of the controller output u increasing or decreasing because of integral action. When the output u is maximized or minimized, saturation occurs. In this project, the controller inputs all range from 0-1 for z_1, z_2, z_3 and z_4. In Simulink saturation limits can be given by a Saturation block. The problem with saturation limits in a dynamic solver is that integrating action will continue to occur even when the controller is saturated. This will cause a problem, as the PID-controller value for the valve will continue to increase or decrease, even though saturation is met and limited in the output u. The continued integration will lead to delay, as the controller must integrate down or up from whatever theoretic value is obtained, causing a potentially huge delay in the controller output. The problem solution is referred to by the name anti-windup. There are several ways to implement anti-windup, but this project will use the back-calculation method implemented in MATLAB Simulink software. The back-calculation method uses a feedback loop to discharge internal integrator within the PID controller when the controller hits the given saturation limits and enters non-feasible operation. The block diagram for Simulink back calculation is given in Figure 2.8.

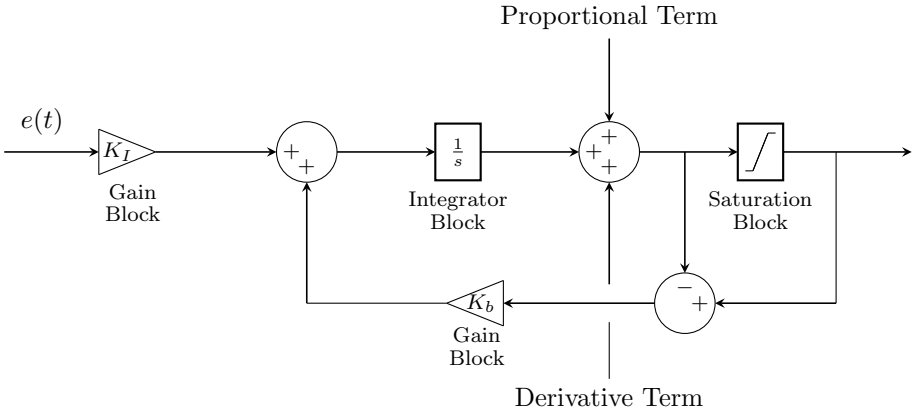

Figure 2.8: Illustration of the back-calculating anti-windup concept in Simulink.

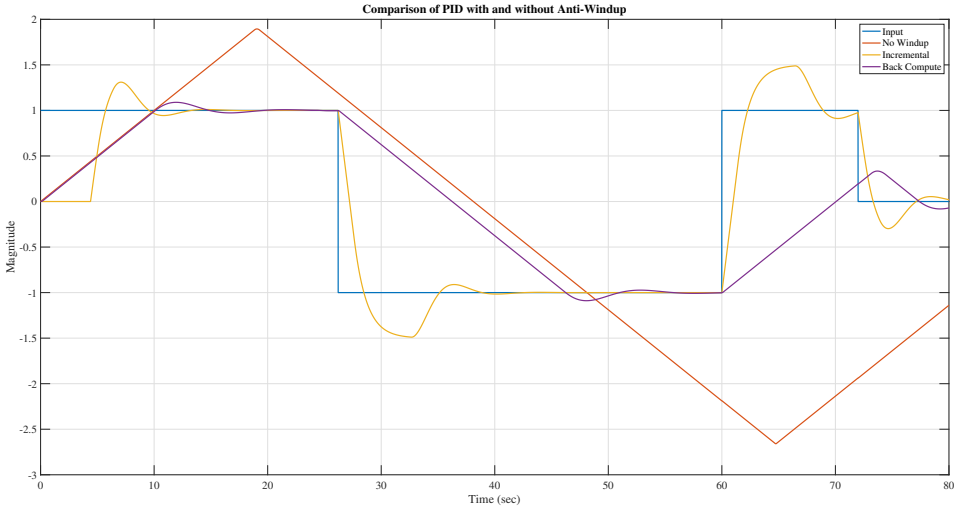

Figure 2.9: Plots showing the response for a controller going from a saturated state to a non-saturated state, showing the input, controller output without anti-windup, with incremental and back calculating anti-windup [4]

In Simulink, the concept of back-calculation anti-windup illustrated in Figure 2.8 is implemented in the internal part of the PID-controller, thus no actual implementation is required except for choosing the feedback gain K_b, which is often set equal to the integral controller gain K_i. This way, Simulink will try to minimize the difference in the integral calculation and output from the saturation limiting block, ideally making it zero. This means no actual overshoot is caused internally in the controller, and when saturation limits again are not reached, the controller will respond much faster to the change. [2]

Another anti-windup method is the incremental algorithm. It calculates a control increment $\Delta u(k)$ at each sampling period and adds to the previous signal, $u(k-1)$, only adding the amount not saturating the actuator. [2] Figure 2.9 shows the actual input u from the saturation block, the internal controller output with no anti-windup, the internal controller output with incremental anti-windup and with back-calculating anti-windup.

2.4 Supervisory Control

Proper PID-controller selection and tuning is an important issue when optimizing the supervisory control layer from Figure 2.1. The main purpose of the supervisory control layer is to keep the outputs at optimal setpoints using degrees of freedom for the regulatory layer and any unused MVs. An important issue here is whether to use a decentralized control structure or multivariable control. Decentralized control is preferred for non-interacting processes, and cases where active constraints remain constant. Thus, this project should be well suited for the use of decentralized control structures, assuming proper CV-MV pairings to minimize interaction. [11]

A number of decentralized control structures exist, therefore only a selection proved for efficiency is considered in this project. The structures that will be considered are default control, parallel control using equal and different setpoints, split range control, valve position control and droop control.

Default control focus on simple control only, involving each MV controlling only one CV in a feedback loop. This structure is usually sufficient for small disturbances and systems operating near steady state. For larger disturbances, this structure will usually not be sufficient enough.

Parallel control using equal and different setpoints are different options for controlling the same CV using different MVs. Parallel control with equal setpoints to control the CV utilizes the same CV setpoint for the different controllers controlling the same CV. Therefore, I-action is only possible in one of the controllers, as multiple integral action at the same time would possibly cause a process disturbance. When using different setpoints, multiple I-controllers are possible, as all the MVs are controlling the CV using different setpoints. The use of different setpoints for different MVs controlling the same CV will cause only one integral action to be active at a time, because only one controller will be active at a given time. [9]

Split range control involves the use of selectors and a split range block Split range control involves multiple MVs controlling one CV. The logic ensures active control on only one MV at a time. This could be advantageous for maintaining a setpoint, though at the cost of MVs saturating to maintain the CV setpoint. Careful consideration and prioritization is therefore required before this kind of implementation. [9]

Valve position control reminds of split range control, where one valve position controller uses two MVs to control one CV. Here also, one MV at a time is controlled until saturation in order to satisfy CV

setpoint. [9]

Droop control is probably the most flexible method. In electrical grids it involves proportionally allocating supplier production according to demand from consumers, but could also translate cases such as gas networks by being defined as a proportionally allocated reduction in consumer demand for supplier disturbance. For this structure, all MVs are controlled at the same time, with the MV setpoint being proportionally allocated according to what CV is prioritized. [16]

Chapter 3

Process Modeling

3.1 Process description

Since steam generation and distribution networks often lack proper optimization, there is much to gain from looking into the network operation, possibly increasing the plant efficiency. Many types of distribution networks exist, but they are commonly based on the same principle. This principle involves a high pressure supplier grid. The high pressure grid is then supplying network consumers through medium-pressure networks which could include splits to even lower pressure networks. In the United Kingdom, this grid system is found in consumer gas distribution networks, where the low pressure consumers essentially consist of hundreds of thousands of households across the country. There is no gain in controlling the hundreds of thousands of household consumers, but balancing the medium pressure supplier demand with the supply could definitely improve stability as well as predictability. [8]

This project will model a typical, simplified steam distribution network commonly found in industrial process plant. The model will include only three consumers sharing the same supplier, shown in Figure 3.1. This network includes a high pressure supplier, P_1, and three low pressure consumers P_2, P_3 and P_4 with corresponding flows q_1, q_2, q_3 and q_4, respectively. To gain the ability to possibly regulate both supplier and consumer side in the system, valves for both the supplier and consumer side were added. The valve opening position ranges from zero to one and is denoted by z_1, z_2, z_3 and z_4. Between the consumers and the supplier the main system pressure is defined as P and the corresponding main flow as q.

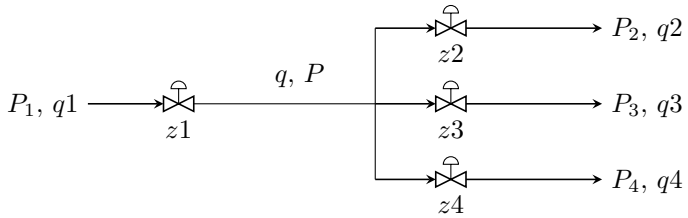

Figure 3.1: Illustrating of a simple gas network with three consumers sharing the same supplier, with possible process variables illustrated.

In gas networks comparable to the one in this project, steam supply is expected to be mostly at steady state. This means the consumers can expect a somewhat constant supply flow as well as

demand. That is, unless a critical disturbance occurs forcing a supplier pressure drop. For a steam network, this could be caused by multiple reasons, in the form of a supplier disturbance or a consumer disturbance. [8]

For this project, a sudden drop or increase in P_1 is projected to be the main disturbance to steady state operation. This means that the system conservation principles are not fulfilled, meaning that accumulation occurs within the system. The main accumulation in this process would be caused by a pressure drop in P_1. This would cause the main flow q to drop, again leading to a drop in consumer flows. Alternatively, a positive disturbance from P_1 could lead to a positive accumulation and thus an increase in consumer flows.

In a real system, disturbances from the consumer side through P_2, P_3 and P_4 would also be affecting the steady state conditions in the system. However, they do not affect the system to the extent a disturbance from P_1 would and are therefore left out as critical disturbances for this project.

In order to keep the system as steady as possible, controller structures should be implemented. However, the system MVs and CVs must first be determined.

To keep the simulations consistent, all parameters and nominal values for the system should be kept equal for all simulations. The most important system parameters and nominal values are presented in Table 3.1. Notice that the valve dimensions for outflow valve z_2 are smaller than the other outflows z_3 and z_4. The nominal flow are also smaller for q_2 compared to q_3 and q_4. This is because for many decentralized control structures, it would be preferable to reduce the flows of the less important consumers to extend operating range for the prioritized consumers, in this project z_2. If the flows that can be given up are larger than the prioritized flow, we obtain more flow to give up, which would be an advantage for the control structures using MV-MV switching. That is, structures where one MV is used until saturation before the next MV is used to control for a given CV.

Table 3.1: A complete depiction of the simulation parameters and nominal values for MVs and CVs.

Variable Name	Description	Value	Unit
T	System Temperature	250	[°C]
R	Universal Gas Constant	8.314∗10e-5	[m^3atm/kmol]
V	System Volume	150	[m^3]
$P1$	Inlet Pressure	15	[bar]
P	Nominal Main Pressure	10	[bar]
$P2$	Outlet Pressure 1	2	[bar]
$P3$	Outlet Pressure 2	2	[bar]
$P4$	Outlet Pressure 3	2	[bar]
q	Nominal Main Flow	2	[kmol/s]
q_2	Nominal Flow 2	0.5	[kmol/s]
q_3	Nominal Flow 3	0.75	[kmol/s]
q_4	Nominal Flow 4	0.75	[kmol/s]
Cv_1	Valve 1 Coefficient	0.3578	[bar kmol/s]
Cv_2	Valve 2 Coefficient	0.1021	[bar kmol/s]
Cv_3	Valve 3 Coefficient	0.1531	[bar kmol/s]
Cv_4	Valve 4 Coefficient	0.1531	[bar kmol/s]
z_1	Nominal Valve Opening 1	0.5	[-]
z_2	Nominal Valve Opening 2	0.5	[-]
z_3	Nominal Valve Opening 3	0.5	[-]
z_4	Nominal Valve Opening 4	0.5	[-]

3.2 Model Assumptions and System Considerations

For network modeling some considerations had to be made. First, and most importantly the gas network is assumed to be represented by water in pure gas form (steam) for modelling simplicity. This is ensured by keeping the system above its dew point, that is the point where the first drop of condensate is formed. One way to ensure this is by using a temperature-pressure diagram, such that the system temperature in the system is set above dew point according to the pipeline with the highest pressure. This is done because the pipeline segment with the highest pressure will also have the dew point occurring at the highest temperature. To ensure a temperature above the dew point at 15 bar pipeline pressure, a temperature of at least 198 °C should be maintained. As a buffer because of possible positive pressure disturbances in the system the actual modeling temperature were set slightly above this, to 200°C. For a real system requiring pure gas without condensate, keeping close to the dew point would also be realistic, as keeping a higher system temperature would require higher heating costs. However, this is a trade-off: Staying closer to the dew point conditions would make the system more vulnerable to disturbance such as pressure increase or drop. [6]

The reason why we model a system in pure gas form is that is follows that we can assume that the equation of state for an ideal gas is valid. The ideal gas low is used to relate pressure P and volume V with temperature T, the amount of gas in moles n, using the universal gas constant R in $m^3atm/Kmol$ to relate the equation variables. The relation is presented in the network model section.

The basis for using the ideal gas law is that the velocities and pressures in the flow network is assumed to vary only in the flow direction, with no condensate forming within the control volume. No condensate formation is ensured through maintaining the system conditions above dew point conditions. Variation only in the flow direction is a direct assumption, but for large flows this should yield an appropriate approximation.

For convenience and model simplicity, it is also assumed that the system is isolated, with no heat loss from the system to the environment is assumed. The valves in the system are also assumed all to be linear valves. This simplifies the relation for the valve pressure drops and the system pressures in the modeling section.

3.3 Network model

Knowing the system variables and model assumptions, a mathematical model based the objective requirements could be obtained. The objective states that a relation between the flows, pressures and valves in the system should be obtained while maintaining the physical constraints for the system. This system only involves one person-made physical constraint, that is, the max flow in the main system, $q^{max} = 2.2$[kmol/s]. Other physical constraints would also be present, but they would come natural from physical limitations when modeling of the system, such as a minimum of 0 for all the flows and pressures, and a max of 2.2[kmol/s] for any of the consumer flows following from q^{max} when consumer pressure is constant.

In any of the different control controller structures to be used, the mathematical model with physical constraints will be the same since different control systems are to be modeled, whereas the mathematical model only represents the system itself with no control. Skogestad proposes that we need to consider two things when modelling a system, that is,

1. What is the control volume of the system? [13]

2. Which type of balance should we use (Mass, component or energy)? [13]

As we are looking at the overall system in Figure 3.1, this will obviously be our control volume. Also, from the Control Objective Section, since we want to control the outlet consumers, we should use a dynamic mass balance. Considering the system the dynamic mass balance with the flows we are interested in will be represented by the following equation:

$$\frac{dn}{dt} = q - \sum_{i}^{N} q_i \qquad \text{for i = 1,...,N [13]} \tag{3.1}$$

Where n is the holdup in kmol, q is the consumer flow in kmol/s, and q_i represents the flow from outflow i. For the system modeled in this project, q would be the molar flow out from the supplier valve, while q_i would consist of q_2, the molar flow from consumer valve 2, q_3, the molar flow from consumer valve 3, and likewise q_4 would represent the molar outlet flow from consumer valve 4. Thus, $\frac{dn}{dt}$ represents the total molar mass flow change per time. Since the flows should be related to the pressure P, the equation of state for an ideal gas should be used. The equation of state for an ideal gas is expressed in Equation 3.2:

$$P = \frac{nRT}{V} \tag{3.2}$$

The ideal gas law expresses the pressure P in terms of temperature T, system volume V and the number of moles particles n in the system using the universal gas constant R to relate it. Using the equation of state for an ideal gas, the molar mass balance can be reformulated in terms of pressure and the flows relevant from the problem formulation, expressed in Equation 3.3:

$$\frac{dp}{dt} = \frac{RT}{V} \cdot (q - q_2 - q_3 - q_4) \tag{3.3}$$

Now, an expression for the system flows q, q_2, q_3 and q_4 are required. They can be related using the valve equation, which is presented in Equation 3.4:

$$q = C_v \cdot f(z) \cdot \Delta P = C_v \cdot f(z) \cdot \sqrt{\frac{(P_{in} - P_{out})}{\rho}} \text{ [7]} \tag{3.4}$$

Equation 3.4 raises some new parameters to the problem. ρ is actually a simplification of the equation and represents the average density between the flow before and after the valve. C_v, the valve coefficient is a constant parameter representing the valve flow capability. That is, a large coefficient would mean a larger flow at given pressure differential. Traditionally, a C_v value of one is defined as the C_v value required to flow one gallon per minute of water at 60°F with a pressure drop of one PSI. The value for C_v is therefore often chosen based on system size, which would also be the case in this project. P_{in} is defined as the pressure from the flow before the valve, while P_{out} defines as the pressure after the valve. The remaining coefficient, $f(z)$, is the valve characteristic. There are different types of valves with different characteristics, but for this project, linear valve characteristics are used for simplicity. This is the simplest form of valve dynamics, with $f(z) = z$, meaning the valve dynamics are taking

a linear value between $[0, 1]$. This value represents the opening state of the valve, with 0 being fully closed and 1 being fully closed. [7]

Now, using the assumption that an average density can be used for the pressure drop in the valve, along with the assumption that the system follows the ideal gas law principles, the density can be left completely out of the relation. This yields a relation with the flow only depending on the pressures entering and exiting the valve as well as the valve opening. The general, simplified, density-independent expression for the valve equation is presented in Equation 3.5.

$$q_i = C_{v,i} \cdot z_i \cdot \sqrt{P_{i,in}^2 - P_{i,out}^2} \text{ [7]} \tag{3.5}$$

The relation in Equation 3.5 yields the algebraic equations for the flows q, q_2, q_3 and q_4, presented in Equations 3.6-3.9.

$$q = C_{v,1} \cdot z_1 \cdot \sqrt{P_1^2 - P^2} \tag{3.6}$$

$$q_2 = C_{v,2} \cdot z_2 \cdot \sqrt{P^2 - P_2^2} \tag{3.7}$$

$$q_3 = C_{v,3} \cdot z_3 \cdot \sqrt{P^2 - P_3^2} \tag{3.8}$$

$$q_4 = C_{v,4} \cdot z_4 \cdot \sqrt{P^2 - P_4^2} \tag{3.9}$$

This section has now presented a relation for the flows q, q_2, q_3 and q_4 using pressure differences, as well as a dynamic mass balance in terms of pressure. The mass balance in terms of pressure is an ordinary differential equation, while the flow equations are algebraic equations as a function of the pressure solution of the differential equation. In order to solve the system, a numerical solver should be used.

3.4 MATLAB and Simulink Implementation

For this project, the programming language MATLAB along with its Simulink extension were chosen to simulate and solve the system model for a given time frame. This could also be done using an open source programming language such as Python instead. Using MATLAB and Simulink, the open loop network model was implemented around the MATLAB differential equation solving software to obtain open loop system responses. Then, control loops were built and tuned around the open loop responses using SIMC tuning rules and later proposed controller structures. The control loops were made such that Simulink Gain blocks decides which controller structure is active.

3.4.1 Open Loop Solver

The network model was built using the illustrative model shown in Figure 3.2. First, a MATLAB script including the differential equation from Equation 3.3 and the algebraic valve Equations 3.6-3.9 was built as a basis for the MATLAB solver. The MATLAB Model was then imported to Simulink using the Simulink Model block. From here, the equations were split into the differential and the

algebraic equations in the system using Simulink Gain blocks. From here, the MATLAB solver could integrate the differential equation solving for pressure and solve the algebraic equations for flows in parallel. From the solver, the output would therefore be two fold, that is values for P, q_2, q_3 and q_4 would be given for each time step solution of the equation. For the differential equation solver to work, a minimum of one input and one output were required. At least one output is fed back as the current system state x. To solve the system, MATLAB standard differential equation solver ode45 were used. This proved sufficient for this system as is was small and non-stiff, but this solver could prove issues for larger, stiff systems.

