Utgångspunkter för modellprediktiv reglering av integrerande processer

Diplomarbete i kemi- och processteknik

Jonathan Häggvik

Handledare: Jari Böling

Fakulteten för naturvetenskaper och teknik

Åbo Akademi

30/03/2023

Abstrakt för diplomarbetet

Åbo Akademi
Naturvetenskaper och teknik

Diplomarbete i kemi- och processteknik

Utgångspunkter för modellprediktiv reglering av integrerande processer

Handledare: Jari Böling

Modellprediktiv reglering är en modellbaserad reglermetod där den framtida responsen av ett styrt system beräknas med hjälp av en matematisk modell, och för att reglera systemet väljs de optimala styråtgärderna med hjälp av optimering. Det finns ett antal olika regleralgoritmer som använder olika modeller för att uppskatta beteendet av processen. Läsaren av arbetet bekantar sig med några av dessa modeller och de grundläggande begreppen inom modellprediktiv reglering. I den experimentella delen av arbetet skapas en simuleringsomgivning av ett behållarsystem med integrerande dynamik, eftersom den integrerande dynamiken är bland annat utmanande för dynamic-matrix-control-algoritmen. Därför skapas också en sådan algoritm för att utforska varför den integrerande verkan av processen orsakar problem. För att styra simuleringsomgivningen prövades algoritmen med en vanlig prediktionsmodell och en modell som blivit anpassad för den integrerande dynamiken. Prestationen av de två modellerna jämfördes både numeriskt och grafiskt. Modellen som är anpassad för processer utan stationärtillstånd presterade bättre än algoritmen med den vanliga modellen.

Modellprediktiv reglering, prediktionsmodeller, reglerteknik, integrerande processer

Datum: 30/03/2023 Sidantal: 79

Förord

Ett stort tack till NAPCON för förslaget av ämnet och för möjligheten att skriva diplomarbetet i samband med jobbet. Ett tack till Stefan Tötterman, Samuli Bergman och Jari Böling som varit delaktiga i processen.

Jag bör tacka Eleonora för hennes tålamod och hårda arbete hemma medan jag jobbat på diplomarbetet. Jag vill också tacka min mamma Gun Häggvik för korrekturläsningen av arbetet.

I would like to thank NAPCON for the topic and possibility to make this work a reality. I would also like to thank Stefan Tötterman, Samuli Bergman and Jari Böling who were involved during the process.

Innehållsförteckning

Nomenklatur

y	Teoretiska modellen/Predikterade utsignalen
$\hat{y}$	Empiriska modellen/Uppmätta utsignalen
d_k	Förskjutningsterm/felet
u	Insignal
Δu	Insignalens förändring
r	Börvärdet
M	Modellhorisont
P	Prediktionshorisont
S	Styrhorisont
H, h	Stegsvaret och dess koefficienter
m	Massa
z	Höjd
V	Volym
$\dot{V}$	Volymflöde
$P_{förlust}$	Effektförlust
G, g	Impulssvaret och dess koefficienter
ζ'	Friktions förlustfaktor
C	Matris med stegkoefficienter i Toeplitz-mönster
τ_{CL}	Tidskonstanten för den slutna kretsen

1 Inledning

På 1950-talet byttes fabriksanläggningens pneumatiska styrkretsar mot den elektroniska motsvarigheten. Ett årtionde senare befann sig den första datorn på fabriksgolvet. Sedan det utvecklades avancerade industriella kommunikationssystem och beräkningsförmågan ökade. Tack vare dessa framsteg blev det möjligt att implementera mera ambitiösa reglermetoder än den klassiska PID-regulatorn. Modellprediktiv reglering (*Model Predictive Control*, MPC) utvecklades mot slutet av 1970-talet. En drivande kraft var processindustrin, där uppmärksamheten styrdes mot metoder för att kontrollera hela fabriksanläggningen. Modellprediktiv reglering blev en attraktiv lösning i och med att det var möjligt att styra processer som påverkades av flera styrsignaler med en regulator och dessutom införa villkor. I dag anses modellprediktiv reglering vara ett standardtillvägagångssätt för begränsade multivariabla reglerproblem.

Det finns ett antal tolkningar av tillvägagångssättet, vissa bättre än andra. Detta arbete behandlar de grundläggande beståndsdelarna av en modellprediktiv regleralgoritm och fördjupar sig i modeller för att uppskatta det framtida beteendet av ett system. För att få en helhetsbild behandlas också systemidentifiering. Informationen för de förutnämnda ämnen forskades genom litteraturforskning.

I den praktiska delen av arbetet undersöks en brist för många modellprediktiva regleralgoritmer, nämligen processer med integrerande dynamik. För att få en uppfattning av problemet skapas en simuleringsomgivning med integrerande dynamik och en enkel modellprediktiv algoritm, och med den skapade algoritmen prövas en metod för att lösa problemet (Lundström, et al., 1995).

Premissen för arbetet kan sammanfattas enligt följande. Processer med integrerande dynamik har visat sig vara en utmaning för vissa modellprediktiva algoritmer. Att styra en integrerande process är i regel inte ett problem, eftersom flera av de olika algoritmerna utnyttjar stegförändringen Δu i beräkningen av prediktionen som ger

implicit integrerande verkan. Däremot kan en varierande integrerad störning orsaka problem för regulatorn som visar sig som en förskjutning mellan börvärdet och utsignal.

2 Modellprediktiv reglering

Modellprediktiv reglering är ett tillvägagångssätt för att lösa reglerproblem. Tillvägagångssättet kan anses vara en reflektion av hur vi människor närmar oss reglerproblem. En modellprediktivregulator är en regleralgoritm som utnyttjar en matematisk tolkning av ett dynamiskt system. Med hjälp av den matematiska modellen uppskattar algoritmen det framtida beteendet av systemet för att sedan beräkna den bästa styråtgärden för att nå börvärdet. En liknelse som ofta används i texter om modellprediktiv reglering är bilkörning (Rossiter, 2018).

Anta att en chaufför kör längs en väg, och chauffören har bra sikt över hur vägen vrider och vänder. Chaufförens mål är att hålla hastigheten vid ett mål och hålla bilen på vägen. Chauffören kan bekräfta sin hastighet på hastighetsmätaren och ser om bilen avviker från körbanan.

Analogin innehåller flera beståndsdelar som kan hittas i modellprediktiv reglering, följande stycken kommer att behandla dessa beståndsdelar i detalj och använda analogin som stöd i förklaringen. Uppdelningen är aningen förenklad och generaliserar modellprediktiv reglering, men läsaren torde få en uppfattning om hur modellprediktiv reglering fungerar. I förarproblemet antas föraren vara i rörelse och reglerar sin hastighet endast med gaspedalen och sin riktning med ratten. Systemet är ett så kallat MIMO-system (Multi-Input Multi-Output), eftersom det har två insignaler som manipuleras (positionen på ratten och gaspedalen) och två utsignaler som regleras (hastighet och riktning).

2.1 Prediktionsmodell – modeller och identifiering

I analogin är prediktionsmodellen förarens kännedom om fordonet, det vill säga hur en justering på gaspedalen påverkar hastigheten eller hur en svängning på ratten påverkar riktningen, men också hur omgivningen påverkar fordonet. Motsvarande är prediktionsmodellen en förenklad matematisk tolkning av systemets dynamik. Modellen används för att uppskatta det framtida beteendet av det dynamiska systemet.

Ett dynamiskt system är ett system med ett minne, med andra ord ett system som påverkas av rådande och tidigare händelser. Fordonet i analogin är ett dynamiskt system, det vill säga fordonets projicerade riktning och hastighet påverkas av tidigare beslut (MathWorks, 2023).