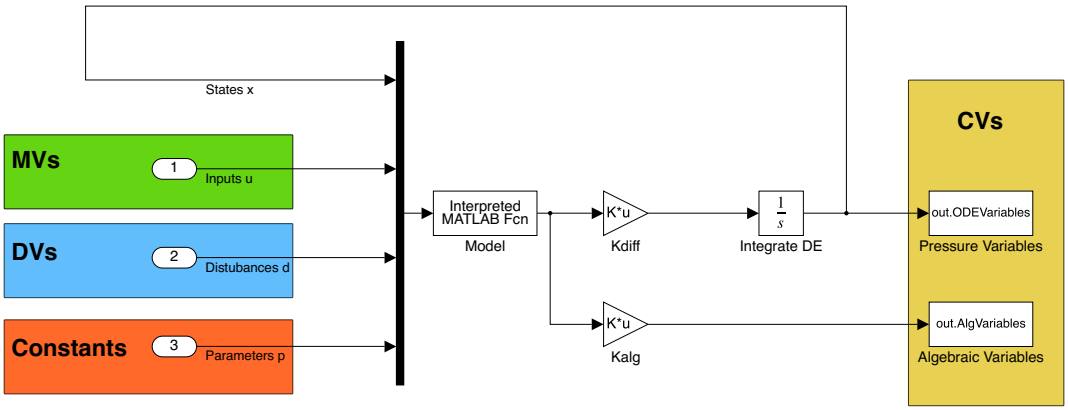

Figure 3.2: Illustration of the open loop system in Simulink, using the states, MVs, DVs and constants as inputs, outputting states, CVs.

Translating to the control objective, the input, u, would be the MV value, while the output y would be the current CV value. At least one CV value would then have to be fed back to the system as a state. In addition to the MV input and CV state value, the differential equation was set up such that disturbances and constant parameters were also inputted to the solver. Thus, the differential equation along with the algebraic equations were stated as a function in MATLAB, with the function being $f(x, u, d, p)$ involving the states x, inputs u, disturbances d and the constant parameters p.

3.4.2 Closed Loop Solver

Knowing that the open loop system solver worked as expected, closed loop control structures could be built around it. Figure 3.3 represents a system with a PI-controller using z_1 as a MV and P as a CV.

In order to manipulate z_1 dynamically, a controller outputting a value to z_1 would be required. But, in order for the controller to give an output, it would also need an input. The input to a feedback controller is the offset e between the setpoint value for the CV to be controlled and its measured value. This is illustrated in Figure 3.3. Here, the CV setpoint is inputted along with the calculated value for the CV, in this case P. A tuned controller would then manipulate z_1 in order to minimize e towards zero. Note that for a saturation in z_1, control for the CV would be lost. Also, for normal operation, the controller input would be be zero in Simulink. Therefore, a Constant-block were added, inputting the nominal value for z_1 such that for normal operation, the valve would be at nominal position.

In the actual simulated system controllers for the three consumer valves were also added. Simulink Gain blocks were used in order to select what CVs the controllers should maintain. The complete Simulink model and MATLAB files along with explanation can be found in the Appendices in Figure 4.

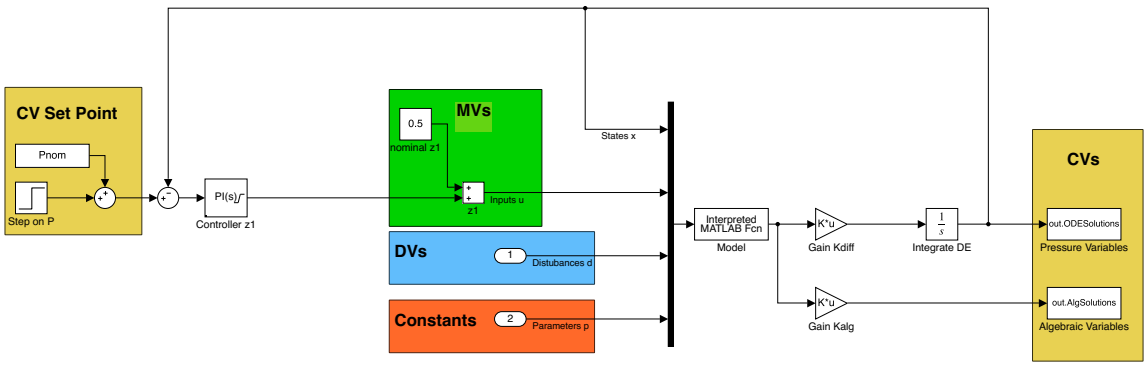

Figure 3.3: Illustration of the closed loop system model in Simulink, using the states, MVs, DVs and constants as inputs outputting states, CVs.

Chapter 4

Control Structures

4.1 Control Objectives

In order to provide a sufficient supervisory control structure, control objectives must be defined. Since the focus of this project remains on the decentralized control structures in the supervisory control layer, the control objective involves keeping the primary controller outputs, that is the primary CVs, at optimal setpoint using unused degrees of freedom, that is MVs, to control the secondary CVs, that is setpoints for the regulatory control layer.

This project disposes the use of both the consumer and supplier valves for control of the supervisory layer. Since the dimensions of the system in this project totals four unused dynamic valves, that would mean the system includes four unused degrees of freedom.

The optimal system goal is to maintain the consumer flow q_2 and if possible also maintaining the main pressure P, at their given setpoints. Since the flow constraint is the most important, it will always be active. Therefore, maintaining q_2^{SP} is defined as the main CV constraint. The secondary constraint the system faces is maintaining the system pressure at P^{SP} to ensure stability, both in the main system, but also to consumers. This is also a CV constraint, but regarded less important than the flow constraint, so it is a soft constraints. This means the constraint is not required to always be active, it can be given up if needed. The same applies for the setpoints for q_3 and q_4, they are soft constraints which can be given up in order to maintain the primary and secondary objective of maintaining q_2^{SP} and P^{SP}, respectively.

Since the outflow q_2 and main pressure P are prioritized, the control structures should build around satisfying those constraints. This poses some possibilities for which MVs and CVs to use in the system. There are three possible scenarios for choosing MVs and CVs, involving only supplier control, only consumer control and combined supplier and consumer control. Since supplier control would only represent a simple feedback structure, it does not classify as an advanced controller structure and the use of supplier control alone is not considered in this project.

The next case would involve using only consumer control. The idea is suited because of the main system disturbance P_1. A critical disturbance from P_1 implies the supplier side would likely saturate for a critical disturbance whether it is controlled or not. Thus, controller structures can be made that are simpler in implementation because one less MV is present in the system. It could also be assumed

that the supplier valve is fixed, thus the supplier valve would not represent a degree of freedom.

Even though consumer control possibly poses somewhat simpler structures, combined control should be used if possible. The reason for this is that for normal operation, which usually involves the majority of operation time for a process plant, the supplier valve would not be saturated and should be utilized for optimal CV control. In this project, however, the disturbance for P_1 is set so large that for combined control the supplier side will saturate. Saturating the supplier valve is done to get comparable results for the cases where only consumer control is used and the combined supplier and consumer control is used.

The reason both consumer control and combined consumer and supplier control are not used in every simulation is to simplify controller implementation for some of the more advanced decentralized controller structures. For some controller structures, controlling both consumer and supplier would yield an implementation complexity that there were simply not enough time for to implement in this project. However, since the supplier is always saturated for a critical disturbance in P_1, all results in this project should still be comparable. In the next section, the possible MV and CV selections for controlling only the consumer side and controlling both supplier and consumer side are presented.

4.1.1 Consumer Side Control

The idea of only using consumer side control is that a simpler decentralized controller structure can be achieved since P_1 will cause saturation in z_1 anyway. As shown later in the project, this is not necessarily the case, and in reality, it could be smart to control both consumer and supplier side. There could also be other reasons forcing control of consumer side only such as the supplier valve opening being fixed, thus not representing a degree of freedom, so this section aims to assess what MVs and CVs should be selected when for any reason only consumer side control is available.

The consumer side control would yield two CV structure possibilities following Skogestads pair close rule. That is, pairing the MVs, the consumer valve openings, with the main system pressure or the outlet flows. Reminding ourselves that the most important active constraint is the one on q_2 and using the pair close rule, it was determined that a good MV-CV pairing would be the outflows flows as CVs for the corresponding valves as MVs. This way the less important CVs can be given up to keep the setpoint for q_2. Consumer control for the flows yields the MVs and CVs presented in Figure 4.1.[12]

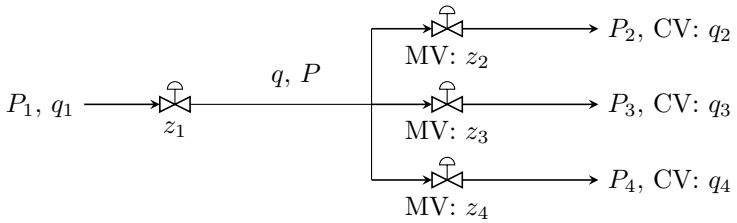

Figure 4.1: Illustration a possible subset of MVs and CVs for consumer side control.

The structure in Figure 4.1 represents the system CVs and MVs for control structures split range control using selectors and valve position control using selectors. It should still be noted that supplier control could be added to those control structures given that the supplier valve is a unused degree of freedom. However, as will be shown in the next section, these control structures are already posing a high level of implementation complexity with only consumer side control.

Though the idea of using consumer control may sound like the most feasible, it would also be possible to use the pressure as a CV for multiple MVs single CV control structures. The idea is that as long as the CV, P is kept at a setpoint, the most important constraint, q_2^{SP}, will be kept at setpoint. The concept with system MVs and CVs are shown in Figure 4.2. Examples of control structures utilizing these MVs and CVs go under parallel control with equal setpoints and parallel control with different setpoint.

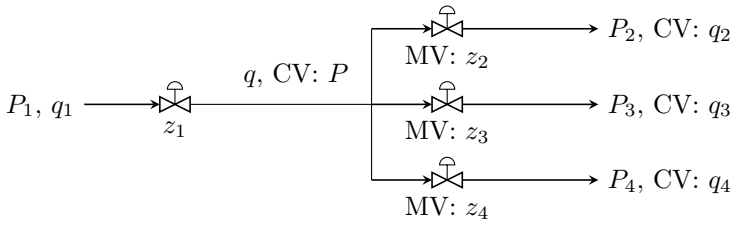

Figure 4.2: Other illustration a possible subset of MVs and CVs for consumer side control.

4.1.2 Controlling both supplier and consumer side

The more realistic case for the problem in this project would be controlling both the supplier and consumers for normal operation. For this combined control, the consumer side control would be equal to the one in control for consumer side only.

The difference for this structure would be that also the supplier valve z_1 could be utilized as a MV. Following Skogestads rule about pairing close, where MV-CV pairs should be chosen such that effective delays and loop interactions are minimal, we should always pair the supplier valve with P.[12] Since the flows from the consumer valves are not really directly related to the supplier valve the only other viable option would be choosing q as a CV. Still, this would not make sense because P is prioritized above q. Also, controlling the supplier valve using P_1 or q_1 would make the system infeasible in terms of controllability and radiating rules for controller placement.[12]

Thus, the possible MV-CV structure would be either the supplier MV pairing on the main pressure P, and the consumer MVs pairing on their corresponding flows or the main pressure in the system, depending on the controller structure. The possible MVs and CVs in the system are shown in Figure 4.3. The control structures utilizing both supplier and consumer control setup of MVs and CVs in this project are default control for both consumer and supplier side, and droop control.

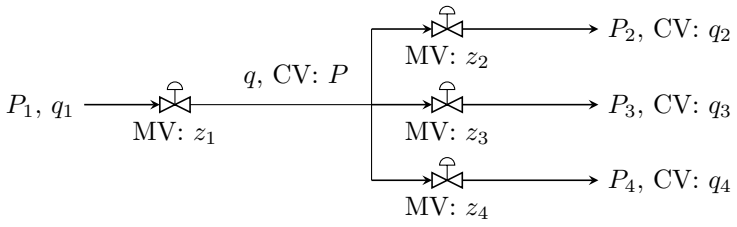

Figure 4.3: Illustration a possible subset of MVs and CVs for combined supplier and consumer side control.

All three cases presented with different CVs and MVs will be used in the control structures presented in the next section. It shows that even for a simple system with limited degrees of freedom, many choices for control are available. Thus, the point of obtaining a systematic approach for decentralized control structure design enlarges.

When implementing decentralized controller structures in the supervisory control layer, a sufficient logic in controller structure must be implemented to avoid unwanted interactions. Because it is possible for multiple controllers to control the same CV, this could pose problems if the interactions between those controllers try to optimize the CV differently. This could lead to abnormal plant operation. For centralized control structures, this is usually not a problem as the centralized controller will work to avoid these interactions. For decentralized control structures, however, avoiding unwanted controller action is important to keep in mind when constructing controller logic as there is no centralized control working to avoid interactions. Having assessed what CVs and MVs should be used to control our system, we can now move on to the actual controller logics.

4.2 Decentralized Control Structures

The next sections looks at the implementation specifics for each of the proposed controller structures from the previous section, adapted to the problem this project presents. The logic descriptions are supported using block diagram logic schematics.

4.2.1 Default Control Using Consumers

The first implementation that should be done is the default controller structure, used as a basis underlying the more advanced decentralized controller structures. This structure would be used as a basis to obtain open loop responses, tuning parameters etc, as the structure was not expected to be able to prioritize q_2 over the other consumer flows. Default control using only the consumers involves the controller structure presented in the block diagram shown 4.4. Integral action controllers should be chosen as the expected open loop responses for the flows would be static responses. The default controller structure for consumer control utilizes z_2, z_3 and z_4 as MVs, controlling their corresponding CVs q_2, q_3 and q_4. The structure involves three basic feedback loops for the integral flow controllers.

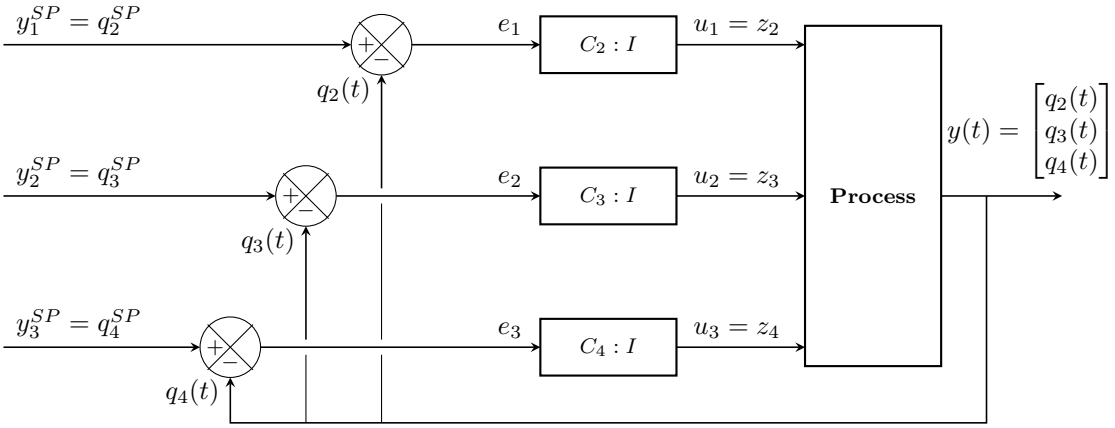

Figure 4.4: Block diagram showing controller logic for default control using only consumers.

In the block diagram, the output representing the system CVs from the process is represented as a vector. The remaining block diagrams will also present the process output as this. The inputs, that is the MVs, is presented as variables of u_i for this block diagram. Knowing this, and keeping the default controller structure for the consumer side in mind, a model for default control using both the supplier and consumers were developed.

4.2.2 Default Control Using Suppliers and Consumers

Building on the structure using default control on consumer side only, default control using both supplier and consumer control were constructed. This control structure would therefore likewise use the MVs on the consumer side to control their consumers as CVs, as well as using the added supplier MV to control pressure as a CV, thus using all available MVs. The controller structure can be observed in a block diagram for 4.5.

From knowledge about open loop step responses for pressure it would be expected that the response for the pressure P will be a first order response. Thus, a proportional integral-action should be used for the controller C_1. Still, integral controllers would be used for the flow controllers, now denoted as $C_1 - C_3$.

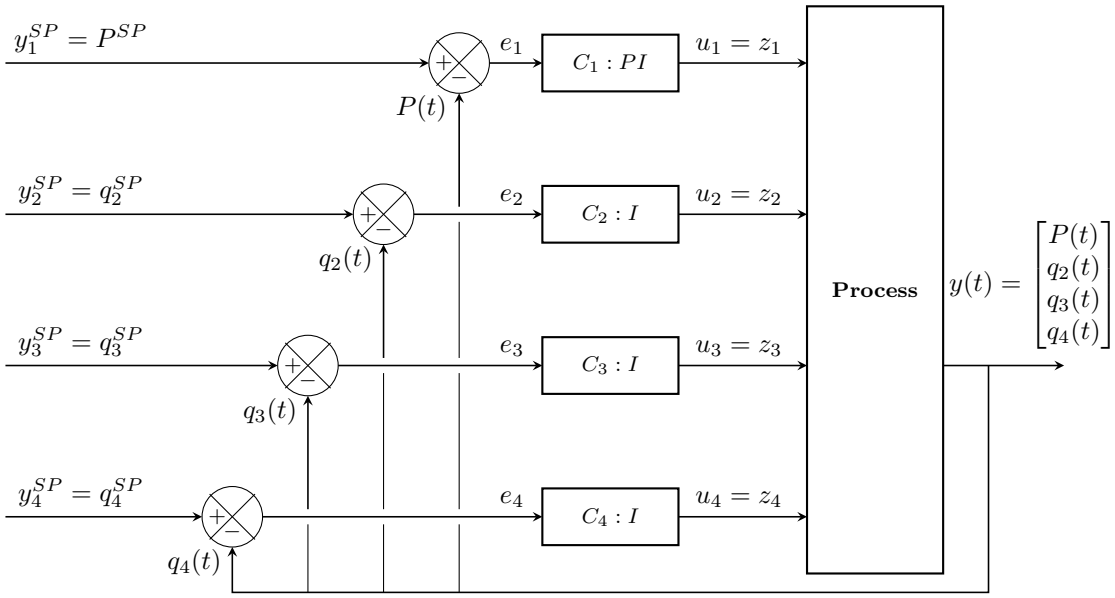

Figure 4.5: Block diagram showing controller logic for default control using both suppliers and consumers.

Because the two variations of default control essentially would be the same controller for a critical disturbance in the inlet pressure P_1, saturating the supplier valve z_1, only implementation of the latter would be performed. This is to better get an idea of how the system would react also before P_1 would saturate.

4.2.3 Parallel Control

Parallel control is a better alternative to default control in terms of possibly extending the operating range and keeping q_2 and P at desired values, while giving up the less important consumer flows. This is because default control would have no actual way of prioritizing and giving up specific consumer flows above others, while as this in theory should be possible for parallel control structures.