Med prediktionsmodellen förutspås systemets framtida beteende på basen av systemets tidigare tillstånd och rådande insignaler. Prediktionsmodellen är grundläggande för de beslut på styråtgärder som beräknas med prestationsindexet och optimeringen, vilka kommer att behandlas i ett senare kapitel. Prediktionsmodellen kan byggas utgående från mätdata eller med kännedom av processen, i följande stycke behandlas identifiering och därefter några olika prediktionsmodeller (Haber, et al., 2011) (Xi & Li, 2019).

2.1.1 Identifiering

System där man känner till dynamiken behöver inte identifieras, men i system där dynamiken är okänd måste den definieras och särskiljas. Med systemidentifiering försöker man skapa en lämplig matematisk modell från data, i industrin insamlas processdata med till exempel stegförsök. Genom att mäta hur systemet besvarar en stegförändring i insignalen, kan man formulera ett samband mellan in- och utsignal. Ifall det finns uppmätta störningar i systemet sökes också sambanden mellan störningen och utsignalen.

Black-box-modellering utnyttjas i system där dynamiken är okänd och då modelleras systemet utgående från mätdata. Sambandet mellan in- och utsignal mäts med hjälp av

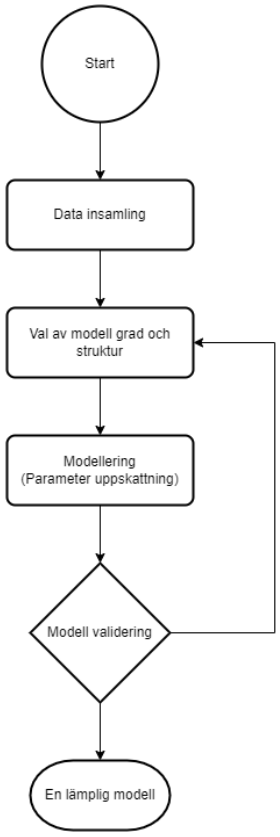
empiriska metoder. Ifall kännedomen av systemet är tillräckligt kan systemet modelleras med balansekvationer och konstitutiva relationer, en sådan modelleringsmetod kallas white-box-modellering. En blandning av de två metoderna kallas gray-box-modellering består modellformuleringen av både fysikaliska samband och parametrar som fåtts genom empiriska metoder (Tötterman, 2019) (Zhu, 2001).

I regel känner man till den generella dynamiken av system i industrin. För att modellera dessa system används ändå ofta mätdata, eftersom modellering utgående från information av systemet kan vara resurskrävande. Det kan också vara utmanande att få en modell som motsvara det verkliga beteendet av systemet på grund av slitage och andra okända faktorer som påverkar dynamiken. Därför kan mätdata ge en bättre modell av den verkliga dynamiken av systemet.

I texter om modellprediktiv reglering brukar det reglerade systemets insignal kallas en MV (Manipulated Variable) och systemets utsignal som regleras kallas CV (Controlled Variable). Utöver insignaler och utsignaler kan det finnas uppmätta störningar som brukar kallas DV (Disturbance Variable). Vid modellering av ett system bör man välja de manipulerade, reglerade och störningsvariablerna rätt, eftersom mängden variabler påverkar beräkningsbördan men också frihethetsgraden, därför kan det löna sig att överväga vilka variabler som är väsentliga. De reglerade variablerna bestäms enligt reglerbehovet, och de manipulerade variablerna väljs enligt deras inflytande på de reglerade variablerna. Utöver uppmätta störningar räknas de insignaler som är uppmätta och de som inte används för att reglera processen som störningsvariabler. Det är också önskvärt att välja sådana in- och utsignals samband som är så gott som linjära eller möjliga att linjärisera, eftersom ickelinjära samband är svårare att modellera. Vissa olinjära samband kan linjäriseras runt en referenspunkt, men om processen åker utanför området där modellen blivit linjäriserad är modellen inte längre tillräcklig för att prediktera beteendet av systemet.

I vissa fall går det att kompensera för icke linjära samband med till exempel PID-regulatorer, då styr den manipulerade variabeln börvärdet på den PID-reglerade kretsen. Scenarion där en PID-regulator kan användas är till exempel döda zoner eller regler trösklar i ställdon (Zhu, 2001) (Völker, et al., 2007) (Darby & Nikolaou, 2012).

Ifall det finns ett samband mellan en uppmätt störningsvariabel och en reglerad variabel kan framkoppling användas för att kompensera för störningen. Det finns vissa förutsättningar för att framkoppling kan användas, till exempel måste störningen ha en långsam verkan på den reglerade variabeln, eller så måste den manipulerade variabeln ha ett större inflytande på den reglerade variabeln än störningsvariabeln. En fördel med att använda framkoppling är att den minskar på modellosäkerheten (Zhu, 2001) (Bittanti, 2019).



Figur 1: Flödesschema för identifiering

Figur 1 är exempel på ett flödesschema som visar hur modellidentifieringen kan gå till. Processen börjar med datainsamling och bearbetning, och kan utföras med till exempel stegsvar. Med den preparerade data kan en passande modellstruktur och ordning skattas. Modelleringen utförs sedan med till exempel minstakvadratmetoden som behandlas i nästa stycke. Till slut provas modellen med en annan uppsättning av data, för att försäkra att modellen passar systemet, i vissa fall kan modellen också vara överanpassad. För att hitta en lämplig modell kan det behövas några iterationer av olika struktur och grad.

2.1.2 Minstakvadratmetoden

Ett typiskt identifieringsproblem är att söka konstanta samband mellan två eller flera variabler. För att hitta dessa samband måste typen av sambandet mellan variablerna

skattas, och sedan kan den lämpligaste modellen sökas. En av de mest effektiva teknikerna för att hitta sambanden är minstakvadratanpassning som också kallas linjär regression inom statistik. Med linjär regression kan man bland annat beräkna parameter för impulssvar och olika parametriska modeller. Med linjär regression söker man de optimala parametrarna genom att minimera felkvadratsumman SSE (*Sum-of-Squares Error*) (Bittanti, 2019) (Zhu, 2001).

$$SSE = \sum_{i=0}^{k} (\hat{y}_i - y_i)^2 \tag{2-1}$$

I ekvation (2-1) är $\hat{y}$ den empiriska modellen, och y den teoretiska modellen det vill säga data. Parametrarna för den empiriska modellen beräknas genom att minimera minstakvadrat summan, det vill säga felet mellan data och responsen för den empiriska modellen där parametrarna blivit skattade. I praktiken blir problemet ett optimeringsproblem (Kurula, 2021).

För att bevisa lämpligheten av modellen kan man utföra residualanalys eller beräkna modellens förklaringsgrad. Med residualanalys bekräftas att insignalerna inte korrelerar med felet mellan den skattade responsen och den verkliga responsen. Förklaringsgraden R^2 beskriver hur väl modellen passar mätdata inom intervallet $0 \leq R^2 \leq 1$. En bra modell har en förklaringsgrad som är större än 85 %.

R^2 beräknas enligt ekvation,

$$R^2 = \frac{SST - SSE}{SST} \tag{2-2}$$

där SST är kvadratsumman av den totala variansen och beräknas med responsen y och dess medelvärde som betecknas av $\bar{y}$, N är längden på responsvektorn det vill säga antalet försök (Kurula, 2021).

$$SST = \sum_{i=0}^{N} (y_i - \bar{y})^2 \tag{2-3}$$

Förklaringsgraden beskriver inte fullständigt lämpligheten av modellen, utan endast mängden av variationen mellan den predikterade och den verkliga responsen. En hög förklaringsgrad räcker inte alltid för att avgöra om modellen är bra, modellen kan lida av överanpassning och då omfattar modellen endast processens lokala egenskaper, det vill säga modellen passar endast den data som parameterestimeringen utförts med (Tangirala, 2017) (Kurula, 2021).