The method involves controlling the same CV using multiple MVs, that is for this project the idea would be to control P using the consumers, in the thought case of the supplier z_1 loosing control of P. This control structure would in theory be most ideal for maximizing throughput for temporary or moving flow constraints. Parallel control is divided into two methods, equal setpoints and different

setpoints. Both use different variations of the same idea, to control one CV using multiple MVs. The idea for parallel control is to control the CV using the same setpoint, as opposed to different setpoints, where the idea is also to control one CV using different setpoints for the given CV. [9]

Equal Setpoints

The concept of parallel control using equal setpoints is illustrated in the block diagram in Figure 4.6. Because the idea is to control P using the consumer valves, the response is expected to be first order. Proportional action is therefore required for all the controllers. Static integral action should also be added in one of the controller loops, in order to eliminate steady state offset. The reason static integral action is not desired for all the controllers in this controller structure, is that the same setpoint on the controllers would cause integral action to be active at the same for all the controllers. Only one integral action should be active at a time for a given process, or unsteady operation and interacting integral action could occur.

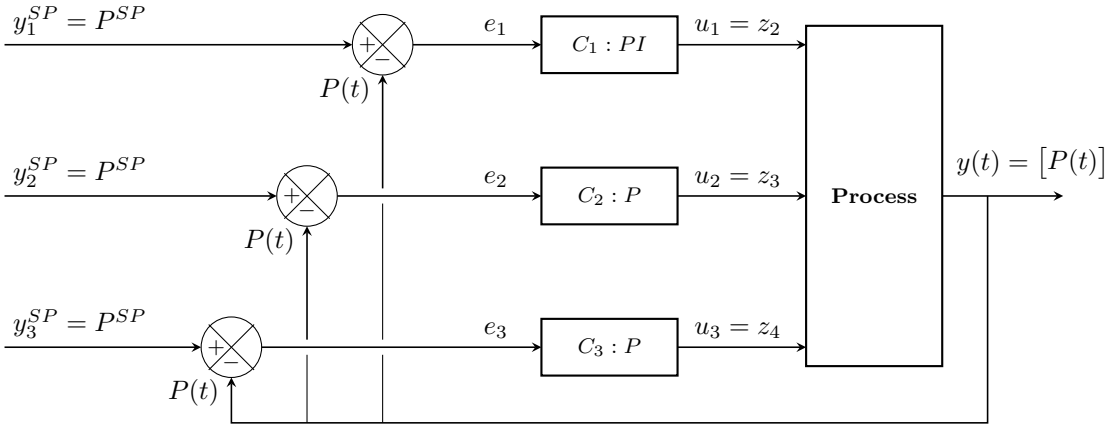

Figure 4.6: Block diagram showing controller logic for parallel control with equal setpoints on consumer side.

Different Setpoints

Using parallel control with different setpoints also involves one controller for each MV in the system, all set to control the same CV. Thus, control using different setpoints will resemble the parallel control structure, separated by the fact that the CV now should be controlled using different setpoints for each controller. The effect aims to resemble MV-MV switching. MV-MV switching means that at any given time, one MV will be active to control a given CV. When the MV saturates, a new MV will start saturating. Thus, one MV at a time will saturate, extending the CV operating range until possibly all MVs are saturated. The use of multiple controllers in this way, yields the advantage that each controller can be tuned independently.

Implementation is achieved by first tuning one controller at the desired CV setpoint, in this project that would be P. The idea of using P as the CV instead of q_2, is that when maintaining P at setpoint by giving up certain flows, q_2 would be ensured at nominal value, thus being prioritized.

The setpoint for P in the first controller would denote $P_{c,1}^{SP} = P^{SP}$. The notation used means that the setpoint for the controller for z_2, C_1 is P^{SP}. The controller setpoint for z_2 would then be used as a basis for the setpoints in the other controllers. For the next MV, z_3, the setpoint should be given as

a difference from the first setpoint, where the difference needs to be large enough to guarantee that only one controller is active at any given time. This denotes as $P_{c,2}^{SP} = P_{c,1}^{SP} - \Delta P^{SP}$. Thus, the last controller setpoint difference would therefore be $P_{c,3}^{SP} = P_{c,2}^{SP} - \Delta P^{SP}$, yielding it different both from the setpoint for C_1 and C_2.[15]

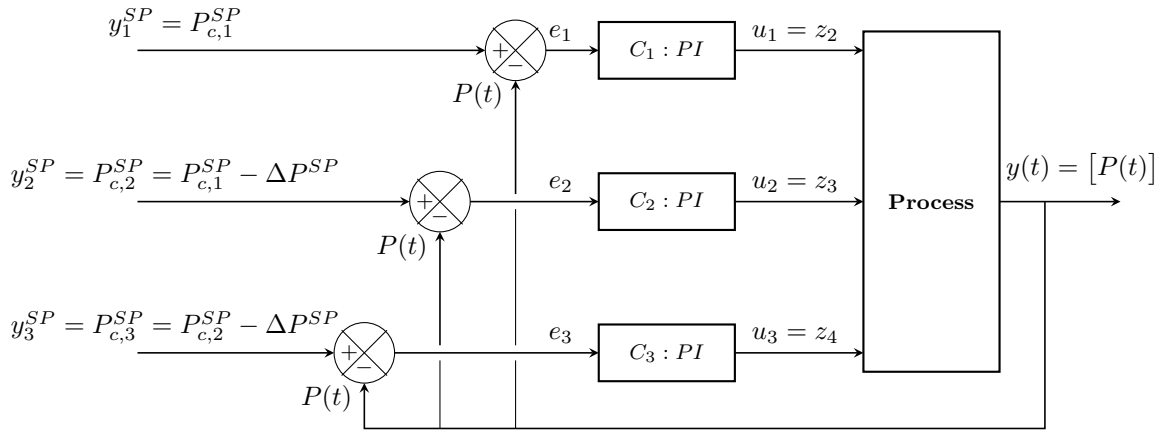

Figure 4.7: Block diagram showing controller logic for parallel control using different setpoint control on consumer side.

The controller logic needed in this project for different setpoints using consumers to control P is shown in 4.7. Since we only have one MV and thus controller active at a time, PI-control should be implemented in all controllers. This is to ensure steady state offset elimination at all operating conditions as only one controller will be active at a time.[15]

Independent tuning through the use of different setpoints could both be an advantage and disadvantage. If the relationship between all inputs in the process and the cost function is linear it would be optimal using a constant setpoint deviation between the controllers. However, if the relationship and the cost function is not linear, the independent tuning could turn out to be a disadvantageous controller structure.[9]

4.2.4 Split Range Control on Consumer Side

Split range control is another method of MV-MV switching for extending the control range for a single CV. The main contrast from parallel control is that split range control uses a common controller for multiple inputs. The controller receiving the measured process error e produces an internal signal in deviation variables v, which is the input for a split range block. The split range block then calculates the physical values for the inputs u. The concept of using internal signals could be used to keep q_2 at setpoint while saturating each consumer valve. The concept of internal signals in a split range block for this project is illustrated in Figure 4.8. In the illustrative figure the internal signal v is scaled for the purpose of illustration, but since it is not a physical value, it does not need to go from $0 - 100\%$. It is rather a design parameter, used as a degree of freedom in order to counteract the differences in the effect for the various outputs.[10]

Figure 4.8 shows that the internal signal is split into areas of active MVs. That is, in the first area, z_2 is the active MV, then z_3, then z_2. Only one MV is used at a time until saturation for the active MV is reached. Because of this z_2 would be used until saturation from fully opening before z_3 would start closing. When z_3 would become also become fully saturated, z_4 would start to saturate, essentially

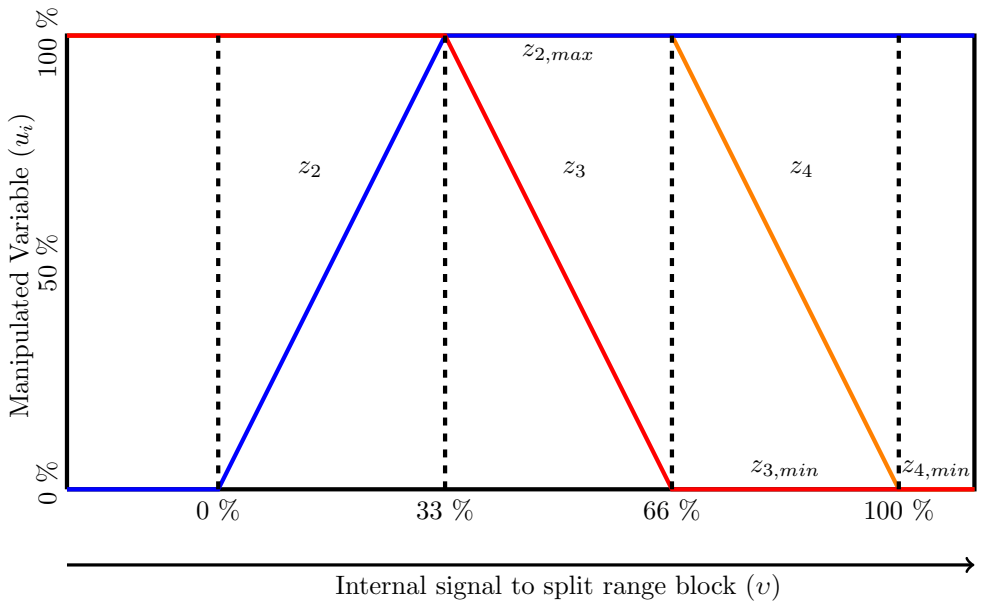

Figure 4.8: Split range block for extended control of q_2.

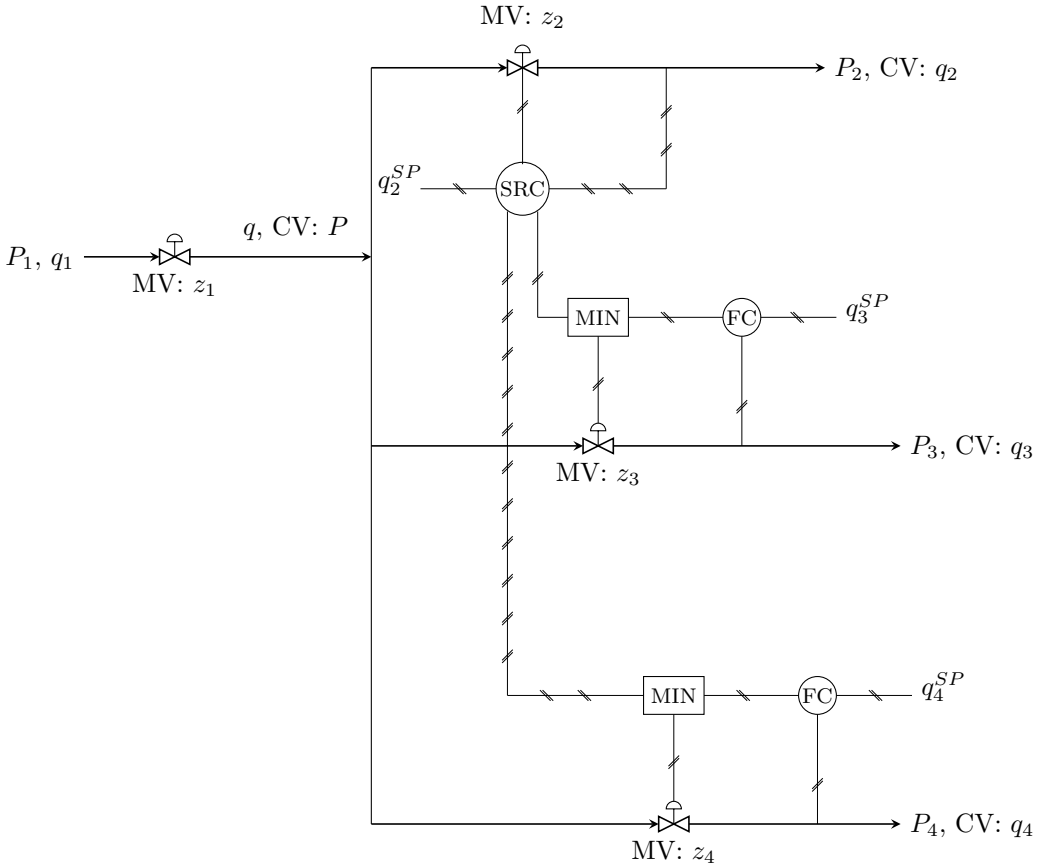

Figure 4.9: Flowsheet showing controller logic for split range control control on consumer side.

the same thing that would happen when using parallel control with different setpoints.

In order to actually utilize z_3 and z_4 when z_2 is not saturated, it would be required to also to use minimum selectors and separate flow controllers in addition to split range. The min selectors, as seen

in Figure 4.9, would ensure keeping the MVs not currently used by the split range block active as well. The block diagram and implementation of the split range control with added selectors would be very bulky and add unnecessary complexity. Properly implemented parallel control would for example implement the same idea in a much simpler fashion. Because of the unnecessary degree of complexity this controller structure poses, it is considered unfit for the extent of this project.

4.2.5 Valve Position Control

As an alternative to parallel control with different setpoints and split range control, valve position control is another decentralized control structure where the idea is to use only one MV at a time to control a single CV in order to extend the steady state operating range. In some ways, valve position control will in this case remind of split range control operation. The main difference is that while split range implements any number of MVs into one controller, valve position control only allows control between two MVs outputting one CV at the same time. This means the system could possibly require $n-1$ valve position controllers where n is the number of MVs, as opposed to one for split range. [9]

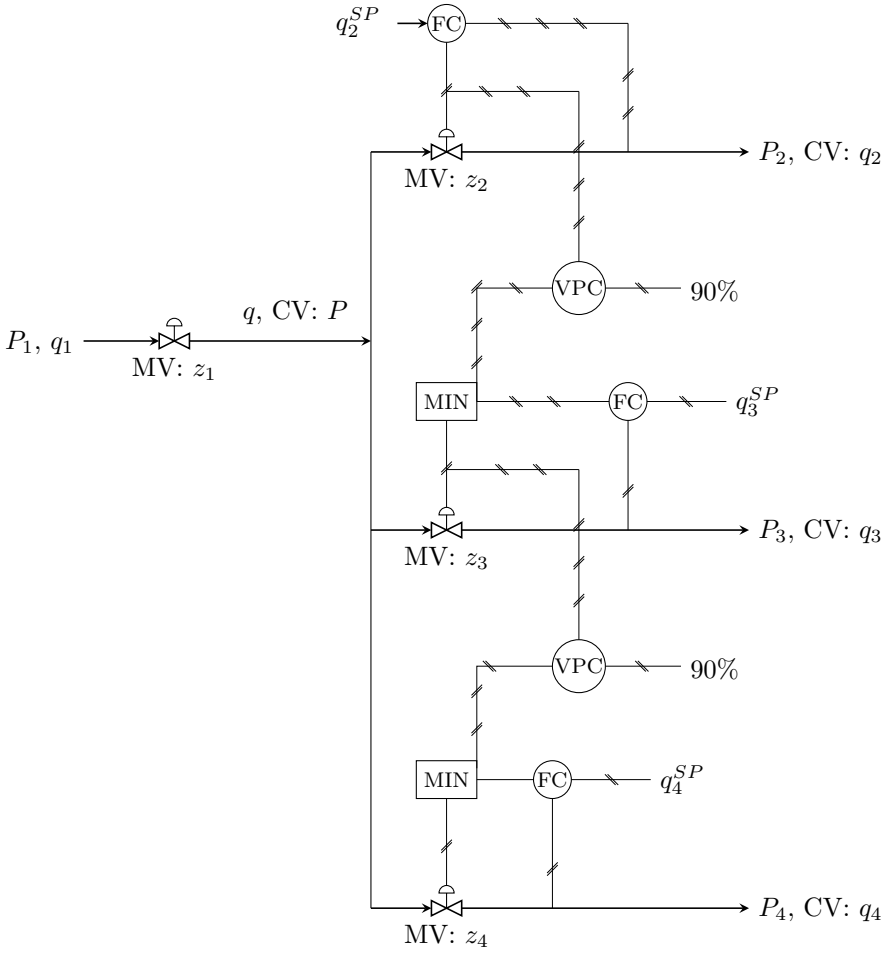

Figure 4.10: Flowsheet showing controller logic for valve position control control on consumer side.

The idea for valve position control between z_2 and z_3 controlling q_2 would be first using z_2 until saturation, z_3 would act to keep q_2 at setpoint, $q_2^{SP} = q_2^{lim} + \Delta q_2$, extending the operating range for q_2. By also adding a valve position controller between z_3 and z_4, the operating range for q_2 would also extend to the saturation of z_4. This means that for normal operation without saturation of z_2, we

would face the same problem as for split range, requiring selectors in a equal fashion as for split range control to efficiently control z_3 and z_4. Therefore, only the controller flowsheet for valve position is shown in Figure 4.10.

An issue occurring in valve position control is the fact that a back-off from the limit q_2^{lim} would be required in order to ensure that z_2 has a range to control q_2. This means that $\Delta q_i \neq 0$, meaning the full steady state range of z_2 cannot be utilized using valve position control. This is illustrated in Figure 4.10, where the 90 % max input limit for the valve position controller illustrates the need for required back-off. [9]

Also, the logic for valve position control would be equally or more complex to implement as compared to split range. Therefore also this controller structure were deemed unfit for implementation in terms of extent and goals of this project.

4.2.6 Droop Control

A problem that looks more or less the same as the gas distribution network in terms of the grid principle is electrical electrical alternating currents (AC) for electrical power generators. The difference is that the consumer defines as the frequency consumption, opposed to flow/pressure consumption for gas networks. The system is usually recognized by a power generator working as a supplier with various supply to satisfy the demand in the grid. A solution commonly found in electrical networks is droop control. The idea for droop control in electrical networks is to proportionally allocate how much each power supplier unit should increase production depending on disturbances from the frequency consumer side side. [16] Turning this around, the idea would be applicable for gas networks, by instead proportionally allocating the consumer demand for disturbances in the supplier feed.

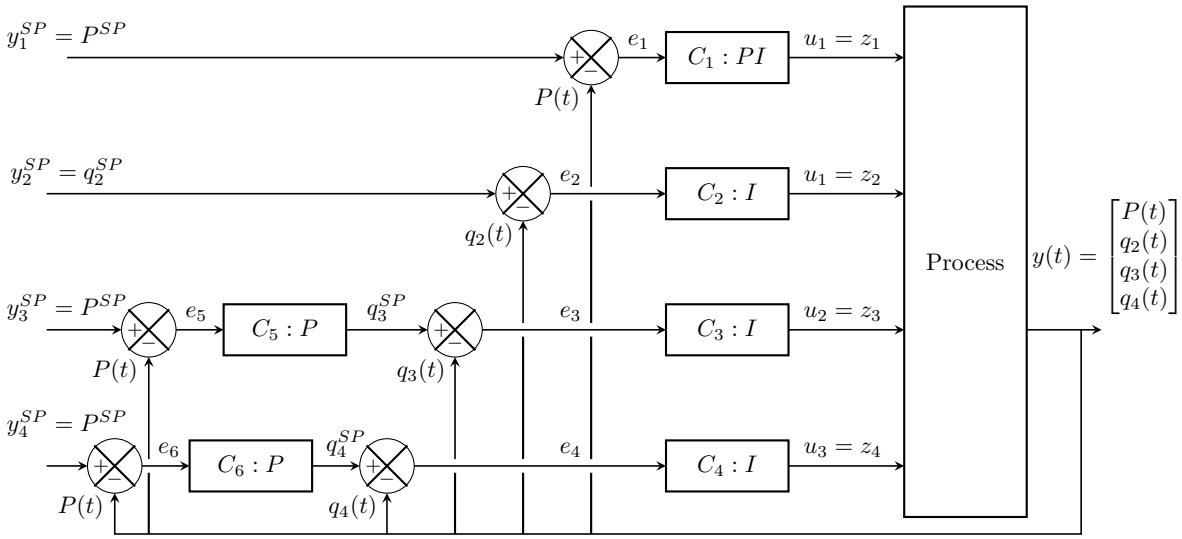

Figure 4.11: Block diagram showing controller logic for droop control control on consumer side.

Since droop is a proven method for efficiency, the odds are good in terms of this control structure possibly being able to control both q_2, as well as maintaining main pressure in the system P. Droop control would make it possible to allocate consumer demand such that for a drop in P_1, z_3 and z_4 should proportionally reduce their demand. In other words, a dynamic setpoint for the MVs that does not have to follow setpoint or are less economic important should be implemented. An easy way to do this, is to use the default supplier and consumer control structure, with the setpoints for q_3 and q_4

being dynamic, set by P-controllers obtaining their setpoint from measurement of P. This will allow dynamic setpoints for flows that are not prioritized, as well as maintaining pressure P, and flow q_2. The block diagram for droop control using consumers is shown in Figure 4.11.[16]

Because droop control shows indications of being one of the simplest and yet most efficient control structures in terms of implementation, the simulation results for this control structure is expected to be more appreciated than for any of the previously presented control structures. The next section presents the results from implementation of the most promising control structures.

Chapter 5

Results

Having obtained a model and system solver, actual implementation of the proposed controller structures could start. This section presents the implementation for the proposed controller structures of this project, starting by investigating the system open loop responses, then investigating the use of consumer control using supplier and consumers, then parallel control with equal and different setpoints, then droop control.

5.1 Open-Loop Step Responses

Open-loop step responses should be analysed to determine the type of process the system dynamics represent, and thus which controller type should be used. The open-loop step responses were obtained from doing steps on the system MVs, that is possibly z_1, z_2, z_3 or z_4. In order to tune the controller, the response for the related CV should were observed. That is, if z_2 was used as a MV related to P as its corresponding CV, the response for P would be analysed.

Figure 5.1 shows a +10% step on z_2. As expected, the increased valve opening will cause an increase in the related flow q_2. This causes a drop in P because the pressure drop will be smaller when the valve opening increases, and the model keeps consumer pressures constant. It is observed that the corresponding response in P is a first order process with no time delay, while the response for q_2 is an integrating process with no time delay, a static process. This means, that if P is to be used as a CV with z_2 as an MV, PI- or P-control should be used. If q_2 is to be used as a CV instead, I-control should be used. The lack of time delay is because of the simplicity in the model itself. Time delay could be added through the use of Time Delay blocks in Simulink, but this is not a necessity as it would only change the tuning dynamics in the process, and not the responses themselves, other than adding a delay.

The responses for q_3 and q_4 were correspondingly equal in shape for steps on z_3 and z_4 respectively. Because the valve and flow dimension for z_3 and z_4 differs, so did also the magnitudes in the responses compared to the open-loop step on z_2, but not anything else. Thus, to control outflow from any of the consumer valves, I-control should be implemented. For controlling the main pressure P as a CV, P- or PI-control should always be be used.

Doing an open-loop step on z_1, shown in Figure 5.2 confirms the fact that a P- or PI controller should

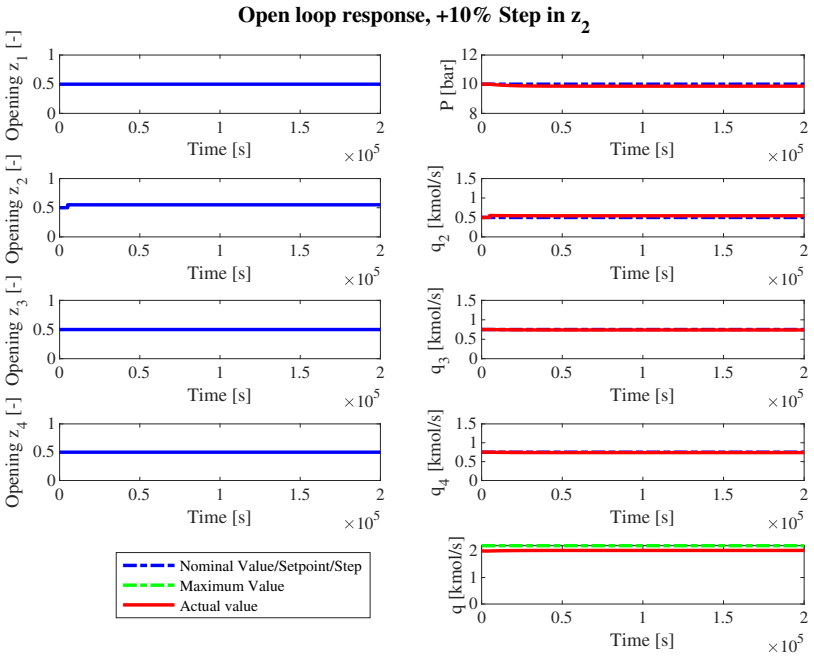

Figure 5.1: Open loop system response for a +10% step on z_2

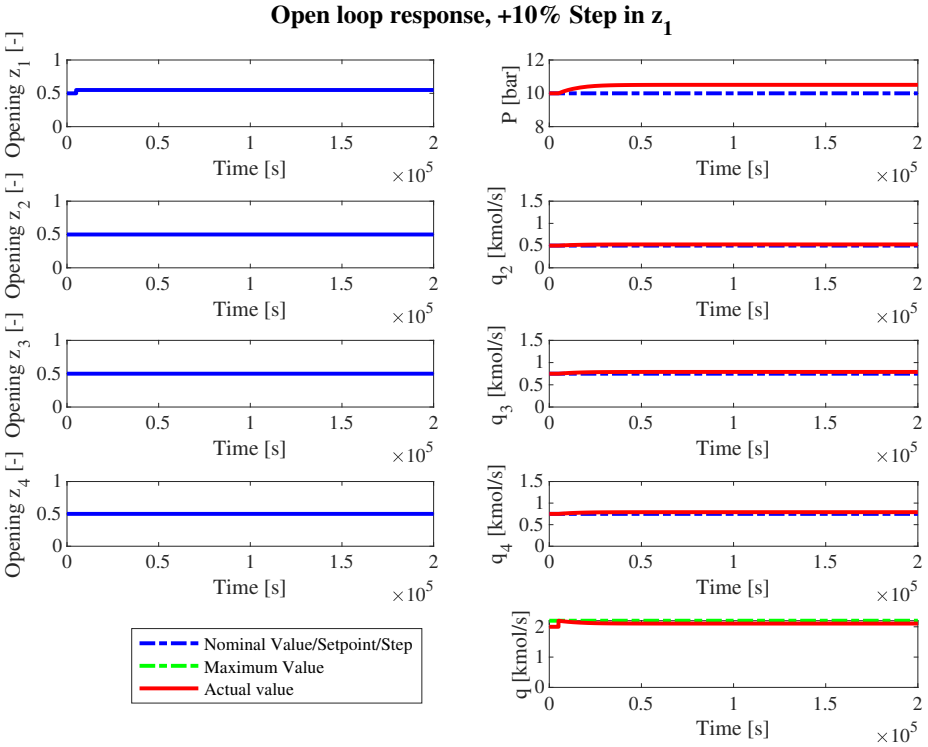

Figure 5.2: Open-loop system response for a +10% step on z_1

be used for controlling pressure. In this case, the outlet flows are not static responses, but since they will not be directly manipulated by z_1 in this project, the response does not matter for those flows.

With the open-loop step responses confirming what type of controllers should be used for the different controller cases, advanced controller structures could be implemented. The knowledge about open-loop behaviour is important to obtain optimal control when implementing controller structures.

5.2 Default Control

5.2.1 Presentation of Results

The tunings and disturbance responses from implementation of the logic for default control found in Figure 4.5 are presented in Table 5.1 and Figures 5.3 through 5.4. In the response Figures the controller structure was exposed to a +100% increase and a −30% drop in P_1. All the other controller responses were exposed to the same increase and decrease in the inlet pressure P_1.

Table 5.1: SIMC tuning parameters for inputs using default control using supplier and consumers.

Input	K_c	K_i	θ	τ_1	τ_c	τ_i	k
$u_1 = z_1$	0.0514	5.1406e-06	0	10000	1.9072e+04	10000	10.2
$u_2 = z_2$	0	5.2434e-05	0	0	1.9072e+04	0	-1.16
$u_3 = z_3$	0	3.4956e-05	0	0	1.9072e+04	0	-4.02
$u_4 = z_4$	0	3.4956e-05	0	0	1.9072e+04	0	-2.44

The tuning from Table 5.1 were obtained from first doing an open-loop step changes on z_1, then tuning a static PI-controller according to the response in P. Then, for open-loop steps in z_2, z_3 and z_4, the responses from the respective outflows q_2, q_3 and q_4 were used to tune the remaining controllers. As expected, the responses were those of a static process, meeting the expectation that integral controllers should be used for the consumers in this controller structure.

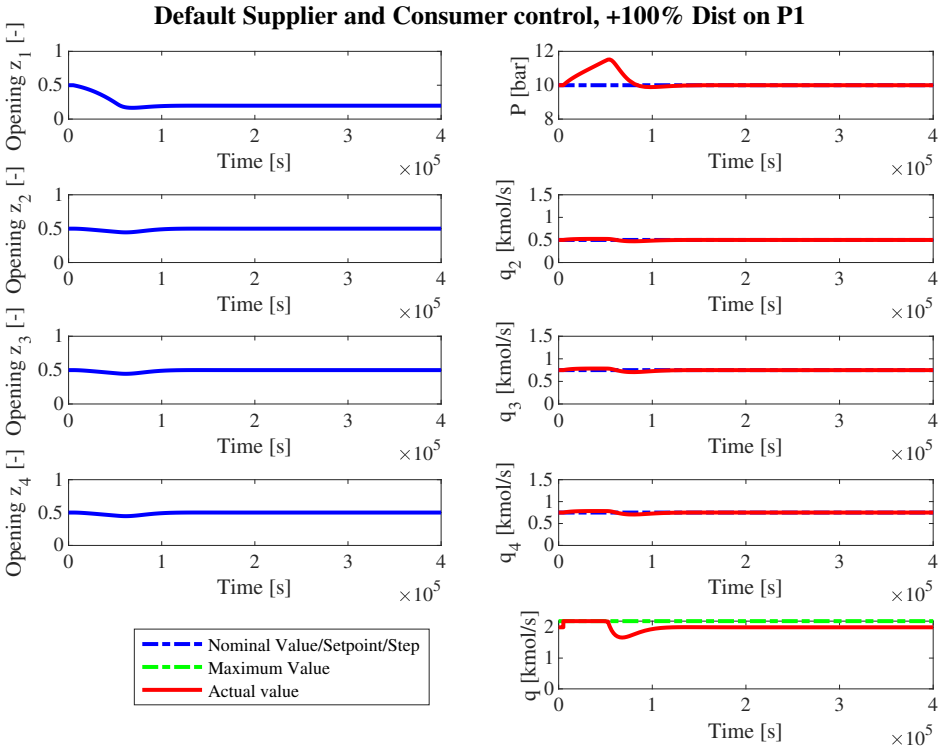

Figure 5.3: Default supplier and consumer control response for a +100% disturbance from P_1

5.2.2 Results Analysis

From the disturbance response Figures 5.3 through 5.4 it is observed by looking at control for q_2, the controller does a good job in returning q_2 to setpoint value. However, there are two main issues with the structure.

The first, main issue would be the fact that q_2 is in no way prioritized above the other flows. This becomes most apparent in Figure 5.3, where it is becomes clear that the responses for all flows are equal. The only reason the magnitude for the response in q_2 is smaller is because it is a smaller flow, and thus also the valve dimensions, thus the disturbance results in what could look like a smaller magnitude, but in reality is not. Thus, a control structure where prioritization of the outflows are possible is desired.

For the second issue, it is observed that for a negative pressure drop, a large steady state offset is maintained for P. The offset is larger than 10% which could clearly have a great effect on the overall steady state conditions in the system. The steady state offset is caused by the saturation in the valve z_1, and would only the offset would only increase if the drop in P_1 was larger.

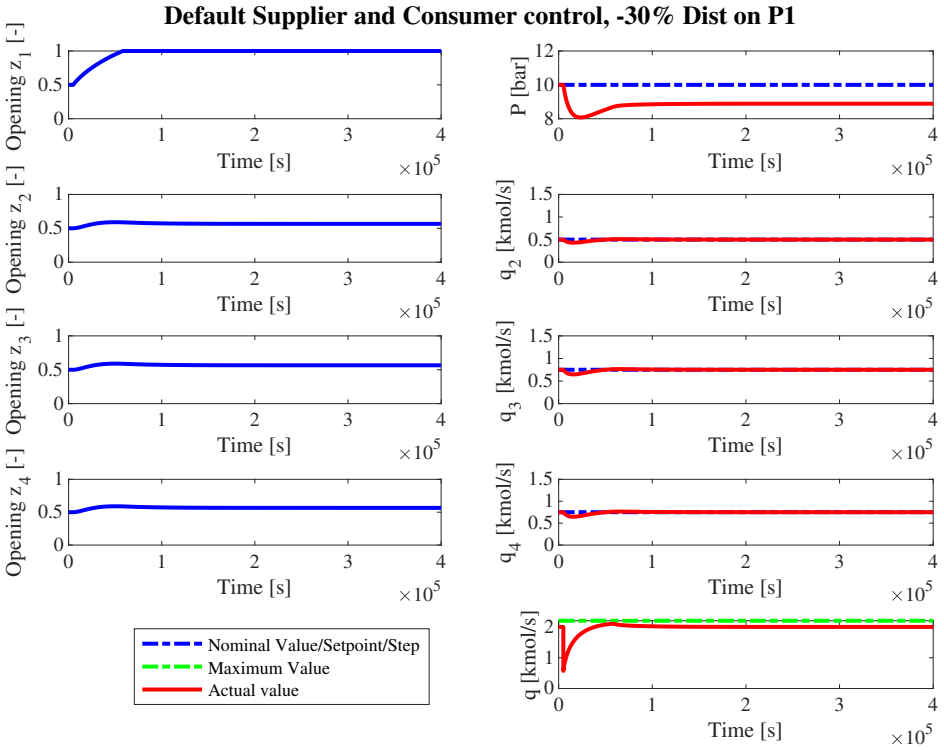

Figure 5.4: Default supplier and consumer control response for a -30% disturbance from P_1

The drop in P_1 for Figure 5.4 also provides a drop in q. This is a direct response to the drop in P_1 which essentially is a static process. The reason for this drop and the smaller drops in the outlet flows, it the sudden drop in P followed by the delay in fully opening z_1, and is not really a problem in the process, it is a natural consequence followed by controller delay before maintaining nominal value as controller action progresses.

The conclusion for this control structure is therefore that is usable for a disturbance in P_1, but not very robust. It would be desired rather to give up more flow from q_3 and q_4, maintaining P as well as prioritizing q_2.

5.3 Parallel Control - Equal Setpoints

5.3.1 Presentation of Results

The tunings and disturbance responses from implementation of the logic for parallel control using equal setpoints found in Figure 4.6 are presented in Table 5.2 and Figures 5.5 through 5.6. In the response Figures the controller structure was exposed to a +100% increase and a −30% drop in P_1.

Table 5.2: SIMC tuning parameters for inputs using parallel control on consumers.

Input	K_c	K_i	θ	τ_1	τ_c	τ_i	k
$u_1 = z_2$	-0.226	-4.5201e-05	0	5000	1.9072e+04	5000	-1.16
$u_2 = z_3$	-0.163	0	0	12500	1.9072e+04	0	-4.02
$u_3 = z_4$	-0.1612	0	0	7500	1.9072e+04	0	-2.44

The tunings in Table 5.2 were found by first obtaining open-loop responses were from stepping on the first MV, z_2, looking at the response for P, then tuning one controller, before stepping on the next consumer MV, z_2, studying the response for P to tune the next controller. The responses were as expected first order, meaning proportional control with integral action should be implemented. The integral action were decided to implement in the controller for z_2. Because the controller was quite slow, it was tuned the other way around, by tuning the pure P-controllers first. This yielded a faster controller, and it was decided to use the more aggressive tunings instead.

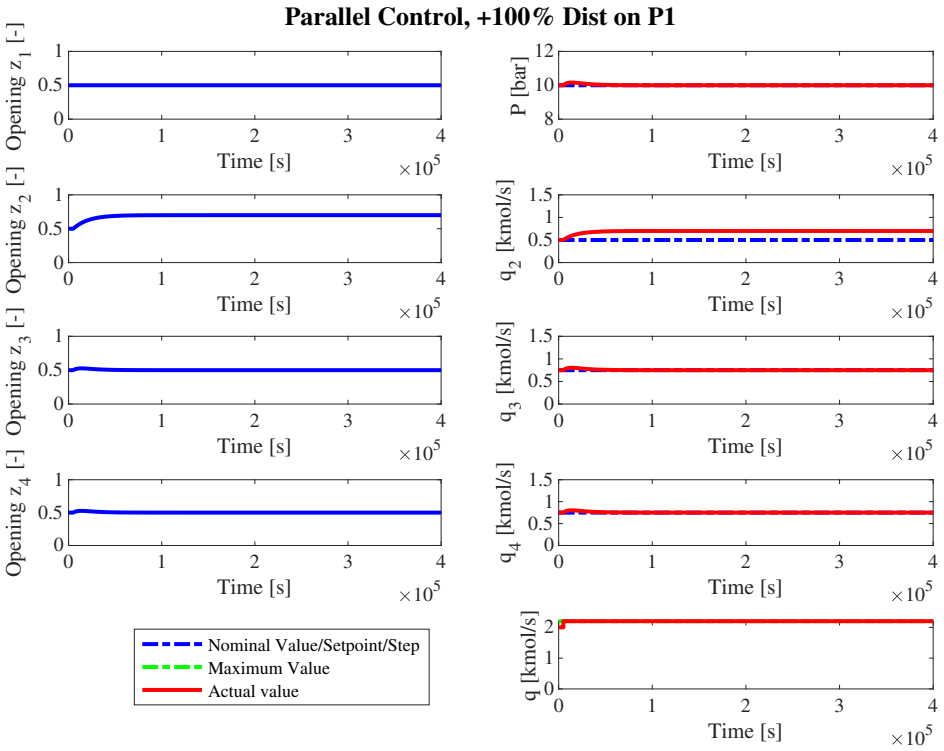

Figure 5.5: Parallel control using equal setpoints for a +100% disturbance from P_1

5.3.2 Results Analysis

From a controller perspective, the parallel control using equal setpoints does what it intends, which is saturating the MVs equally to maintain the CV setpoint. In Figure 5.5, the controller for z_2 deviates from it setpoint value to maintain P. Remember that the idea is to first saturate one controller at a time to maintain the CV-value, P. For Figure 5.6, z_2 saturates fully in order to maintain P, however z_3, which should do so, is not. This implies that there are some issues with this controller structure.

First of all, z_2 fully saturates for P_1 whereas the desired effect were z_4 saturating, then z_3 then z_2 lastly. But even though this is a large issue, the main issue lies in the fact that all consumers are given up more or less equally at the same time. The reason q_2 saturates fully is because of the I-action in the controller, whereas without this, the setpoint deviation would probably be about the same as for q_3 and q_4. But this is a paradox, because removing the I-action would remove the corrective action taken to maintain P at setpoint. For the positive disturbance on P_1 in Figure 5.5, we see that the setpoint for P is actually maintained, whereas since z_2 is saturated for the negative disturbance in 5.6 the PI-controller for z_2 is saturated, making the system essentially loose control.