2.1.3 Modeller

Prediktionsmodellerna som används i modellprediktiv reglering är alltid diskreta. Det finns olika sätt att uttrycka diskreta modeller, i denna avhandling används bakåtskiftsoperatorn q^{-1}. Prediktionsmodellen kan definieras med parametriska och icke-parametriska modeller. De icke-parametriska modellerna kan byggas direkt från mätdata och beskriver systemdynamiken med ett oändligt antal parametrar. I praktiken är det dock inte möjligt att beräkna en prediktion med en oändlig mängd punkter och därför måste modellerna förkortas till en hanterlig mängd. De parametriska modellerna beskriver prediktionen med ett mindre antal parametrar som blivit approximerade. Det finns ett starkt samband mellan de olika modellerna och det går ofta att omvandla från en modell till en annan. Modellerna kan anses vara grunden till modellprediktiv reglering, därför behandlas de i detalj i detta arbete (Tangirala, 2017).

Återkopplingen av den modellprediktivregleralgoritmen är ofta en del av prediktionsmodellen. Återkoppling innebär att mätvärdet beaktas i beräkningen. Utan mätvärdet kan störningar orsaka att prediktionen blir felaktig och de beräknade styråtgärderna blir falska, återkopplingen beskrivs ofta med uttrycket (2-4), där $\hat{y}_k$ är mätvärdet och y_k prediktionen (Xi & Li, 2019) (Rossiter, 2018).

$$d_k = \hat{y}_k - y_k \tag{2-4}$$

Det att föraren kan följa banan av fordonet och kan avläsa mätvärdet av hastigheten kan anses vara mätningar, om föraren också utnyttjar mätningarna i sina styråtgärder kan systemet anses vara en sluten krets. Med återkoppling kan man bekämpa störningar till

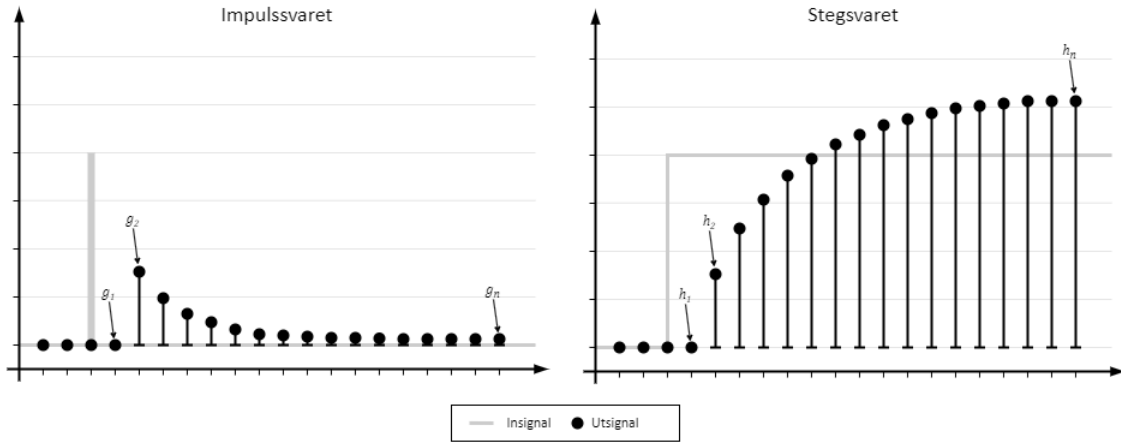
exempel kurvor på vägen som påverkar körbanan och andra störningar som påverkar hastigheten som höjdkurvor, luftmotstånd och så vidare.

Med andra ord kan återkopplingen kompensera för små fel i modelleringen eller störningar som påverkar prediktionen. Fastän prediktionsmodellen vore perfekt skulle processen inte nödvändigtvis visa ett önskvärt beteende, eftersom även små störningar kan ha en stor inverkan i ett längre lopp om styråtgärderna bestäms endast på basis av prediktionsmodellen (Bittanti, 2019).

2.1.3.1 Stegsvaret (Finite Step Response, FSR) och impulssvaret (Finite Impulse Response, FIR)

En av de första varianterna av modellprediktiv reglering var *Dynamic Matrix Control* (DMC) och *Model Algorithmic Control* (MAC). I dessa algoritmer använde man icke-parametriska modeller för att beräkna prediktionen. DMC algoritmen beräknade den framtida responsen av processen med hjälp av stegsvaret. Stegsvaret förekommer ofta i reglerteknik och används bland annat vid identifiering av processer, som diskuterats i kapitel 2.1.1. Impulssvaret som är starkt kopplat till stegssvaret används bland annat i MAC-regleralgoritm. Stegsvaret och impulssvaret omfattar teoretiskt sätt samma information, men i praktiken används stegsvaret oftare i jämförelse med impulssvaret, eftersom stegsförändringar är enklare att mäta i industriella omgivningar.

För att använda stegsvaret och impulssvaret måste superpositionsprincipen gälla för processen. Superpositionsprincipen innebär att en linjär sammansättning av insignaler ger samma linjära kombination av utsignaler. Det vill säga att prediktionen med de icke-parametriska modellerna antar att utsignalen av en linjär process är den linjära kombinationen av de tidigare insignalerna. Därför måste processen vara stabil och konvergera mot ett värde, med andra ord måste processen vara självreglerande för att de icke-parametriska modellerna ska vara tillräckliga för att beräkna en prediktion. Superpositions principen är inte giltig för icke-linjära processer (Holkar, et al., 2010) (Haber, et al., 2011) (Rossiter, 2018) (Tangirala, 2017).



Figur 2: Impulssvaret och stegsvaret

Impulssvaret beskriver systemresponsen med hjälp av en uppmätt puls av en enhetsamplitud som verkar under en tidsperiod. Med impulssvar beräknas prediktionen endast med hjälp av förfluten information på insignaler och med en term som korrigerar felet mellan målet och responsen, eller den så kallade förskjutningen (*Offset*).

$$y_{k+j} = \sum_{i=1}^{\infty} g_i \, u_{k+j-i} + d_k \approx \sum_{i=1}^{M} g_i \, u_{k+j-i} + d_k \tag{2-5}$$

$$y_{k+j} = \underbrace{\sum_{i=1}^{j} g_i u_{k+j-i}}_{framtid} + \underbrace{\sum_{i=j+1}^{M} g_i \, u_{k+j-i}}_{förflutet} + d_k \tag{2-6}$$

Som tidigare noterats går det inte i verkligheten att ta hänsyn till en oändlig mängd förflutna värden, därför väljs värdet på modellhorisonten M, så att prediktionen motsvarar den prediktionen som beaktar en oändlig mängd punkter. För att få prediktionen är G impulssvarskoefficienterna och d_k korrigeringstermen, för att få en prediktion som är enligt förväntningen. För strikt propra system gäller att $g_0 = 0$ (Rossiter, 2018).

d_k beräknas enligt ekvation (2-7) där $\hat{y}_k$ och y_k betecknar den mätvärdet respektive den beräknade responsen av systemet vid tiden k.

$$d_k = \hat{y}_k - \underbrace{\sum_{i=1}^{M} g_i \, u_{k-i}}_{y_k} \qquad (2\text{-}7)$$