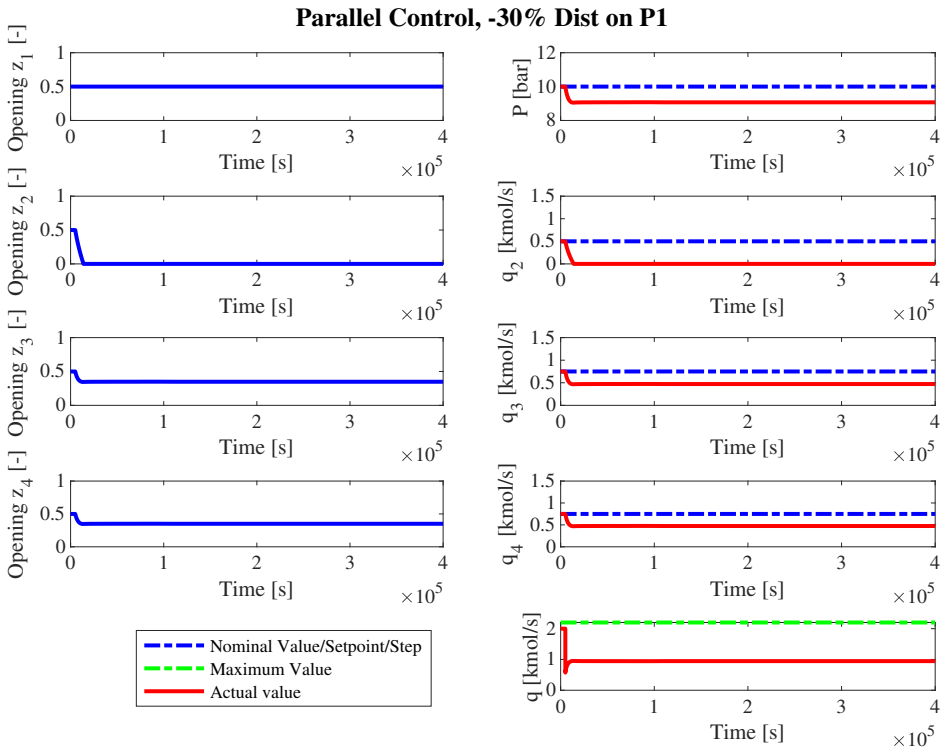

Figure 5.6: Parallel control using equal setpoints response for a -30% disturbance from P_1

So, the feasible alternative for this controller structure would possibly be if each consumer flow were equally prioritized and pressure was the main CV. We would also have to assume that the disturbance would not be large enough to saturate the consumer flows. Thus, putting PI-control on a large, buffer flow, could help the parallel control with equal setpoints extend its range of operation. The controller structure is however not recommended for controlling a single outflow.

5.4 Parallel Control - Different Setpoints

5.4.1 Presentation of Results

The tunings and disturbance responses from implementation of the logic for parallel control using different setpoints found in Figure 4.7 are presented in Table 5.3 and Figures 5.7 through 5.8. In the response Figures the controller structure was exposed to a +100% increase and a −30% drop in P_1.

Table 5.3: SIMC tuning parameters for inputs using different setpoints control using supplier and consumers.

Input	K_c	K_i	θ	τ_1	τ_c	τ_i	k	P^{SP}
$u_1 = z_2$	-0.2330	-1.9420e-05	0	5000	1.9072e+04	5000	-2.7	10
$u_2 = z_3$	-0.1116	-2.7890e-05	0	12500	1.9072e+04	12500	-1.88	9.5
$u_3 = z_4$	-0.1192	-3.9722e-05	0	7500	1.9072e+04	7500	-1.32	9

The tunings in Table 5.3 were found from doing open-loop steps on z_2, z_3 and z_4, looking at the respective responses for P in each open-loop step simulation. The main difference from using equal setpoints were that now the setpoints for each consumer valve were different. Since no economical objective function were defined for this project, defining how large the difference in the pressure setpoints for z_2, z_3 and z_4 proved difficult. The setpoint difference between each controller were set to 0.5 bar, and are presented in the tuning Table. Because no optimal setpoint calculation were done, it is unknown if that number would be to small or large for optimal implementation of parallel control using different setpoints.

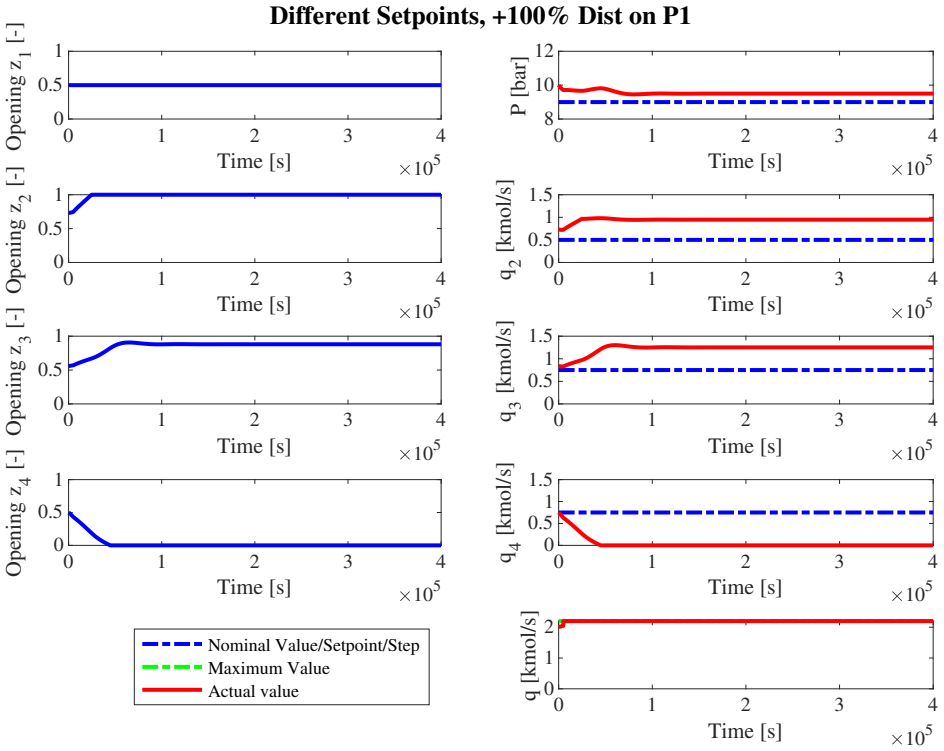

Figure 5.7: Parallel control with different setpoints response for a +100% disturbance from P_1

5.4.2 Results Analysis

Knowing that the starting point for this method was not optimal, the method performed somewhat as expected. This time, since all controllers can be tuned independently, they all consist of proportional and integral action. This implies some problems. For the case with a negative pressure drop in P_1 in Figure 5.7, the controller I-action interacts, as warned about earlier, forcing two of the controllers to fully close and the last one to fully open. This causes q_2 to stay above setpoint while q_3 and q_4 is fully closed, which makes sense. However it is not how optimal operation should be implemented for the objectives in this project. This controller structure would be better suited for a case where it is favorable to only have one MV active at a time. If this control structure were to work for the objectives in this project, some way of allocating the flow from q_2 to either q_3 or q_4, should be implemented. This is what ideally would happen with this control structure, but because of integral interaction, the controllers work against each other.

The same thing actually happens in Figure 5.7, even though it may not be as clear here. We must remind ourselves that z_2, z_3 and z_4 have different setpoints, so what happens in Figure 5.7, is that z_3 actually reached setpoint. Therefore, an offset is caused in P also. This shows an obvious disadvantage when using parallel control with different setpoints.

The reason why the integral action is working against us is because when more than one integral action is active in a loop, there is no unique steady state solution for the system. In controllers with different setpoints it is desired to give the difference between the setpoints such that only one MV is actively used at the same time, while in this case, interactions make all the MVs being used at the same time.

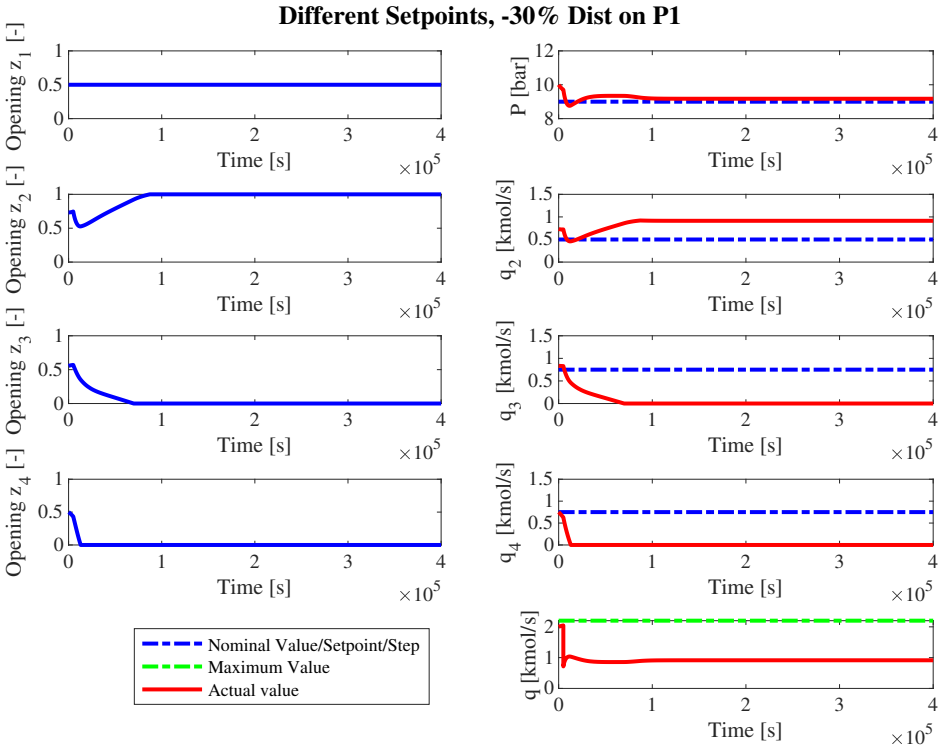

Figure 5.8: Parallel control with different setpoints response for a -30% disturbance from P_1

For this system, doing extensive work on what setpoints should be used could have avoided the setpoint offset for a positive disturbance from P_1 in Figure 4.7. However, it would not eliminate the interaction

in the integrating parts of the controllers, and is therefore not recommended as a structure used for the problem in this project.

5.5 Droop Control

5.5.1 Presentation of Results

The tunings and disturbance responses from implementation of the logic for droop control found in Figure 4.7 are presented in Table 5.3 and Figures 5.7 through 5.8. In the response Figures the controller structure was exposed to a +100% increase and a −30% drop in P_1.

Table 5.4: SIMC PI tuning parameters for droop control using supplier and consumers as an extension to default control of supplier and consumers.

Input	K_c	K_i	θ	τ_1	τ_c	τ_i	k
$u_4 = q_3^{SP}$	-0.0518	3.4956e-05	0	0	1.9072e+04	7500	-2.44
$u_5 = q_4^{SP}$	-0.0782	3.4956e-05	0	0	1.9072e+04	7500	-2.44

The droop control tunings in Table 5.4 were found using the already tuned controller structure for u_1, u_2, u_3 and u_4 using default control tunings for found in Table 5.1. Then, steps on the setpoints for q_3 and q_4 were performed, and controller tunings for u_5 and u_6 were based on the first order pressure responses this caused. This required proportional controller action for the droop control, as expected.

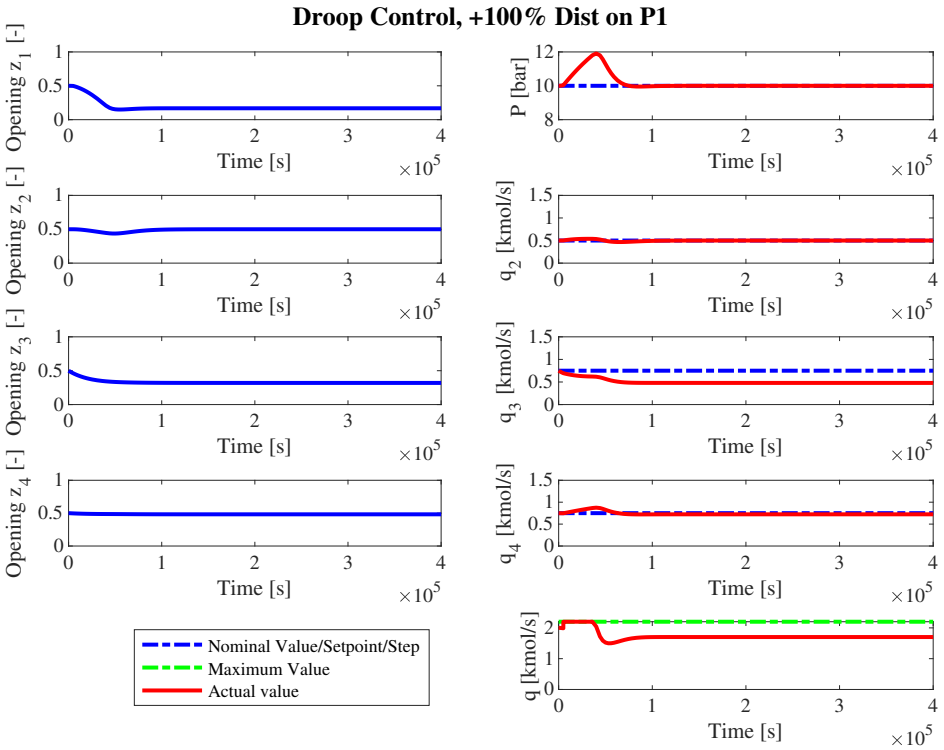

Figure 5.9: Droop Control response for a +100% disturbance from P_1

This controller structure reminds of a cascade controller-hierarchy. Cascade implies a controller hierarchy with a fast slave controller and a slow master controller.[12] With the proportional controllers

being the master controllers and the integral controllers being the slaves, the tunings were modified such that the integral controllers were faster than the proportional ones. Also, to illustrate the effect of individual prioritization of the lesser prioritized consumers, the proportional gain for the P-controllers for the controllers for q_3^{SP} and q_4^{SP} were set slightly different.

5.5.2 Results Analysis

As expected, the implementation of droop control would require implementing proportional controllers for pressure control on top of an existing control hierarchy. The obvious choice was therefore to add the proportional action on top of the existing integral controllers for default control. This would give also proportional control in the loops with droop control, possibly yielding better control. As observed, the controller structure works exactly as expected for maintaining q_2 and proportionally allocating a lower demand from the other consumers, prioritizing q_4 above q_3 as we set a higher proportional gain in absolute value for q_3. For a negative disturbance in Figure 5.10, a small setpoint deviation is observed for P. This is caused by the fact that the supplier valve saturates. This offset could probably have been avoided by allocating an even higher proportional gain for the consumers that could be given up.

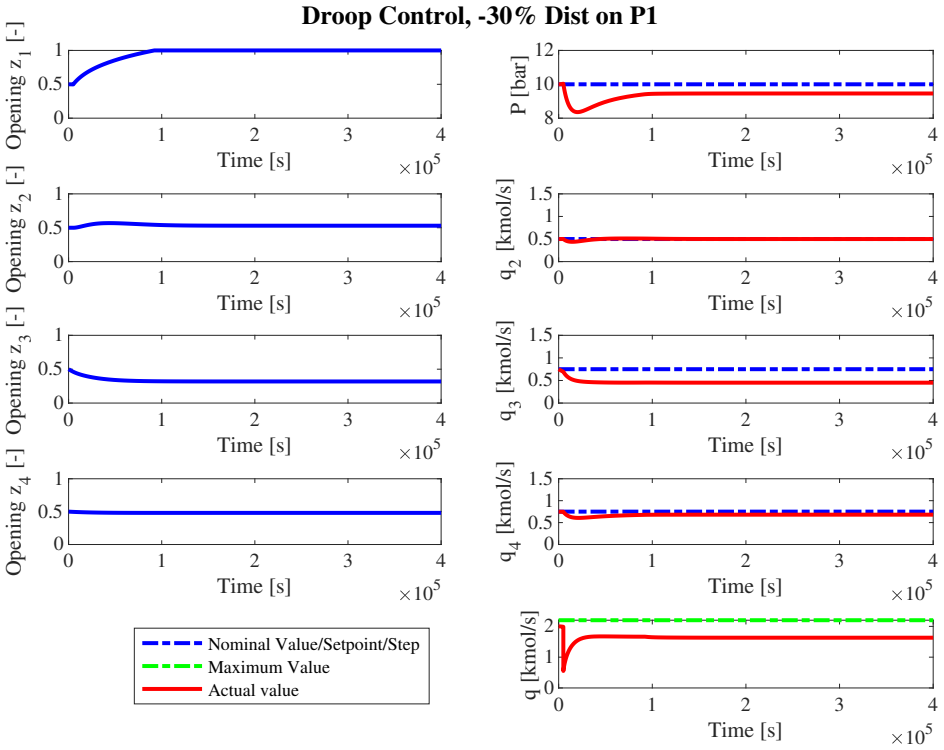

Figure 5.10: Droop Control response for a -30% disturbance from P_1

A droop loop were not added to the controller for q_2 as this flow should be prioritized above the others, and we do not want it do deviate from setpoint at all, because of the hard setpoint CV-constraint. If the objective was different, it could be applicable to implement a proportional controller for this controller structure as well. In this simulation, a controller for controlling z_1 is also included, as it has been shown that when the main pressure P is controlled using z_1, we are able to maintain nominal outflows. This means than in a real situation, P control should always be implemented. In this simulation it is not really necessary, because z_1 will be forced to saturation anyways.

Since z_1 will saturate, a simpler control structure using only the consumer side could also be feasible. The only difference that would make to the controller structure is removing u_1 and associated controllers and setpoints. In practice however, we would probably want to have P control using the supplier as well if possible, as keeping P constant maintains predictability in the plant. By keeping system pressure at desired setpoints we ensure more stable operation throughout the process, after the consumer valves, as a loss of pressure is less favorable in terms of possible extra compression costs, thermodynamic instabilities etc.

As a controller structure, droop control turns out to be the most feasible option for the conditions and model in this project. It must however be noted that some modifications were done in order to make the system optimal for droop control. These are discussed in the next section.

5.6 Performance Comparison

A comparison of all the controllers responses for critical disturbances in P_1 is shown in Figure 5.11 and 5.12. It becomes clear here that each controller unit has its strengths and weaknesses, but most of all that droop control is way more suited than any of the other structures. That is, for the objective in this project, which is maintaining q_2^{SP} and P^{SP} if possible. Droop control gives tight control for both the consumer flow as well as the main pressure, which could yield other advantages, such as more predictable operating conditions. While decreased flow would lead to a slower system, a lower or higher pressure could lead to completely different system dynamics, following the thermodynamic laws. For gas supply networks, a minimum pressure is often required for optimal transport and processing conditions.

Even though all controller structures were forced to saturation for z_1 with a negative step P_1, this would be mostly to get comparable results for the different control structures with and without supplier control. In reality, supplier control would most likely be desired, as for most of the structures with supplier control, q_2 can be controlled fine as long as z_1 do not saturate. If normal operation with no saturation of z_1 could be assured already from system sizing and planning, then a good alternative to droop would be simple feedback control for z_1. Again, this is not realistic as in a real process, both demand and supply are likely to be dynamic, so to ensure as extended and wide operating conditions as possible and increase robustness of the system, droop is still preferred.

It should also be noted that some modifications were done in order to make the system optimal for droop control. In general, for droop control to work as well as possible, the prioritized flow or flows should be small compared to the system in total. Therefore the nominal flow for q_2 is modelled as smaller than the other consumers, including the valve coefficients. However, this only implies that the other consumers have more flow to give up to maintain q_2, so in reality, this tweak does not affect the performance of the controller structures, still making droop control the definite winner in this project.