Med impulssvaret utförs prediktionen med hjälp av faltningssumman, en summa av funktioner (förflutna insignaler) bildar prediktionen. På ett liknande sätt beräknas prediktionen med stegsvaret som en summa av förflutna stegförändringar i insignalen. Prediktionen med stegsvaret kan beräknas med ekvation (2-8), där h är stegsvarskoefficienterna, och Δu betecknar stegförändringen.

$$y_{k+i} = \hat{y}_k + \underbrace{\sum_{j=1}^{i} h_i \Delta u_{k+i-j}}_{framtid} + \underbrace{\sum_{j=1}^{M} h_j \Delta \, u_{k-j}}_{förflutet} + d_k \qquad (2\text{-}8)$$

Jämfört med impulssvaret ser d_k ut enligt (2-9) med stegsvaret.

$$d_k = \hat{y}_k - \underbrace{\sum_{i=1}^{M} h_i \, \Delta u_{k-i}}_{y_k} \qquad (2\text{-}9)$$

Stegsvaret kan bli härlett från impulsvaret, eftersom stegsvarets koefficienter är summan av impulssvarsets koefficienter under ett tidssteg. Sambandet kan beskrivas med ekvation (2-10) (Tangirala, 2017).

$$H(q^{-1}) = \frac{G(q^{-1})}{1 - q^{-1}} \qquad (2\text{-}10)$$

Som tidigare noterat kan man utnyttja mätdata direkt med de icke-parametriska modellerna. Bland annat med stegsvaret kan man använda data man fått från stegförsöket direkt i sambandet H (Kufoalor, et al., 2016).

2.1.3.2 CARIMA (Controlled Autoregressive Integrating Moving Average)

Överföringsfunktioner kan även användas för att uppskatta den framtida system responsen, en form av överföringsfunktion som används är bland annat CARIMA modellen, i statistik kallas modellen ARIMAX (Auto-Regressive Integrated Moving average). Överföringsfunktioner är parametriska modeller där värdet på prediktionen är en linjär kombination av föregående in- och utsignaler, och modellerna byggs upp av

en mindre mängd parametrar jämfört med de icke-parametriska modellerna. Ändå finns det ett starkt samband mellan överföringsfunktioner och stegsvaret som kan beskrivas med uttrycket (2-11) (Rossiter, 2018).

$$H(q^{-1})\Delta u_k = \frac{A(q^{-1})}{B(q^{-1})}\Delta u_k \tag{2-11}$$

I CARIMA modellen representeras processen och störningen med varsin pulsöverföringsfunktion för att få en enligt väntevärdes riktig prediktion vid stationärtillståndet. Till skillnad från stegsvaret och impulssvaret kan CARIMA modellen hantera processer utan stationärtillstånd. I pulsöverföringsfunktionen används bakåtskiftoperatorn q^{-1}, som förskjuter en signal ett tidssteg bakåt (Clarke, 1988) (Xi & Li, 2019).

$$A(q^{-1})y_k = B(q^{-1})u_k + \frac{T(q^{-1})}{\Delta}\zeta_k \tag{2-12}$$

Funktioner A, B och T är polynom av skiftoperatorn q, som blivit estimerad med hjälp av någon identifieringsalgoritm som till exempel minstakvadratmetoden som behandlas i kapitel 2.1.2.

$$\begin{aligned} A(q^{-1}) &= 1 + a_1 q^{-1} + \cdots + a_{n_a} q^{-n_a} \\ B(q^{-1}) &= b_0 + b_1 q^{-1} + \cdots + b_{n_b} q^{-n_b} \\ T(q^{-1}) &= 1 + t_1 q^{-1} + \cdots + t_{n_t} q^{-n_t} \end{aligned} \tag{2-13}$$

$A(q^{-1})$ representerar systemets allmänna dynamik och är monist, det vill säga högstagraden i polynomet är lika med ett (den första koefficienten). Polynomet $B(q^{-1})$ är sambandet mellan in- och utsignalen, för strikt propra system gäller att $b_0 = 0$. I ett strikt propert system finns det fler termer i nämnaren jämfört med täljaren ($n_a \geq n_b$), om $n_a = n_b$ är systemet endast propert. Om $b_0 \neq 0$ påverkas utsignal direkt av insignalen (*direct feedthrough*), vilket är ovanligt i praktiken. De första termerna i polynomet $B(q^{-1})$ kan representera dötiden av processen. Polynomet $T(q^{-1})$ är det glidande medelvärdet för störningen, som i regel uppfattas som en designparameter, eftersom $\frac{1}{T(q^{-1})}$ fungerar som ett låg-passfilter som kan användas för att filtrera brus som verkar i hög frekvens.

Helheten $\frac{T(q^{-1})}{\Delta}\zeta_k$ är störningen som påverkar systemet. Termen ζ_k är omätbar stokastiskt icke-korrelerande brus med noll i medelvärde, det vill säga vitt brus (Rossiter, 2018) (Clarke, 1988) (Clarke & Mohtadi, 1989) (Xi & Li, 2019) (Haber, et al., 2011) (Häggblom, 2015) (Tangirala, 2017).

I ekvation (2-12) används absoluta värden på in- och utsignalen, och på grund av parameterosäkerheten kan prediktionen förskjutas. För att förebygga en bristande prediktion kan man utnyttja differensvariabler mellan nutida värdet och det föregående värdet. Differensvariablerna fås med hjälp av operatorn $\Delta = (1 - q^{-1})$, som motsvarar exemplet i ekvation (2-14). Den relativa förändringen på styrsignalen är noll när utsignalens stationärtillstånd ligger vid börvärdet.

$$\Delta u_k = u_k - u_{k-1} \tag{2-14}$$

När styrsignalen beräknas som en förändring och med en lämplig prediktionsmodell är $\Delta u = 0$ vid stationärtillståndet, även i belastningsförhållanden som integrerande processer. För att bättre förklara hur beräkningsdynamiken fungerar med förändringar utnyttjas förarexemplet igen. Föraren är inte medveten om pedalens absoluta position utan gör små förändringar på pedalen för att nå målet, när föraren är nöjd med hastigheten är pedalpositionen konstant tills en störning sker (Clarke & Mohtadi, 1989).

Med Δ-operatorn kan ekvation (2-12) skrivas som:

$$A(q^{-1})\Delta y_k = B(q^{-1})\Delta u_k + T(q^{-1})\zeta_k \tag{2-15}$$

För att förenkla beräkningarna skapas ett nytt polynom, $D(q^{-1})$ som är härlett ifrån $A(q^{-1})\Delta$.

$$\begin{aligned} D(q^{-1}) &= d_0(q^{-1}) + d_1(q^{-1}) + d_2(q^{-2}) + \cdots + d_{n_a}(q^{-n_a}) \\ &= (1-0) + (1-a_1)(q^{-1}) + (a_1 - a_2)(q^{-2}) + \cdots \\ &+ \left(a_{n_a-1} - a_{n_a}\right)(q^{-n_a+1}) + (-a_{n_a})(q^{-n_a}) \end{aligned} \tag{2-16}$$

Prediktionen för ett tidssteg framåt med CARIMA modellen ser ut enligt ekvation (2-17). Om antagandet är att modellen ger en väntevärdes riktig prediktion är T lika med 1 och

eftersom väntevärdet på ζ_k är lika med noll, kan störningstermen negligeras (Rossiter, 2018).

$$y_{k+N} + d_1 y_{k+N-1} + d_2 y_{k+N-2} + \cdots + d_{n_a} y_{k-n_a+N-1} = b_1 \Delta u_{k+N-1} + b_2 \Delta u_{k+N-2} + \cdots b_{n_b} \Delta u_{k+N-n_b} \quad (2\text{-}17)$$

Enstegsprediktioner kan beräknas för varje tidssteg iterativt med hjälp av substituering fram till prediktionshorisonten N. För en längre prediktion blir beräkningsbördan stor. Beräkningen kan illustreras bäst med Toeplitz- och Hankel-matriser, eftersom beräkningen för prediktionen får en triangelstruktur (Rossiter, 2018) (Weisstein, 2023) (Weisstein, 2023).