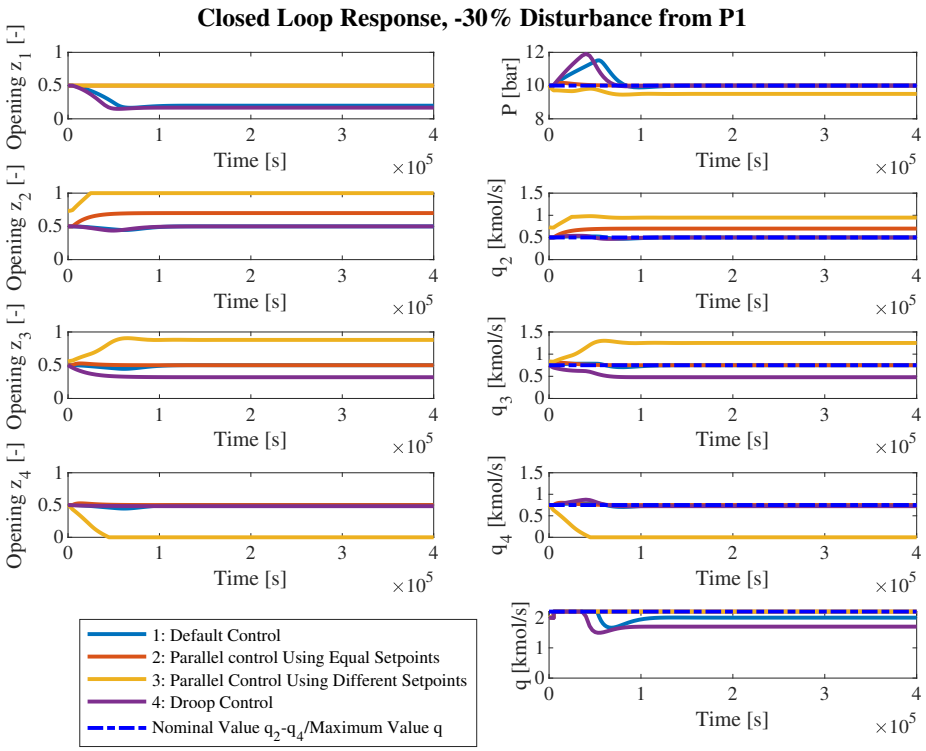

Figure 5.11: Response comparison for all controller structures for a +100% disturbance from P_1

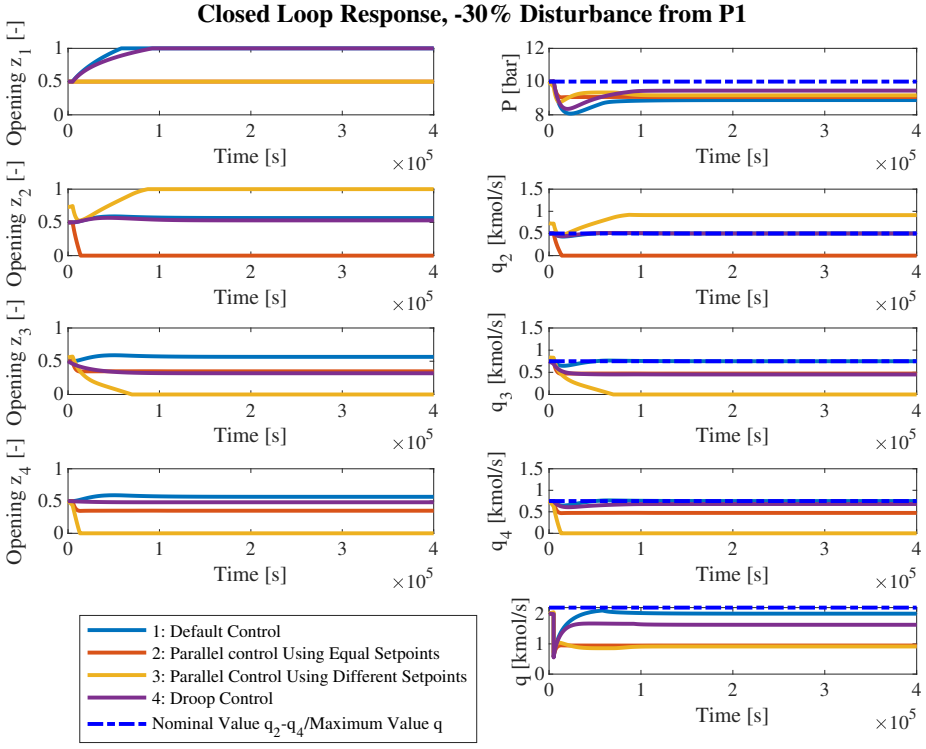

Figure 5.12: Response comparison for all controller structures for a -30% disturbance from P_1

Chapter 6

Discussion

Some subjects require further discussion than what is presented in the results section. They are presented in this section.

6.1 Model simulations

This system is a small system, consisting only of three consumers. Usually gas network system consist of a number of hundreds or thousands of consumers. These large systems are usually stiff. This means the dynamics are expected to be fast, but small. When the system is stiff, the system is expected to be sharp in simulations and could give unfeasible solutions using the MATLAB standard and this project non-stiff ODE solver ode45. This type of solver will catch up on the small but sharp changes, and typically decrease solver algorithm step size to unreasonably small levels. This could happen even in smooth regions of the solution curve. Thus, the solver could end up doing millions of evaluations in a small, even smooth solution interval, leading to the solver using very long time to simulate results, or possibly even failing integration. Therefore, if infeasible solutions turns out to be a problem in large systems, a more suited, stiff ODE solver like ode15s should be considered. These solvers typically have much larger steps, but the trade-off is that they do much more work per step. For this system that was not really a problem whatsoever, as the problem is relatively small. Choosing a suited solver for the system model is anyway something to keep in mind when modelling gas networks.

6.2 Simplifications

The arguments presented for the ideal gas law are valid only for high temperatures and relatively low pressures. Thus, they should be valid for this system. If it turns out that the ideal gas law is not valid, we should also consider density in the valve equation, and we would have to reformulate our entire problem structure, complicating the model to extents.

The assumption that the system models a pure gas system is not so realistic. For positive disturbances in P_1, the main pressure P often exceeded 10 bar, increasing the dew point temperature. Therefore, the simulation temperature should have been even higher than proposed to ensure gas-phase simulations only.

Since the system modelled was small and fast to solve in MATLAB, we could have spent time on obtaining a more realistic system model, not simplifying the valve pressure drop relation using ideal gas law and average densities. In the end, the time extent of this project is what limited the level of complexity in the model.

One simplification that should have been implemented is valve dynamics. This is usually included in a model and would yield different responses regarding tuning of the plant. However, the simulations would likely present the same results with droop control being the best control structure with added valve dynamics, so the conclusion of this project is still valid.

Another simplification were delays not included in the model. A real process would almost always include some form of time delay in the process. However, this would not change the responses and tunings other than the fact that tuning would have to be redone.

The fact that the time delay θ was zero in the model, did that we had to choose τ_c by other means also. Therefore τ_c was chosen to be equal to τ_r/n, where τ_r is equal to the residence time. This turned out to be a good choice for τ_c making the closed loop system responses stable and not oscillatory. Using more aggressive values for τ_c were also simulated in the early simulation stages, yielding oscillatory behaviour of the system. Thus, $\tau_c = \tau_r/n$ turned out to be a good choice, even though adding time delays to the model and tuning τ_c with the help of process delay would likely also be a feasible option.

6.3 Alternative Controller Structures

One alternative for the parallel control using equal setpoints would be to use different τ_C for the different controllers. This would make controllers slower in their response and possibly make one able to prioritize q_2 to some extent, however in reality we would just end up slowing up the inevitable, that is the proportional action from the slower control. Therefore, this would be equal to having no control on the slower controllers until they respond, which is unwanted. It is not desired to completely loose control on any of the flows in case of sudden disturbances from other places such as the consumer outlets.

Also, some of the controller structures such as split range control could be used with z_1 as an MV also be able to equally compare implementation logic. For example, for split range control also utilizing the supplier side, using P as a CV for the supplier side vale. The output from the split range consumer side would then go to added minimum selectors also inputting signal from flow controllers on q_2, q_3 and q_4. Using this structure would only make z_1 and the consumer valves deviate from setpoint in normal operation. Not until z_1 saturate split range would actually be applied, saturating the consumer side. This would still be a less adequate solution than for droop control, because in droop control the consumers all consumers shares the duty of pressure control proportionally, while in SRC only one consumer at a time would be active in controlling the flow, as previously stated for SRC. Also, the controller structure would be even more advanced than the one proposed in Figure 4.9.

Anyway, if we add pressure control to the structures where it could be utilized and is not now, the results would not really make a difference other than in added implementation complexity when z_1 saturates. All in all, droop control is the optimal control structure for the problem statement in this project

Chapter 7

Conclusion

This project analyzed and compared controller structures for default control, parallel control using equal and different setpoints and in the end droop control, for a modeled gas network, implementing the corresponding logic presented in Figure 4.5 through 4.11. The default control responses for a large disturbance caused by P_1 shown in Figure 5.4, showed that control of q_2 can be maintained using this controller structure. The main issue with this structure were that no consumer prioritization is performed.

For parallel control shown in Figures 5.6 and 5.8, the idea that controlling P as a CV would yield control to keep consumer flows at nominal values was implemented. This turned out not to be the case as interaction caused all controllers to give up flows from setpoint or fully saturate the MVs, loosing system control.

In the end, it was shown that droop control was the most viable decentralized controller structure both for maintaining prioritized consumer flow q_2 and keeping main system pressure P at setpoint, as seen from the results of large disturbances in P_1, shown in Figures 5.9 and 5.10. Droop control also has the advantage that it opens for the possibility of prioritizing all consumers differently by proportionally allocating how much each consumer should be given up, thus making a hierarchy of prioritized CVs.

Even though droop control has been proved to work well for balancing supply and demand in gas networks, it should be compared to the use of multivariable control structures for future work. Multivariable control structures include model predictive control, neural networks and more.

The main reason they should be compared is to see whether multivariable control such as model predictive control will actually will perform better than decentralized control structures. This may very well not be the case, and in that case, we have shown that centralized control not necessarily is the optimal solution, even though this is the most common implementation when considering supervisory control. If not, other considerations should be discussed, as MPC implementation is much more advanced and expensive than using decentralized control structures.

Bibliography

[1] Airikka, P. (2012). Balancing steam supply with demand saves energy. *Control Engineering*, 59:P1–P4.

[2] da Silva, L. R., Flesch, R. C., and Normey-Rico, J. E. (2018). Analysis of anti-windup techniques in pid control of processes with measurement noise∗∗this work was supported by the brazilian national council for scientific and technological development (cnpq) under grants 311024/2015-7 and 305785/2015-0. *IFAC PapersOnLine*, 51(4):948–953.

[3] Dictionaries, L. (2020). Optimization: Definition of optimization by oxford dictionary on lexico.com also meaning of optimization.

[4] Jeppu, Y. (2020). Anti-windup pid example.

[5] King, M. (2011). Process control : a practical approach.

[6] Kretzschmar, H.-J. and Wagner, W. (2019). *International Steam Tables: Properties of Water and Steam Based on the Industrial Formulation IAPWS-IF97*. Springer Berlin / Heidelberg, Berlin, Heidelberg.

[7] McMillan, G. K. (2015). Tuning and control loop performance.

[8] Parkinson, J. S. and Wynne, R. J. (1992). Systems modelling and control applied to a low-pressure gas distribution network. *Proceedings of the Institution of Mechanical Engineers, Part E: Journal of Process Mechanical Engineering*, 206(1):35–44.

[9] Reyes-Lúa, A. and Skogestad, S. (2019). Multiple-input single-output control for extending the steady-state operating range—use of controllers with different setpoints.

[10] 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.

[11] Skogestad, S. (2004). Control structure design for complete chemical plants. *Computers Chemical Engineering*, 28(1):219 – 234. Escape 12.

[12] Skogestad, S. (2005). Multivariable feedback control : analysis and design.

[13] Skogestad, S. (2008). Chemical and energy process engineering.

[14] Skogestad, S. and Grimholt, C. (2012). The simc method for smooth pid controller tuning. *Advances in Industrial Control*, pages 147–175.

[15] Smith, C. L. (2010). *Advanced process control: beyond single loop control*. Wiley, Hoboken, 1st ed. edition.

[16] Wood, A. and Wollenberg, B. (1996). Power generation operation and control — 2nd edition. *Fuel and energy abstracts*, 37(3):195–195.

[17] Xue, J.-j., Wang, Y., Li, H., Meng, X.-f., and Xiao, J.-y. (2016). Advanced fireworks algorithm and its application research in pid parameters tuning. *Mathematical Problems in Engineering*, 2016:1–9.

Appendices

Controller Flowsheets and block diagrams

In this appendix, some of the flowsheets and block diagrams for relevant project controller structures that were not used in the main report is found. The controller descriptions are found in the figure texts.

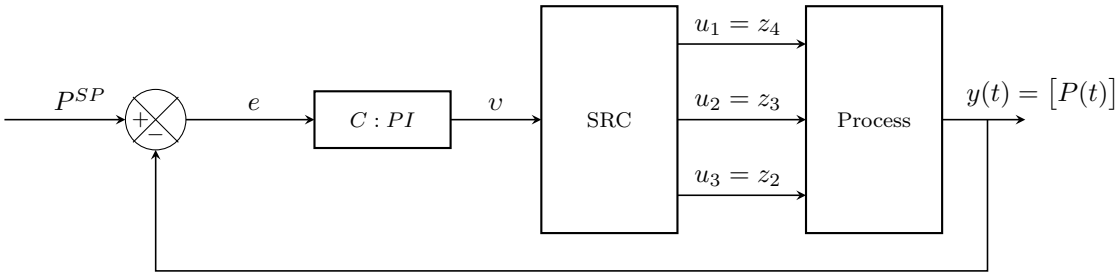

Figure 1: Block diagram showing controller logic for split range control control on consumer side, without implementation of selectors. That is, this block diagram illustrates split range control for control of only one system MV at a time.

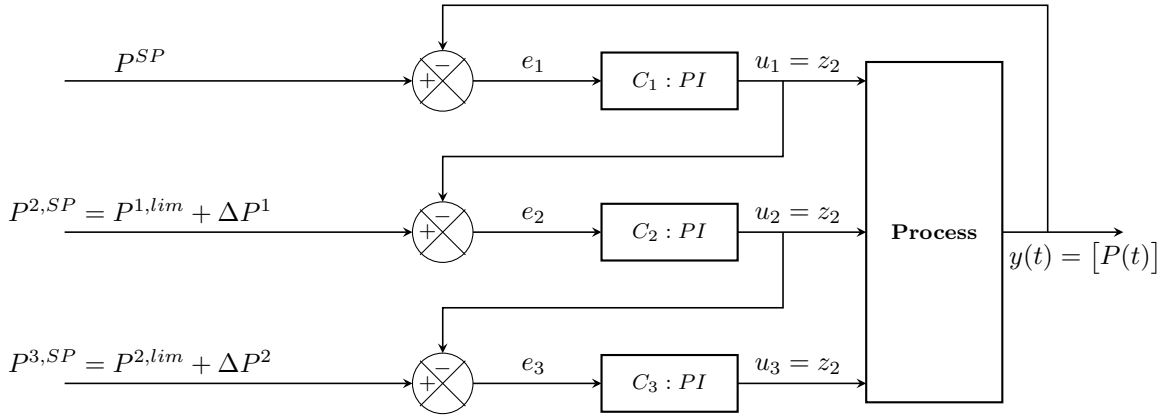

Figure 2: Block diagram showing controller logic for valve position control control on consumer side, without the use of selectors. This means this structure shows logic for controlling only one MV at a time

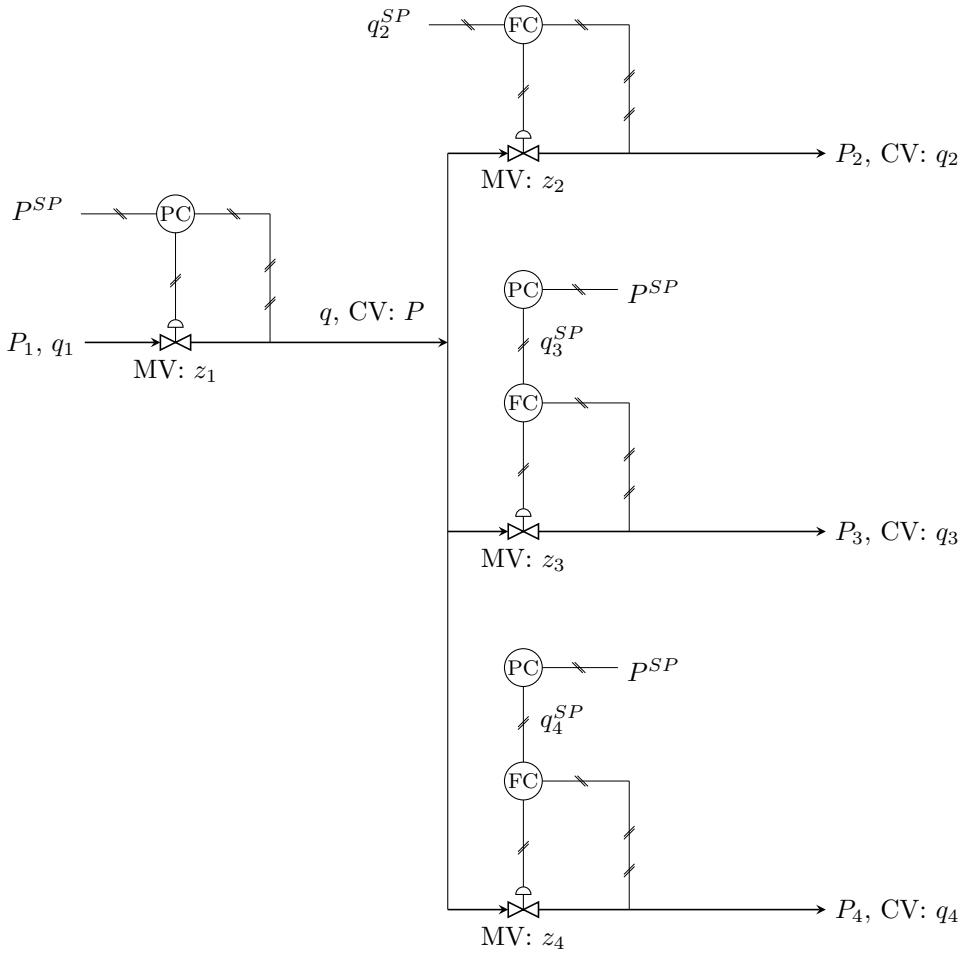

Figure 3: Simple illustration of droop control for the given system in this project.

MATLAB Code

The MATLAB files included in this project surrounds the simulink model of the ODE presented in MATLAB and Simulink implementation section. The MATLAB code involves a file defining the model and problem ODE with constraints, found in ODE code section. Then, the parameter file used to store all parameters is shown. Last, the main file used to simulate the project results is simulated. The files needs the Simulink model in order to run. To run the files in the project, the variable file should be run first to make a .mat file containing standard problem variables, then the main file should be run.