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & & 0 & & 0 \\ d_1 & 1 & 0 & \cdots & 0 & \cdots & 0 \\ d_2 & d_1 & 1 & & 0 & & 0 \\ & \vdots & & \ddots & \vdots & & \vdots \\ d_{n_a} & d_{n_a-1} & d_{n_a-2} & \cdots & 1 & & 0 \\ & \vdots & & & & \ddots & \vdots \\ 0 & 0 & 0 & \dots & d_{n_a} & \dots & 1 \end{bmatrix}}_{framtid} \begin{bmatrix} y_{k+1} \\ y_{k+2} \\ y_{k+3} \\ \vdots \\ y_{k+N} \end{bmatrix} + \underbrace{\begin{bmatrix} d_1 & d_2 & d_3 & & d_{n_a} \\ d_2 & d_3 & d_4 & \cdots & 0 \\ d_3 & d_4 & d_5 & & 0 \\ & \vdots & & \ddots & \vdots \\ d_{n_a} & 0 & 0 & \cdots & 0 \\ & \vdots & & & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}}_{förflutet} \begin{bmatrix} y_k \\ y_{k-1} \\ y_{k-2} \\ \vdots \\ y_{k-n_a} \end{bmatrix} =$$

$$\underbrace{\begin{bmatrix} b_1 & 0 & 0 & & 0 & & 0 \\ b_2 & b_1 & 0 & \cdots & 0 & \cdots & 0 \\ b_3 & b_2 & b_1 & & 0 & & 0 \\ & \vdots & & \ddots & \vdots & & \vdots \\ b_{n_b} & b_{n_b-1} & b_{n_b-2} & \cdots & b_1 & & 0 \\ & \vdots & & & & \ddots & \vdots \\ 0 & 0 & 0 & \dots & b_{n_b} & \dots & b_1 \end{bmatrix}}_{framtid} \begin{bmatrix} \Delta u_k \\ \Delta u_{k+1} \\ \Delta u_{k+2} \\ \vdots \\ \Delta u_{k+N-1} \end{bmatrix} + \underbrace{\begin{bmatrix} b_2 & b_3 & b_4 & & b_{n_b} \\ b_3 & b_4 & b_5 & \cdots & 0 \\ b_4 & b_5 & b_6 & & 0 \\ & \vdots & & \ddots & \vdots \\ b_{n_b} & 0 & 0 & \cdots & 0 \\ & \vdots & & & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}}_{förflutet} \begin{bmatrix} \Delta u_{k-1} \\ \Delta u_{k-2} \\ \Delta u_{k-3} \\ \vdots \\ \Delta u_{k-n_b+1} \end{bmatrix}$$

Alternativt för att minska på beräkningsbördan som kommer ifrån den iterativa substitueringen kan man lösa prediktionsproblemet med Diofantiska ekvationer. Med Diophantine ekvationer söks lösningar där endast heltalslösningar är tillåtna. Ekvation (2-18) är en linjär Diophantine ekvation med två variabler (Weisstein, 2022) (Xi & Li, 2019) (Haber, et al., 2011) (Tangirala, 2017).

$$ax + by = c \tag{2-18}$$

Prediktionsmodellen med CARIMA modellen med Diophantine ekvationer beskrivs av uttryck

$$y_{k+j} = E_j B \Delta u_{k+j-1} + F_j y_k \tag{2-19}$$

där E_j och F_j löses för $j = 1, 2, \dots$. För detaljbeskrivning av beräkningsprocessen kan läsaren hänvisas till Clarke D, et.al. 1989 eller Xi & Li 2019 kapitel 2.2 (Clarke & Mohtadi, 1989) (Xi & Li, 2019).

2.1.3.3 Tillståndsmodellen

Under de senaste årtiondena har användningen av tillståndsmodellen ökat. Tillståndsmodellen skiljer sig från de andra modellerna, eftersom med tillståndsmodellen skattas de framtida utsignalerna med hjälp av tillståndsvariabler. Tillståndsvariablerna omfattar inre samband mellan tidigare och nuvarande insignaler, och därför kan man få en bättre förståelse av processen än med de traditionella in- och utsignalsförhållandena. (Xi & Li, 2019) (Tangirala, 2017).

Det kan ändå vara svårt att identifiera tillståndsvariablerna, eftersom de inte alltid är mätbara och därför måste de estimeras, eller så kan en tillståndsobservatör implementeras. Ett verktyg som kan användas är ett Kalmanfilter för att estimera tillståndsvariablerna (Garriga & Soroush, 2010).

I black-box-systemen där man måste skatta tillstånden kan de identifierade tillstånden vara betydelselösa, det vill säga de beskriver inte några inre samband. Därför kan det anses lönsamt att utvidga modellen med någon form av förhandsinformation. Dessutom är det svårare att identifiera tillståndsmodeller med endast mätdata jämfört med andra modeller, eftersom även tillstånden måste skattas vilket kräver olinjära identifieringsmetoder. Tillstånden som blivit identifierade kan vara observerbara eller icke-observerbara. Observerade tillstånd är tillstånd som blivit uppmätta eller kan bli härledda från ett tillgängligt mätvärde. Icke-observerbara tillstånd är tillstånd som inte går att mäta, men beskriver fysiska egenskaper av processen.

För att använda tillståndsmodell som prediktionsmodellen måste modellen vara identifierbara, det vill säga tillstånden måste vara observerade, eller så måste alla icke-observerade tillstånd konvergera till 0 (Tangirala, 2017) (Santoro & Odloak, 2012).

Tillståndsmodellen, uttryck (2-20) formuleras lika för SISO- och MIMO-systemen. I system som är strikt propra är matrisen $D = 0$, där D representerar insignalens direkta påverkan på utsignalen (*Direct feedthrough*). A matrisen är den enda av A, B, C och D matriserna som alltid är kvadratisk. I A-matrisen representerar de diagonala värdena polerna av processen om matrisen är triangulär och diagonal. Tillståndsvariablerna x_k beskriver systemdynamiken och kan vara abstrakta eller mätbara (*tangible*), men de måste vara tillräckliga för att beskriva den framtida responsen av systemet (Tangirala, 2017).

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k + D + d_k \end{aligned} \tag{2-20}$$

Alternativt kan modellen beskrivas med insignalsförändringar Δu, då blir modellen aningen mera komplicerad.

$$\begin{aligned} \begin{bmatrix} x_{k+1} \\ u_k \end{bmatrix} &= \begin{bmatrix} A & B \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix} + \begin{bmatrix} B \\ 1 \end{bmatrix} \Delta u_k \\ y_k &= \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix} + d_k \end{aligned} \tag{2-21}$$

N-stegsprediktionen med tillståndsformulering i matrisform beräknas enligt uttrycket

$$\begin{bmatrix} y_{k+1} \\ y_{k+2} \\ y_{k+3} \\ \vdots \\ y_{k+N} \end{bmatrix} = \begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ \vdots \\ CA^N \end{bmatrix} x_k + \begin{bmatrix} CB \\ CAB \\ CAB + CA^2B \\ \vdots \\ \sum_{i=0}^{N} CA^iB \end{bmatrix} u_{k-1} + \begin{bmatrix} CB & 0 & 0 & \cdots & 0 \\ CAB & CB & 0 & \cdots & 0 \\ CAB + CA^2B & CAB & CB & & 0 \\ \vdots & & & \ddots & \vdots \\ \sum_{i=0}^{N} CA^iB & \sum_{i=0}^{N-1} CA^iB & \sum_{i=0}^{N-2} CA^iB & \cdots & CB \end{bmatrix} \begin{bmatrix} \Delta u_{k-1} \\ \Delta u_{k-2} \\ \Delta u_{k-3} \\ \vdots \\ \Delta u_{k-N} \end{bmatrix}$$

(Rossiter, 2018).

2.2 Optimering, prestationsindex och begränsningar

En bilförare fattar beslut oavbrutet, beslut om storleken av styråtgärder på ratten och gaspedalen för att nå sitt mål. Med andra ord minimerar föraren felet mellan sina inbillade mål och den projicerade körbanan samt hastigheten. I modellprediktiv reglering utförs besluten om styråtgärder på ett liknande sätt, genom att minimera ett prestationsindex som innehåller termer som till exempel felet mellan den predikterade banan och börvärdet.

Prestationsindexet beräknas diskret för hela längden prediktionshorisonten. Optimeringen utförs för att beräkna de optimala värdena på de manipulerade variablerna, det vill säga de styrbara insignalerna. De manipulerade variablerna beräknas inte för hela prediktionen utan endast över längden av styrhorisonten, varefter förändringen i de manipulerade variablerna är noll för resten av beräkningarna som utförs över prediktionshorisonten. Detta görs för att minska på beräkningsbördan. För att referera tillbaka till analogin kan det tänkas att prediktionshorisonten är hur långt in i framtiden föraren tänker ut den trajektorian som bilen följer. Motsvarande är styrhorisonten de styråtgärder som föraren planerar att tillämpa vid följande ögonblick (Rossiter, 2018) (Xi & Li, 2019).

Optimeringsproblemet löses med en separat algoritm som ett kvadratiskt programmeringsproblem (2-22) och löses ofta med Interior-Point- eller Active-set-metoden, men problemet kan också lösas med andra optimeringsalgoritmer.

$$\min f^T x + \frac{1}{2} x^T H x \qquad (2\text{-}22)$$

Om problemet inte har begränsningar kan det lösas analytiskt, vilket är effektivt med tanke på beräkningsbördan (Clarke & Mohtadi, 1989) (Rossiter, 2018).

För detaljer om optimeringsalgoritmerna kan läsaren hänvisas till Frank Petterssons optimeringskompendium på svenska och Erik Hakalas magisterarbete på engelska.

2.2.1 Prestationsindex

Prestationsindexet eller den så kallade förlustfunktionen är ett numeriskt mått på reglerprestandan. I förlustfunktionen ingår termer som representerar börvärdet och prediktionsmodellen. Förlustfunktionen är formulerad som ett kvadratiskt mål som kan lösas med optimeringsalgoritmen. I prestationsindexet bör det ingå de delar av systemdynamiken som önskas styras. Utöver de termer som önskas minimeras ingår det vikter med vilka man kan prioritera vad som minimeras. I förarexemplet kan man anta att föraren prioriterar att inte avvika från körbanan och i stället avstår från hastighetsmålet, det vill säga vikten på den önskade körbanan är större än på hastighetsmålet. Förlustfunktionen bör vara formulerad på ett sådant sätt att dålig prestanda ger ett stort värde på förlustfunktionen.

I prestationsindexet minimerar man felet mellan börvärdet och den predikterade utsignalen samt mängden styråtgärder. Ett enkelt prestationsindex formuleras enligt

$$\min_{\Delta u} J = \sum_{j=1}^{p} e_{k+j}^{T} Q e_{k+j} + \sum_{j=0}^{m} \Delta u_{k+j}^{T} R\, \Delta u_{k+j} \qquad (2\text{-}23)$$

, där $e_{k+j} = y_{k+j} - r$ samt Q och R är straffmatriser eller de så kallade vikterna (Rossiter, 2018).

Ibland kan så kallade slackvariabler utnyttjas i optimeringsalgoritmen i samband med begränsningar för att öka genomförbarheten (*feasibility*) av problemet. En nackdel med de slackvariablerna är att de kan orsaka att MPC-algoritmen inte uppfyller de ställda begränsningarna (Santoro & Odloak, 2012).

2.2.2 Begränsningar

En fördel med modellprediktiv reglering jämfört med andra reglerstrategier är att man kan implementera begränsningar. En nackdel är att även om algoritmen är stabil utan begränsningar, är den inte nödvändigtvis stabil med dem, och begränsningarna kan

givetvis påverka lösningens optimum. Ofta satureras styrsignalen, det vill säga värdet ligger på någon begränsning, vilket kan orsaka långsam systemrespons (Rossiter, 2018).

Begränsingar kan ställas på samtliga signaler som styrs av den modellprediktiva regleralgoritmen. Villkoren representerar i allmänhet verkliga begränsningar på processutrustningen, som till exempel driftintervallet för en ventil eller olika säkerhetsbegränsningar. Ett typiskt reglerproblem som kan lösas med är ställdonens ställhastigheter och användningsområden (*actuator saturation*). Om de inte beaktas kan det orsaka sämre prestanda. På manipulerade variabler kan man till exempel begränsa storleken på enstaka styråtgärder och lägga en övre och lägre gräns för styrsignalen. Motsvarande gränser kan även ställas på reglerade variablerna (Rossiter, 2018) (Haber, et al., 2011).

Begränsningarna i modellprediktiv reglering kan vara hårda eller mjuka bivillkor. Begränsningarna hanteras av optimeringsalgoritmen och kan vara en del av förlustfunktionen. Hårda bivillkor är begränsningar som under inga omständigheter får brytas. Sådana bivillkor används i till exempel säkerhetsgränser som vätskenivåer.

Hårda begräsningar är vanliga linjära olikhetsvillkor som inte bör brytas och är ofta tidsoberoende och hanteras av optimeringsalgoritmen. Om bivillkoret bryts är lösningen betydelselös och en ny reglerstrategi måste beräknas. En lösning är otillåten om den bryter mot något av de hårda villkoren under prediktionshorisonten (Rossiter, 2018).

Gentemot de hårda begränsningarna får mjuka begränsningar brytas med en bestraffning. I regel används mjuka villkor vid gränser som man inte önskar bryta, men de får brytas under vissa omständigheter. Mjuka villkor är kvadratiska tilläggstermer i förlustfunktionen. För att undvika att villkoret bryts har termen en stor viktfaktor som gör att värdet på förlustfunktionen blir stort, det vill säga brytandet av villkoret bestraffar algoritmen (Haber, et al., 2011).

$$J_{add,\ min} = \sum_{i=1}^{P} W_{min} \max(0, y_{min} - y_{k+i|k})^{\,2}$$
$$J_{add,\ max} = \sum_{i=1}^{P} W_{max} W_{min} \min(0, y_{max} - y_{k+i|k})^{\,2^2} \quad (2\text{-}24)$$

Ekvationerna ovan är nedre och övre gränser för prediktionen formulerade som mjuka villkor. Villkoren är tilläggstermer som läggs till endast om den predikterade variabeln är större än de tidsoberoendegränserna y_{min} och y_{max} (Haber, et al., 2011) (Xi & Li, 2019).

2.3 Inställning av en modellprediktiv regulator – genomförbarhet, stabilitet och integrerande processer

2.3.1 Genomförbarhet av optimeringen

Det finns flera saker som kan påverka genomförbarheten av optimeringen i regleralgoritmen. I processer med ickelinjär dynamik kan prediktionsmodellen vara linjäriserad kring en referenspunkt, i sådana fall är det viktigt att hålla responsen inom referensområdet. Om processen faller utanför referenspunkten är modellen inte längre lika noggrann, och systemets beteende kan skilja sig från responsen som beräknats från den linjäriserade prediktionsmodellen.