ODE Code

```
function out = P0PressureFlowODE(x,u,d,p)
% state: P
% CVs:   P,%%%%q1,q2
% MVs:   z1,z2
P     = x(1); % state. pressure in unknown pipe, this the one we want P in.
z1    = u(1); % -, Large valve opening
z2    = u(2); % -, Small valve opening
z3    = u(3); % -,Medium valve opening
z4    = u(4); % -,Medium valve opening
P1= d(1);
P2= d(2);
P3= d(3);
P4= d(4);
%P1    = p(1); % bar, reservoir pressure
%P2    = p(2); % bar, outlet pressure
%P3    = p(3); % bar, outlet pressure
Cv1   = p(1); % Large valve coeff
Cv2   = p(2); % Small valve coeff
Cv3   = p(3); % last valve coeff
Cv4   = p(4); % last valve coeff
V     = p(5); % m3, pipe1 volume
R     = p(6); % bar*m3/mol/K, gas constant
T     = p(7); % K, temperature
q     = Cv1*z1*sqrt(P1^2-P^2); %q = q1+q2; for simplification, se main
qsaturation = max(0,min(q,2.2)); %qmin is zero, min (q,10) represents qmax
q2    = Cv2*z2*sqrt(P^2-P2^2);
q3    = Cv3*z3*sqrt(P^2-P3^2);
q4    = Cv4*z4*sqrt(P^2-P4^2);
dPdt    = R*T/V*(qsaturation-q2-q3-q4); %use qsaturation instead of q
out   = [dPdt;qsaturation;q2;q3;q4];
end
```

Parameter File

```
% Plotting options
set(0,'DefaultTextFontName','Times',...
'DefaultTextFontSize',18,...
'DefaultAxesFontName','Times',...
'DefaultAxesFontSize',18,...
'DefaultLineLineWidth',2,...
'DefaultLineMarkerSize',7.75,...
'DefaultStairLineWidth',2)

savdir = '/Users/erikandrevik/Dropbox/MATLAB/Project2020/Erik2/88 Data from
↪  Simulations';

%% Small pressures. small flow.

%% Parameters

%load FixedValve.mat    % small pressures
%load Turbine.mat    % large pressures

%Initial Pressures
P1    = 15;                                    % [bar], reservoir pressure
Pnom = 10;                          % [bar], nominal pipe pressure after pump
P2 = 2;                                    % [bar], nominal pipe2 pressure
↪
P3 = 2;                                    % [bar], nominal pipe3 pressure
P4 = 2;

qnom  = 2.0;                                                % [kmol/s], flow
q2nom = 0.5; %Instead we use anom right now to split streams equally
q3nom = 0.75; %Instead we use anom right now to split streams equally
q4nom = 0.75; %Instead we use anom right now to split streams equally
%These are used now. for nominal

z1nom = 0.5;                    % [-], nominal valve opening
z2nom = 0.5;                    % [-], nominal valve opening
z3nom = 0.5;                    % [-], nominal valve opening
z4nom = 0.5;                    % [-], nominal valve opening

V     = 150;                                        % [m3] pipe volume
T     = 200+273;                                  % [K] constant temperature
R     = 8.314*1e-5;                             % [bar*m3/kmol/K] gas constant

anom=0.5;                           %[-] nominal split factor to change qnom
```

```
Cv1 = qnom/z1nom/sqrt(P1^2-Pnom^2);                              % Large valve coeff
Cv2 = 2/8*qnom/z2nom/sqrt(Pnom^2-P2^2)  ;%Different qnom for cv2 and cv3 bc split
↪
Cv3 = 3/8*qnom/z3nom/sqrt(Pnom^2-P3^2);                  % fixed valve coeff
Cv4 = 3/8*qnom/z4nom/sqrt(Pnom^2-P4^2);                  % fixed valve coeff

param = [Cv1;
    Cv2;
    Cv3;
    Cv4;
    V;
    R;
    T];

% For simulation - Dimensions
nx = 1;
nu =4;
np = size(param,1);
nz = 4;

Kdiff = [eye(nx) zeros(nx,nz)];            % selection matrix for diff var
Kalg  = [zeros(nz,nx) eye(nz)];             % selection matrix for alg var
tsim  = 200000;                                    % [s] simulation time
x0    = [Pnom];                                % [bar], initial conditions
%x0     = [1, 1, 0.5];
tstep = 100;                                          % [s] step time

%Disturbances
disturbP1 = P1;
disturbP2 = P2;
disturbP3 = P3;
disturbP4 = P4;



%% Open loop controller parameters

%% Controller parameters - tunings from 01
%PI Controller for pressure (Same as before)
controller_P_P  = 0;
controller_P_I  = 0;

%Outlet Controllers
controller_q2_P = 0;
controller_q3_P = 0;
controller_q4_P = 0;
```

```
controller_q2_I = 0; %k=1, theta(delay) = 0, tau_c = 1
controller_q3_I = 0; %k=1, theta(delay) = 0, tau_c = 1
controller_q4_I = 0; %k=1, theta(delay) = 0, tau_c = 1
controller_q3_Pdroop = 1;
controller_q3_Idroop = 0;
controller_q4_Pdroop = 1;
controller_q4_Idroop = 0;

%Anti Windup Parameters
Kb   = 0;
Kbq2 = 0;
Kbq3 = 0;
Kbq4 = 0;

%Droop loop on/off - 0 = off, 1 = on
droop1 = 0;
droop2 = 0;

%I control loop on/off - 0 = off, 1 = on
case1 = 0;

%P control loop on/off - 0 = off, 1 = on
case2 = 0;

%Nominal feed values, need to be given values in script
z1feed = 0;
z2feed = 0;
z3feed = 0;
z4feed = 0;

%Droop controllers step values - 0 as standard
droop1st = 0;
droop2st = 0;
%% Save to workspace
save('POVariables')
```

Plot File

```
function plottingFunction = POPlot()
load('POWorkspace.mat');

set(0,'DefaultTextFontName','Times',...
'DefaultTextFontSize',18,...
'DefaultAxesFontName','Times',...
'DefaultAxesFontSize',12,...
'DefaultLineLineWidth',2,...
```

```
'DefaultLineMarkerSize',7.75,...
'DefaultStairLineWidth',2)

figure(number)
y = sgtitle(plotname);
y.FontWeight = 'bold';
subplot(5,2,1)
plot(t, u(:,1), 'b')
xlabel('Time [s]')
ylabel('Opening z_1 [-]')
xlim([0 tsim])
ylim([0 1])

subplot(5,2,2)
plot(t,pstepplot,'-.b')
hold on
plot(t,P(:,1), 'r')
hold on
%legend('P^{sp}','P')
xlabel('Time [s]')
ylabel('P [bar]')
xlim([0 tsim])
ylim([8 12])

subplot(5,2,3)
plot(t, u(:,2), 'b')
xlabel('Time [s]')
ylabel('Opening z_2 [-]')
xlim([0 tsim])
ylim([0 1])

subplot(5,2,4)
plot(t,q2plot,'-.b')
hold on
plot(t,q(:,2), 'r')
hold on
%legend('q2^{sp}','q2')
xlabel('Time [s]')
ylabel('q_2 [kmol/s]')
xlim([0 tsim])
ylim([0 1.5])

subplot(5,2,5)
plot(t, u(:,3), 'b')
xlabel('Time [s]')
ylabel('Opening z_3 [-]')
```

```
xlim([0 tsim])
ylim([0 1])

subplot(5,2,6)
plot(t,q3plot,'-.b')
hold on
plot(t,q(:,3), 'r')
hold on
%legend('q3^{sp}','q3')
xlabel('Time [s]')
ylabel('q_3 [kmol/s]')
xlim([0 tsim])
ylim([0 1.5])

subplot(5,2,7)
plot(t, u(:,4), 'b')
xlabel('Time [s]')
ylabel('Opening z_4 [-]')
xlim([0 tsim])
ylim([0 1])

subplot(5,2,8)
plot(t,q4plot,'-.b')
hold on
plot(t,q(:,4), 'r')
hold on
%legend('q3^{sp}','q3')
xlabel('Time [s]')
ylabel('q_4 [kmol/s]')
xlim([0 tsim])
ylim([0 1.5])

subplot(5,2,9)
plot(0,0,'-.b')
hold on
plot(0,0, '-.g')
hold on
plot(0,0, 'r')
axis off
legend('Nominal Value/Setpoint/Step','Maximum Value','Actual value')

subplot(5,2,10)
plot(t,qnomplot,'-.g')
hold on
plot(t,q(:,1), 'r')
hold on
```

```
%legend('q^{sp}','q')
xlabel('Time [s]')
ylabel('q [kmol/s]')
xlim([0 tsim])
ylim([0 2.2])
end
```