Mycket hårda bivillkor kan orsaka att optimeringsproblemet inte längre är genomförbart. Begränsningar kan även orsaka stabilitetsproblem, även om processen är stabil utan begränsningar. I regel lönar det sig att undvika hårda begränsningar och använda mjuka begränsningar i stället som endast bestraffar algoritmen ifall bivillkoren bryts. Ifall hårda villkor på utsignalen används ska man idealt ha så lång prediktionshorisont att signalen konvergerar. Detta är dock inte alltid möjligt på grund av beräkningsbördan, eftersom algoritmen använder prediktionsmodellen för att granska ifall reglerstrategin bryter begräsningar. Med integrerande processer bör man undvika hårda villkor på den styrda variabeln. Integrerande processer är konstant

växande och kombinerat med en lång prediktionshorisont kan optimeringsproblemet bli olösbart (Rossiter, 2018).

2.3.2 Allmänna regler som gäller för att bygga en väl fungerande modellprediktiv algoritm.

I regel är en väl presterande regulator en lämplig medelväg mellan regulatorns aggressivitet och robusthet. Även om påståendet gäller PID regulatorer, stämmer det också överens med modellprediktiva regulatoralgoritmer. För modellprediktiva regleralgoritmer finns det ett antal parametrar som påverkar stabiliteten och aggressiviteten, och flera är specifika för vissa algoritmer, därför behandlas endast några allmänna riktlinjer för inställning av modellprediktiv regulator i följande styck (Häggblom, 2015) (Darby & Nikolaou, 2012).

Det första steget för att skapa en väl presterande regulator är att utveckla en passande prediktionsmodell för den styrda processen. Om modellen är bra är resten av inställandet enkelt. Det är viktigt att den matematiska modellen motsvarar det reglerade systemet. Redan valet av in- och utsignalssambanden kan ha en påverkan på prestationen av den modellprediktiva regleralgoritmen, riktlinjer för val av in- och utsignalssamband har behandlats i kapitel 2.1.1 (Garriga & Soroush, 2010).

Styrhorisonten, prediktionshorisonten och samplingstiden (*sampling time*) påverkar stabiliteten av regleralgoritmen. Därför bör man välja längden på prediktionshorisonten så att den täcker den huvudsakliga systemdynamiken, eller så att insvängningstiden av processen finns inom prediktionshorisonten (Garriga & Soroush, 2010).

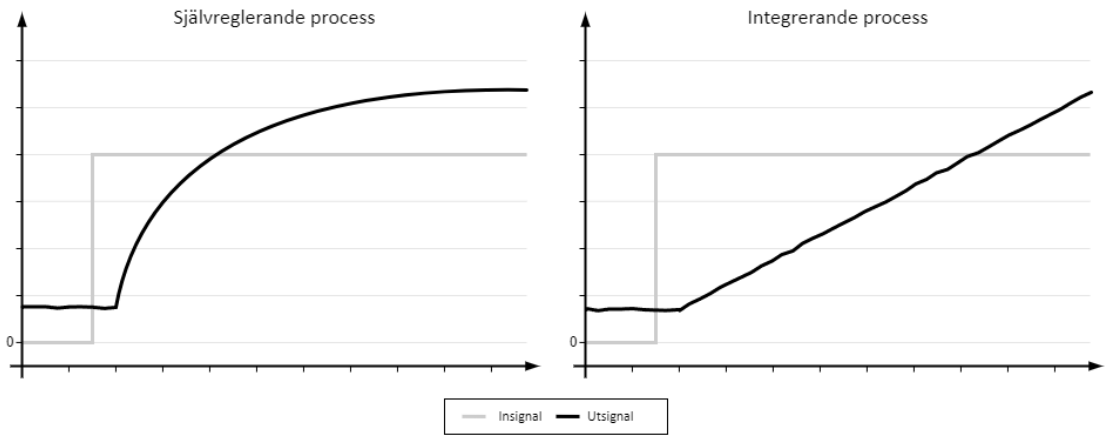
Reglerhorisonten ger regulatorn flera frihetsgrader, vilka kan behövas i processer med komplicerade in- och utsignalssamband. Då kan regulatorn planera flera regleråtgärder och tillämpa ett mera aggressivt tillvägagångssätt till kostnaden av en ökad beräkningsbörda. Inställning av en aggressiv regulator kräver en välanpassad modell som ofta skapats runt en begränsad processdriftpunkt. En mera robust regulator har en

modell som täcker den allmänna dynamiken av processen och är inte centrerad runt en processdriftpunkt och därför är en robust regulator kapabel att återhämta sig även när processen befinner sig utanför driftområdet.

Samplingstiden eller regulatorns exekverings cykel påverkar hur väl algoritmen avvisar störningar, men ett för kort samplingsintervall kan också leda till instabilitet och slitage på ställdon (Häggblom & Böling, 2013).

Med vikter på felet och styråtgärder kan man öka regulatorns stabilitet eller aggressivitet. Bland annat genom att öka på vikten på förändringarna i de manipulerade variablerna kan meningslösa justeringar uteslutas, också genom att sätta en stor matrisvikt på feltermen i förlustfunktionen kan systemets respons ökas (Rossiter, 2018) (Xi & Li, 2019).

2.3.3 Integrerande processer



Figur 3: Exempel på en självreglerande och integrerande process

Överföringsfunktionen för en integrerande process kan beskrivas med ekvation (2-25), där K är förstärkning och L är dödtiden. I en integrerande process ser responsen ut som en ramp när processen testas med en stegförändring.

$$G(s) = \frac{Ke^{-Ls}}{s} \tag{2-25}$$

Integrerande processer är icke-själv reglerande och är utan stationärtillstånd på grund av att de saknar en inre negativ återkoppling. Ett vardagligt exempel på en integrerande process kan vara en behållare med endast ett inflöde. Behållaren har inte en negativ återkoppling och nivån på vätskan i behållaren stiger obegränsat tills ett utlopp hittas. Däremot om behållaren utrustas med ett litet utlopp, skapas en negativ återkoppling i systemet. När vätskenivån stiger ökar trycket i utloppet och till slut balanseras in- och utflödet, och nivån blir stabil.

Det finns flera processer i industrin som kan vara integrerande, i regel är de nivåer på olika behållare som måste regleras, men också gastrycket, olika satssammansättningar, pH och temperaturkretsar kan vara så gott som integrerande om inte integrerande (Santoro & Odloak, 2012) (McMillan, 2023).

När integrerande processer styrs med en MPC algoritm kan stationärtillståndet förskjutas på grund av ihållande belastningsförhållanden. En förskjutning av stationärtillståndet innebär att utsignalen inte söker sig till börvärdet, och kan i stället vara över eller under det önskade värdet. Förskjutningen kan ske i samtliga av de reglerade variablerna även om endast ett in- och utsignalssamband är integrerande. På grund av förskjutningen kan processen börja oscillera vilket i sin tur sliter på processinstrument och utrustning (Gupta, 1998) (Beall, 2022).