Main Simulation File

```
clc;
clear;
close all

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Author: Erik Andre Klepp Vik
%% Mail: eavik@stud.ntnu.no
%% Date: 18.12.2020
%% Notes: Built on tube and valve model supplied by Cristina F. Zotica
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Description:
% This is the main file for showing disturbance rejection for different
% advanced controller structures.
%
% NOTE: To run this file, POVariables.m must be run first!
%
% Tunings are obtained through open loop responses.
% This document only implements the final tunings applied to different
% controller structures and the corresponding responses for a positive
% and negative step in P1. Other disturbances and steps can also be
% implemented.
%
%  - For plotting file, see POPlot.m
%
%  - For ODE file, see POPressureFlowODE.m
%
%  - For Simulink file, see POPressureFlowODE.m
%
%  - For variable file, see POVariables.m
%
% NOTE: For this file to run, POVariables.m should be run first!
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% First Simulation - Open loop responses
```

```
% Load file to reset variables for new simulation

load('P0Variables')

% Select relevant feedback loops using gains in Simulink:
% (Does not really matter for open loop simulation)
case1 = 1;
z1feed = Pnom;
z2feed = q2nom;
z3feed = q3nom;
z4feed = q4nom;

% Simulate step in z1, all loops open

z1st   = 0;
z2st   = 0;
z3st   = 0;
z4st   = 0;
z1dist = z1nom*0.1;
z2dist = 0;
z3dist = 0;
z4dist = 0;
P1dist = 0;
P2dist = 0;
P3dist = 0;
P4dist = 0;

sim('P0PressureFlowODESim')

qnomplot = t.^(0)*2.2;
q2plot   = q2stepplot;
q3plot   = q3stepplot;
q4plot   = q4stepplot;

save(fullfile(savdir,'P9901CL'));

% Plot pressures and and flows

plotname = 'Open loop response, +10% Step in z_1';
number   = 1;
save('P0Workspace');
P0Plot();

% Simulate step in z2, all loops open
```

```
z1st   = 0;
z2st   = 0;
z3st   = 0;
z4st   = 0;
z1dist = 0;
z2dist = z2nom*0.1;
z3dist = 0;
z4dist = 0;
P1dist = 0;
P2dist = 0;
P3dist = 0;
P4dist = 0;

sim('P0PressureFlowODESim')

qnomplot = t.^(0)*2.2;
q2plot   = q2stepplot;
q3plot   = q3stepplot;
q4plot   = q4stepplot;

save(fullfile(savdir,'P9902CL'));

% Plot pressures and and flows

plotname = 'Open loop response, +10% Step in z_2';
number   = 2;
save('P0Workspace');
P0Plot();

%% Second Simulation - Default Control using Supplier and Consumers
% PI-controller for z1, I-controllers for z2,z3,z4
% Load file to reset variables for new simulation

load('P0Variables')

% Select relevant feedback loops using gains in Simulink:
case1 = 1;
z1feed = Pnom;
z2feed = q2nom;
z3feed = q3nom;
z4feed = q4nom;

% Tunings all controllers tuned.
tauC = V/((qnom*R*T)/Pnom);

k_p     = 0.51/0.05;      %10
```

```
tau_p   = 10000;            %57
theta_p = 0;              %0
tau_p_c = tauC;         %25    %1
tau_p_I = min(tau_p,4*tau_p_c);    %4

controller_P_P  = 1/k_p * tau_p/(theta_p+tau_p_c);
controller_P_I  = controller_P_P/tau_p_I;

% Outlet Controllers
k2     = 0.05/0.05;
theta2 = 0;
k3     = 0.075/0.05;
theta3 = 0;
k4     = 0.075/0.05;
theta4 = 0;
controller_q2_I = 1/k2 * 1/(theta2+tau_p_c);
controller_q3_I = 1/k3 * 1/(theta3+tau_p_c);
controller_q4_I = 1/k4 * 1/(theta4+tau_p_c);

% Anti Windup Parameters
Kb   = controller_P_I;
Kbq2 = controller_q2_I;
Kbq3 = controller_q3_I;
Kbq4 = controller_q4_I;

% Simulate SP change for P, to see PI controller response
z1st   = 0;                                    % Pressure SP input
z2st   = 0;                                    % q2 SP input
z3st   = 0;                                    % q3 SP input
z4st   = 0;                                    % q4 SP input
z1dist = 0;                                    % Valve one input
z2dist = 0;                                    % Valve one input
z3dist = 0;                                    % Valve one input
z4dist = 0;                                    % Valve one input
P1dist = P1;                                   % Disturbance P1 input
P2dist = 0;                                    % Disturbance P2 input
P3dist = 0;                                    % Disturbance P3 input
P4dist = 0;                                    % Disturbance P4 input

sim('POPressureFlowODESim')                    % Run Simulink file

qnomplot = t.^(0)*2.2;                         % Prepare plotting variables
q2plot   = q2stepplot;                         % Prepare plotting variables
q3plot   = q3stepplot;                         % Prepare plotting variables
q4plot   = q4stepplot;                         % Prepare plotting variables
```

```
save(fullfile(savdir,'P9903CL'));              % Save variables for later

% Plot pressures and and flows
%pressures
plotname = 'Default Supplier and Consumer control, +100% Dist on P1';
number   = 3;
save('POWorkspace');
POPlot();

% Simulate SP change for P, to see PI controller response
z1st   = 0;                                    % Pressure SP input
z2st   = 0;                                    % q2 SP input
z3st   = 0;                                    % q3 SP input
z4st   = 0;                                    % q4 SP input
z1dist = 0;                                    % Valve one input
z2dist = 0;                                    % Valve one input
z3dist = 0;                                    % Valve one input
z4dist = 0;                                    % Valve one input
P1dist = -P1*0.3;                              % Disturbance P1 input
P2dist = 0;                                    % Disturbance P2 input
P3dist = 0;                                    % Disturbance P3 input
P4dist = 0;                                    % Disturbance P4 input

sim('POPressureFlowODESim')                    % Run Simulink file

qnomplot = t.^(0)*2.2;                         % Prepare plotting variables
q2plot   = q2stepplot;                         % Prepare plotting variables
q3plot   = q3stepplot;                         % Prepare plotting variables
q4plot   = q4stepplot;                         % Prepare plotting variables

save(fullfile(savdir,'P9904CL'));              % Save variables for later

% Plot pressures and and flows
% pressures
plotname = 'Default Supplier and Consumer control, -30% Dist on P1';
number   = 4;
save('POWorkspace');
POPlot();

%% Third Simulation - Parallel Control
% PI controller for z2, P-controllers for z3,z4

% Load file to reset variables for new simulation
load('POVariables')

% Select relevant feedback loops and feeds using gains in Simulink:
```

```
case2 = 1;
z1feed = Pnom; %Only used for plotting, not used becasue controller not on.
z2feed = Pnom;
z3feed = Pnom;
z4feed = Pnom;

% Tune PI controller for P control
tauC = V/((qnom*R*T)/Pnom);

k_p2     = -0.058/0.05;      %10
tau_p2   = 5000;           %57
theta_p2 = 0;            %0
tau_p_c2 = tauC;        %25    %1
tau_p_I2 = min(tau_p2,4*tau_p_c2);    %4

controller_q2_P  = 1/k_p2 * tau_p2/(theta_p2+tau_p_c2);
controller_q2_I  = controller_q2_P/tau_p_I2;

k_p3     = -0.201/0.05;
tau_p3   = 12500;
theta_p3 = 0;
tau_p_c3 = tauC;
tau_p_I3 = min(tau_p3,4*tau_p_c3);

controller_q3_P  = 1/k_p3 * tau_p3/(theta_p3+tau_p_c3);

k_p4     = -0.122/0.05;
tau_p4   = 7500;
theta_p4 = 0;
tau_p_c4 = tauC;
tau_p_I4 = min(tau_p4,4*tau_p_c4);

controller_q4_P = 1/k_p4 * tau_p4/(theta_p4+tau_p_c4);

% Anti Windup Parameters
Kbq2   = controller_q2_I;
% Simulate SP change for P, to see PI controller response
z1st   = 0;                                    % Pressure SP input
z2st   = 0;                                    % q2 SP input
z3st   = 0;                                    % q3 SP input
z4st   = 0;                                    % q4 SP input
z1dist = 0;                                    % Valve one input
z2dist = 0;                                    % Valve one input
z3dist = 0;                                    % Valve one input
z4dist = 0;                                    % Valve one input
P1dist = P1;                                   % Disturbance P1 input
```

```
P2dist = 0;                                    % Disturbance P2 input
P3dist = 0;                                    % Disturbance P3 input
P4dist = 0;                                    % Disturbance P4 input

sim('P0PressureFlowODESim')                    % Run Simulink file

pstepplot = q2stepplot;                        % Preparing plotting
qnomplot = t.^(0)*2.2;                         % Preparing plotting
q2plot   = t.^(0)*0.5;                         % Preparing plotting
q3plot   = t.^(0)*0.75;                        % Preparing plotting
q4plot   = t.^(0)*0.75;                        % Preparing plotting

save(fullfile(savdir,'P9905CL'));              % Save variables for later

% Plot pressures and and flows
%pressures
plotname = 'Parallel Control, +100% Dist on P1';
number   = 5;
save('P0Workspace');
P0Plot();

% Simulate SP change for P, to see PI controller response
z1st   = 0;                                    % Pressure SP input
z2st   = 0;                                    % q2 SP input
z3st   = 0;                                    % q3 SP input
z4st   = 0;                                    % q4 SP input
z1dist = 0;                                    % Valve one input
z2dist = 0;                                    % Valve one input
z3dist = 0;                                    % Valve one input
z4dist = 0;                                    % Valve one input
P1dist = -P1*0.3;                              % Disturbance P1 input
P2dist = 0;                                    % Disturbance P2 input
P3dist = 0;                                    % Disturbance P3 input
P4dist = 0;                                    % Disturbance P4 input

sim('P0PressureFlowODESim')                    % Run Simulink file

pstepplot = q2stepplot;                        % Preparing plotting
qnomplot = t.^(0)*2.2;                         % Preparing plotting
q2plot   = t.^(0)*0.5;                         % Preparing plotting
q3plot   = t.^(0)*0.75;                        % Preparing plotting
q4plot   = t.^(0)*0.75;                        % Preparing plotting

save(fullfile(savdir,'P9906CL'));              % Save variables for later

% Plot pressures and and flows
```

```
%pressures
plotname = 'Parallel Control, -30% Dist on P1';
number   = 6;
save('P0Workspace');
P0Plot();

%% Fourth Simulation - Different Setpoint Control
% PI-controllers for z2,z3,z4

% Load file to reset variables for new simulation
load('P0Variables')

% Select relevant feedback loops and feeds using gains in Simulink:
case2 = 1;
z1feed = Pnom; %Only used for plotting, in reality 0.
z2feed = Pnom-1; %This is where we change setpoints.
z3feed = Pnom-0.5; %This is where we change setpoints.
z4feed = Pnom; %This is where we change setpoints.

% All controllers tuned
tauC = V/((qnom*R*T)/Pnom);

k_p2     = -0.135/0.05;
tau_p2   = 12000;
theta_p2 = 0;
tau_p_c2 = tauC;
tau_p_I2 = min(tau_p2,4*tau_p_c2);

controller_q2_P  = 1/k_p2 * tau_p2/(theta_p2+tau_p_c2);
controller_q2_I  = controller_q2_P/tau_p_I2;

k_p3     = -0.094/0.05;
tau_p3   = 4000;
theta_p3 = 0;
tau_p_c3 = tauC;
tau_p_I3 = min(tau_p3,4*tau_p_c3);

controller_q3_P  = 1/k_p3 * tau_p3/(theta_p3+tau_p_c3);
controller_q3_I  = controller_q3_P/tau_p_I3;

k_p4     = -0.066/0.05;
tau_p4   = 3000;
theta_p4 = 0;
tau_p_c4 = tauC;
tau_p_I4 = min(tau_p4,4*tau_p_c4);
```

```
controller_q4_P  = 1/k_p4 * tau_p4/(theta_p4+tau_p_c4);
controller_q4_I  = controller_q4_P/tau_p_I4;


%Anti Windup Parameters
Kbq2   = controller_q2_I;
Kbq3   = controller_q3_I;
Kbq4   = controller_q4_I;

% Simulate SP change for P, to see PI controller response
z1st   = 0;                                    % Pressure SP input
z2st   = 0;                                    % q2 SP input
z3st   = 0;                                    % q3 SP input
z4st   = 0;                                    % q4 SP input
z1dist = 0;                                    % Valve one input
z2dist = 0;                                    % Valve one input
z3dist = 0;                                    % Valve one input
z4dist = 0;                                    % Valve one input
P1dist = P1;                                   % Disturbance P1 input
P2dist = 0;                                    % Disturbance P2 input
P3dist = 0;                                    % Disturbance P3 input
P4dist = 0;                                    % Disturbance P4 input

sim('P0PressureFlowODESim')                    % Run Simulink file

pstepplot = q2stepplot;                        % Preparing plotting
qnomplot = t.^(0)*2.2;                         % Preparing plotting
q2plot   = t.^(0)*0.5;                         % Preparing plotting
q3plot   = t.^(0)*0.75;                        % Preparing plotting
q4plot   = t.^(0)*0.75;                        % Preparing plotting

save(fullfile(savdir,'P9907CL'));              % Save variables for later

% Plot pressures and and flows
%pressures
plotname = 'Different Setpoints, +100% Dist on P1';
number   = 7;
save('P0Workspace');
P0Plot();

% Simulate SP change for P, to see PI controller response
z1st   = 0;                                    % Pressure SP input
z2st   = 0;                                    % q2 SP input
z3st   = 0;                                    % q3 SP input
z4st   = 0;                                    % q4 SP input
z1dist = 0;                                    % Valve one input
```

```
z2dist = 0;                                     % Valve one input
z3dist = 0;                                     % Valve one input
z4dist = 0;                                     % Valve one input
P1dist = -P1*0.3;                               % Disturbance P1 input
P2dist = 0;                                     % Disturbance P2 input
P3dist = 0;                                     % Disturbance P3 input
P4dist = 0;                                     % Disturbance P4 input

sim('P0PressureFlowODESim')                     % Run Simulink file

pstepplot = q2stepplot;                         % Preparing plotting
qnomplot = t.^(0)*2.2;                          % Preparing plotting
q2plot   = t.^(0)*0.5;                          % Preparing plotting
q3plot   = t.^(0)*0.75;                         % Preparing plotting
q4plot   = t.^(0)*0.75;                         % Preparing plotting

save(fullfile(savdir,'P9908CL'));               % Save variables for later

% Plot pressures and and flows
%pressures
plotname = 'Different Setpoints, -30% Dist on P1';
number   = 8;
save('P0Workspace');
P0Plot();

%% Fifth Simulation - Droop Control
% I-controllers for z2,z3,z4 P-controllers for q3sp and q4sp
% Load file to reset variables for new simulation

load('P0Variables')

% Select relevant feedback loops using gains in Simulink:
droop1 = 1;
droop2 = 1;
case1 = 1;
z1feed = Pnom;
z2feed = q2nom;
z3feed = q3nom;
z4feed = q4nom;

%% Tunings all controllers tuned.
tauC = V/((qnom*R*T)/Pnom);

k_p     = 0.51/0.05;
tau_p   = 10000;
theta_p = 0;
```

```
tau_p_c = tauC;
tau_p_I = min(tau_p,4*tau_p_c);

controller_P_P  = 1/k_p * tau_p/(theta_p+tau_p_c);
controller_P_I  = controller_P_P/tau_p_I;

%Outlet Controllers
k2     = 0.05/0.05;
theta2 = 0;
k3     = 0.0825/0.05;
theta3 = 0;
k4     = 0.08787/0.05;
theta4 = 0;
controller_q2_I = 1/k2 * 1/(theta2+tau_p_c);
controller_q3_I = 1/k3 * 1/(theta3+tau_p_c);
controller_q4_I = 1/k4 * 1/(theta4+tau_p_c);

%Anti Windup Parameters
Kb   = controller_P_I;
Kbq2 = controller_q2_I;
Kbq3 = controller_q3_I;
Kbq4 = controller_q4_I;

k_p4     = -0.57/0.05;
tau_p4   = 17000;
theta_p4 = 0;
tau_p_c4 = tauC;
tau_p_I4 = min(tau_p4,4*tau_p_c4);

k_p3     = -0.86/0.05;
tau_p3   = 17000;
theta_p3 = 0;
tau_p_c3 = tauC;
tau_p_I3 = min(tau_p3,4*tau_p_c3);

controller_q3_Pdroop = 1/k_p3 * tau_p3/(theta_p+tau_p_c3);
controller_q4_Pdroop = 1/k_p4 * tau_p4/(theta_p+tau_p_c4);

% Simulate SP change for P, to see PI controller response
z1st   = 0;                                    % Pressure SP input
z2st   = 0;                                    % q2 SP input
z3st   = 0;                                    % q3 SP input
z4st   = 0;                                    % q4 SP input
z1dist = 0;                                    % Valve one input
z2dist = 0;                                    % Valve one input
z3dist = 0;                                    % Valve one input
```

```
z4dist = 0;                                 % Valve one input
P1dist = P1;                                % Disturbance P1 input
P2dist = 0;                                 % Disturbance P2 input
P3dist = 0;                                 % Disturbance P3 input
P4dist = 0;                                 % Disturbance P4 input

sim('P0PressureFlowODESim')                 % Run Simulink file

qnomplot = t.^(0)*2.2;                      % Prepare plotting variables
q2plot   = q2stepplot;                      % Prepare plotting variables
q3plot   = q3stepplot;                      % Prepare plotting variables
q4plot   = q4stepplot;                      % Prepare plotting variables

save(fullfile(savdir,'P9909CL'));           % Save variables for later

% Plot pressures and and flows
%pressures
plotname = 'Droop Control, +100% Dist on P1';
number   = 9;
save('P0Workspace');
P0Plot();

% Simulate SP change for P, to see PI controller response
z1st   = 0;                                 % Pressure SP input
z2st   = 0;                                 % q2 SP input
z3st   = 0;                                 % q3 SP input
z4st   = 0;                                 % q4 SP input
z1dist = 0;                                 % Valve one input
z2dist = 0;                                 % Valve one input
z3dist = 0;                                 % Valve one input
z4dist = 0;                                 % Valve one input
P1dist = -P1*0.3;                           % Disturbance P1 input
P2dist = 0;                                 % Disturbance P2 input
P3dist = 0;                                 % Disturbance P3 input
P4dist = 0;                                 % Disturbance P4 input

sim('P0PressureFlowODESim')                 % Run Simulink file

qnomplot = t.^(0)*2.2;                      % Prepare plotting variables
q2plot   = q2stepplot;                      % Prepare plotting variables
q3plot   = q3stepplot;                      % Prepare plotting variables
q4plot   = q4stepplot;                      % Prepare plotting variables

save(fullfile(savdir,'P9910CL'));           % Save variables for later

% Plot pressures and and flows
```

```
%pressures
plotname = 'Droop Control, -30% Dist on P1';
number   = 10;
save('P0Workspace');
P0Plot();

% %% Save Figures as .png and eps
% savdir = '/Users/erikandrevik/Dropbox/MATLAB/Project2020/Erik2/88 Plots';
% saveas(figure(1),fullfile(savdir,'P8801CL'),'epsc');
% saveas(figure(1),fullfile(savdir,'P8801CL.png'));
% saveas(figure(2),fullfile(savdir,'P8802CL'),'epsc');
% saveas(figure(2),fullfile(savdir,'P8802CL.png'));
% saveas(figure(3),fullfile(savdir,'P8803CL'),'epsc');
% saveas(figure(3),fullfile(savdir,'P8803CL.png'));
% saveas(figure(4),fullfile(savdir,'P8804CL'),'epsc');
% saveas(figure(4),fullfile(savdir,'P8804CL.png'));
% saveas(figure(5),fullfile(savdir,'P8805CL'),'epsc');
% saveas(figure(5),fullfile(savdir,'P8805CL.png'));
% saveas(figure(6),fullfile(savdir,'P8806CL'),'epsc');
% saveas(figure(6),fullfile(savdir,'P8806CL.png'));
% saveas(figure(7),fullfile(savdir,'P8807CL'),'epsc');
% saveas(figure(7),fullfile(savdir,'P8807CL.png'));
% saveas(figure(8),fullfile(savdir,'P8808CL'),'epsc');
% saveas(figure(8),fullfile(savdir,'P8808CL.png'));
% saveas(figure(9),fullfile(savdir,'P8809CL'),'epsc');
% saveas(figure(9),fullfile(savdir,'P8809CL.png'));
% saveas(figure(10),fullfile(savdir,'P8810CL'),'epsc');
% saveas(figure(10),fullfile(savdir,'P8810CL.png'));

clc;
clear;
close all

%% Plotting all results in same plots
Path = "/Users/erikandrevik/Dropbox/MATLAB/Project2020/Erik2/88 Data from
↪  Simulations";
addpath(Path);

ListOfPlots = ["P9901CL";
    "P9902CL";
    "P9903CL";
    "P9904CL";
    "P9905CL";
    "P9906CL";
    "P9907CL";
    "P9908CL";
```

```
    "P9909CL";
    "P9910CL"];

%% Plotting + 100% Disturbance from P1
for i = 3:2:length(ListOfPlots)
load(ListOfPlots(i))

pstepplot = t.^(0)*10;                          % Preparing plotting
qnomplot = t.^(0)*2.2;                           % Preparing plotting
q2plot   = t.^(0)*0.5;                           % Preparing plotting
q3plot   = t.^(0)*0.75;                          % Preparing plotting
q4plot   = t.^(0)*0.75;                          % Preparing plotting

figure(11)
y = sgtitle('Closed Loop Response, -30% Disturbance from P1');
y.FontWeight = 'bold';
subplot(5,2,1)
plot(t, u(:,1))
hold on
xlabel('Time [s]')
ylabel('Opening z_1 [-]')
xlim([0 tsim])
ylim([0 1])

subplot(5,2,2)
plot(t,P(:,1))
hold on
xlabel('Time [s]')
ylabel('P [bar]')
xlim([0 tsim])
ylim([8 12])

subplot(5,2,3)
plot(t, u(:,2))
hold on
xlabel('Time [s]')
ylabel('Opening z_2 [-]')
xlim([0 tsim])
ylim([0 1])

subplot(5,2,4)
plot(t,q(:,2))
hold on
xlabel('Time [s]')
ylabel('q_2 [kmol/s]')
xlim([0 tsim])
```

```
ylim([0 1.5])

subplot(5,2,5)
plot(t, u(:,3))
hold on
xlabel('Time [s]')
ylabel('Opening z_3 [-]')
xlim([0 tsim])
ylim([0 1])

subplot(5,2,6)
plot(t,q(:,3))
hold on
xlabel('Time [s]')
ylabel('q_3 [kmol/s]')
xlim([0 tsim])
ylim([0 1.5])

subplot(5,2,7)
plot(t, u(:,4))
hold on
xlabel('Time [s]')
ylabel('Opening z_4 [-]')
xlim([0 tsim])
ylim([0 1])

subplot(5,2,8)
plot(t,q(:,4))
hold on
xlabel('Time [s]')
ylabel('q_4 [kmol/s]')
xlim([0 tsim])
ylim([0 1.5])

subplot(5,2,10)
plot(t,q(:,1))
hold on
xlabel('Time [s]')
ylabel('q [kmol/s]')
xlim([0 tsim])
ylim([0 2.2])
end

figure(11)
subplot(5,2,2)
plot(t,pstepplot,'-.b')
```

```
hold on

subplot(5,2,4)
plot(t,q2plot,'-.b')
hold on

subplot(5,2,6)
plot(t,q3plot,'-.b')
hold on

subplot(5,2,8)
plot(t,q4plot,'-.b')
hold on

subplot(5,2,10)
plot(t,qnomplot,'-.b')
hold on

subplot(5,2,9)
plot(0,0, 0,0, 0,0, 0,0, 0,0, '-.b')
axis off
legend('1: Default Control',...
    '2: Parallel control Using Equal Setpoints', ...
    '3: Parallel Control Using Different Setpoints',...
    '4: Droop Control',...
    'Nominal Value q_2-q_4/Maximum Value q',...
    'location','southwest')

%% Plotting -30% Disturbance from P1
for i = 4:2:length(ListOfPlots)
load(ListOfPlots(i))

pstepplot = t.^(0)*10;                          % Preparing plotting
qnomplot = t.^(0)*2.2;                           % Preparing plotting
q2plot   = t.^(0)*0.5;                           % Preparing plotting
q3plot   = t.^(0)*0.75;                          % Preparing plotting
q4plot   = t.^(0)*0.75;                          % Preparing plotting

figure(12)
y = sgtitle('Closed Loop Response, -30% Disturbance from P1');
y.FontWeight = 'bold';
subplot(5,2,1)
plot(t, u(:,1))
hold on
xlabel('Time [s]')
ylabel('Opening z_1 [-]')
```

```
xlim([0 tsim])
ylim([0 1])

subplot(5,2,2)
plot(t,P(:,1))
hold on
xlabel('Time [s]')
ylabel('P [bar]')
xlim([0 tsim])
ylim([8 12])

subplot(5,2,3)
plot(t, u(:,2))
hold on
xlabel('Time [s]')
ylabel('Opening z_2 [-]')
xlim([0 tsim])
ylim([0 1])

subplot(5,2,4)
plot(t,q(:,2))
hold on
xlabel('Time [s]')
ylabel('q_2 [kmol/s]')
xlim([0 tsim])
ylim([0 1.5])

subplot(5,2,5)
plot(t, u(:,3))
hold on
xlabel('Time [s]')
ylabel('Opening z_3 [-]')
xlim([0 tsim])
ylim([0 1])

subplot(5,2,6)
plot(t,q(:,3))
hold on
xlabel('Time [s]')
ylabel('q_3 [kmol/s]')
xlim([0 tsim])
ylim([0 1.5])

subplot(5,2,7)
plot(t, u(:,4))
hold on
```

```
xlabel('Time [s]')
ylabel('Opening z_4 [-]')
xlim([0 tsim])
ylim([0 1])

subplot(5,2,8)
plot(t,q(:,4))
hold on
xlabel('Time [s]')
ylabel('q_4 [kmol/s]')
xlim([0 tsim])
ylim([0 1.5])

subplot(5,2,10)
plot(t,q(:,1))
hold on
xlabel('Time [s]')
ylabel('q [kmol/s]')
xlim([0 tsim])
ylim([0 2.2])
end

figure(12)
subplot(5,2,2)
plot(t,pstepplot,'-.b')
hold on

subplot(5,2,4)
plot(t,q2plot,'-.b')
hold on

subplot(5,2,6)
plot(t,q3plot,'-.b')
hold on

subplot(5,2,8)
plot(t,q4plot,'-.b')
hold on

subplot(5,2,10)
plot(t,qnomplot,'-.b')
hold on

subplot(5,2,9)
plot(0,0, 0,0, 0,0, 0,0, 0,0, '-.b')
axis off
```

```
legend('1: Default Control',...
    '2: Parallel control Using Equal Setpoints', ...
    '3: Parallel Control Using Different Setpoints',...
    '4: Droop Control',...
    'Nominal Value q_2-q_4/Maximum Value q',...
    'location','southwest')
```

Simulink Model

In Figure 4 the overall model used together with the MATLAB code is shown. It is found in the file P0PressureFlowODESim.slx. The Simulink file is built up using four PI-controllers, one for each possible MV in the system. The tuning parameters for these controllers are set to zero as standard. The system also includes two P-controllers on top of the controllers for z_3 and z_4. The P-gain for those controllers are set to one as standard, making the system an open loop system as a standard simulation with nominal value input from $z1feed$, $z2feed$, $z3feed$ and $z4feed$.

Gains are used to select which MVs are used in the simulations. To choose whether to use P, q_2, q_3 and q_4 or only P as a CV, gains must be set to one for $case1$ or $case2$. In order to implement droop control, also $droop1$ and $droop2$ must be also be set to one.

Disturbances can be implemented from P_1, P_2, P_3 or P_4 using the variables $P1dist$, $P2dist$, $P3dist$ or $P4dist$ They can be found and modified in the main file, P0Main.m. Parameter values can be changed using the parameter list. Parameters are found in the P0Variables.m file.

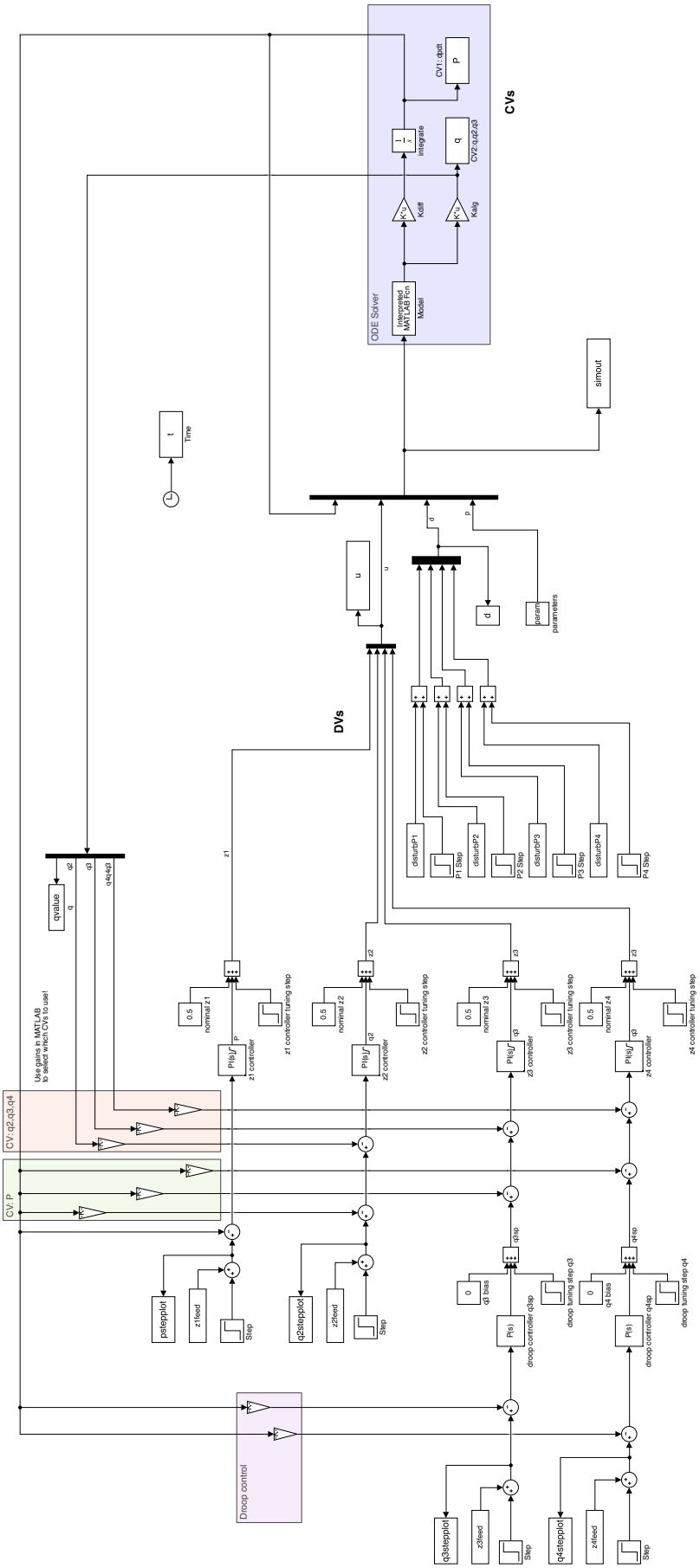

Figure 4: Screenshot from the full Simulink solver.