Vissa prediktionsmodeller hanterar integrerande dynamik bättre än andra, till exempel GPC-algoritmen som i regel används med CARIMA modellen har inte problem med störningar som blir integrerade på grund av den bakom liggande integrerande systemdynamiken, eftersom integrerande verkan är implicit i prediktionsmodellen. Modellvalet i MPC har styrts mot tillståndsmodellen, eftersom den går att anpassa till samtliga processer, bland annat integrerande, och är mera mångsidig i jämförelse med de andra modellerna. Speciellt för integrerande processer är tillståndsmodellen enklare att anpassa, eftersom tillståndsvariablerna kan beskriva de integrerande sambanden. Andra prediktionsmodeller kan behöva specifika verktyg som utvecklats för att styra en

viss integrerande process (Darby & Nikolaou, 2012) (Rossiter, 2018) (Garriga & Soroush, 2010).

Vanligtvis antas prediktionshorisonten vara tillräckligt lång för att observera början av ett stationärtillstånd. Med integrerande processer är det svårt att bestämma längden på horisonten, eftersom responsen inte konvergerar, även om insignalerna är konstanta.

För att styra processer med integrerande dynamik har det utvecklats flera metoder, i det följande stycket presenteras några valda metoder. En av metoderna tillämpas i den experimentella delen av arbetet.

Gupta, Y.P. 1998 presenterade en metod som enligt texten förenklar DMC regulatorn till en PI-regulator för att styra integrerande processer. Metoden eliminerar förskjutningen i stationärtillståndet i processen, utan att påverka optimeringsalgoritmen och förmågan att hantera villkor. En nackdel med metoden är att regulatorn blir mera känslig för brus, men bruset kan filtreras genom att använda medelvärdet av utsignalen i stället för det verkliga värdet av utsignalen (Gupta, 1998).

Dougherty D. och Cooper D. forskade i påverkan av åtgärdsdämpandefaktorn eller styråtgärdsvikten i reglerlagen. Åtgärdsdämpandefaktorn (*move suppression*) existerar för att dämpa aggressiviteten av styråtgärderna och öka stabiliteten av regulatoralgoritmen. I deras forskning använde de Guptas härledda ekvationer för integrerande processer. Till skillnad från Guptas forskning skapade Dougherty med flera också inställningslagar för integrerande processer som gäller DMC regleralgoritmen. Inställningsrekommendationerna beräknades med hjälp av integrerande-med-dödtid-modellen modellen, som bland annat används för inställning av PI och PID-regulatorer (Dougherty & Cooper, 2003).

Inställningsrekommendationerna för åtgärdsdämpandefaktorn prövades på två olika processer, varav båda visade önskvärd prestation. Regulatorn med en för liten dämpningsfaktor visade processen ett oscillerande beteende, medan en stor dämpningsfaktor orsakade långsam respons.

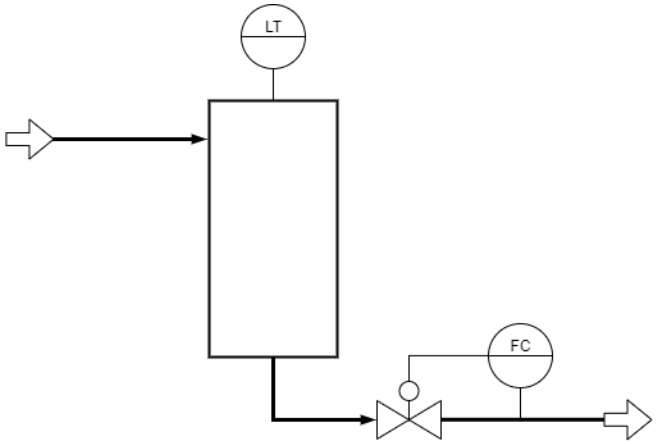
Alternativt kan integrerande processer styras med kaskadreglering där algoritmen högre i hierarkin förser den lägre algoritmen med åtkomliga stationärtillstånd. En sådan metod beskrivs bland annat av Sorensen och Cutler, där börvärdet för DMC algoritmen styrs av en skild linjärprogrammeringsalgoritm som implementerats för ett ekonomiskt syfte. Lee och Xiao utvecklade en två-stegs reglermetod för integrerande processer. För integrerande system måste utsignalen bli oförändrad, eftersom det allmänna antagandet är att utsignalen blir konstant efter ett antal tidssteg. Modellen för metoden utgår ifrån att stegförändringarna orsakar att utsignalen blir en ramp från och med ett visst antal tidssteg. Ytterligare har stegförändringarna i insignalen hastighetsvillkor som styrs av den linjära programmeringsalgoritmen. Villkoret styr värdet på insignalen, eftersom värdet på insignalen i regel är av ekonomiskt intresse. För integrerande processer behövs det två villkor jämfört med de icke integrerande processerna (Sorensen & Cutler, 1998) (Lee & Xiao, 2000).

3 Experimentella delen

I den experimentella delen av arbetet skapas en DMC algoritm som sedan jämförs med samma algoritm där prediktionsekvationerna blivit modifierade för att hantera integrerande processer. Experimentella delen är uppdelad enligt följande, först härleds simuleringsmodellen, sedan skapas DMC algoritmen för att sedan redigera prediktionsekvationerna för integrerande processer. Till slut skapas prediktionsmodellen genom White-box metoden utgående från simuleringsmodellen. Resultaten diskuteras i kapitel 4.

3.1 Simuleringsmodellen

Simuleringsmodellen för testandet av DMC-algoritmen är en behållare med ett inflöde och ett utflöde. Inflödet är oreglerat och slumpmässigt inom ett intervall medan nivån på behållaren regleras med hjälp av utflödet. Simuleringsmodellen skapas i simuleringsverktyget Simulink.



Figur 4: Grunden till simuleringsmodellen

I modellering av systemet utnyttjas balansprincipen och konstant densitet antas.

$$inström = lagring + utström$$

$$\dot{V}_{in} = \frac{dV}{dt} + \dot{V}_{ut} \tag{3-26}$$

Där $\dot{V}$ betecknar volymflöden och V volymen av behållaren.

I simuleringen är nivån eller höjden av vätskan z av intresse och då kan ekvationen ovan skrivas om enligt ekvation (3-27).

$$A\frac{\mathrm{d}z}{\mathrm{d}t} = \dot{V}_{in} - \dot{V}_{ut} \tag{3-27}$$

Storleken av utflödet beräknas med hjälp av ekvationen (3-28) härledd från Bernoullis ekvation,

$$\dot{m}_1\left(g \cdot z_1 + \frac{1}{2}\cdot v_1^2\right) + p_1 \cdot \dot{V}_1 + P_{pump} = \dot{m}_2\left(g \cdot z_2 + \frac{1}{2}\cdot v_1^2\right) + p_2 \cdot \dot{V}_2 + P_{förlust} \tag{3-28}$$

eftersom $\dot{V}_{ut}$ är produkten av arean av röret $A_{rör}$ och hastigheten v_{ut} med vilken vätskan lämnar behållaren. Variablerna med indexet ett beskriver tillstånden i behållaren medan variablerna med indexet två beskriver variablerna vid utloppet.

$$\dot{V}_{ut} = A_{rör} \cdot v_{ut} \tag{3-29}$$

Alternativt kan hastigheten beräknas med ekvation (3-30) som är ekvationen för flödet ur en öppning, med elementmotståndstalet C som är beroende av formen av utloppet.

$$v = \frac{C \cdot \sqrt{2 \cdot g \cdot \mathrm{z}}}{A_{öppning}} \tag{3-30}$$

Ekvationen är dock olämplig i det här fallet, eftersom öppningen övergår till ett rör. Med ekvation (3-30) beräknas hastigheten en bit bort från öppningen i en lägre trycks omgivning. I röret minskar trycket men inte hastigheten på grund av massbalans.

I Bernoullis lag ingår termer som representerar olika energier som förekommer i ett rörsystem. Systemet som simuleras har inte någon tryckförändring som är beroende av pumpeffekt. Nollpunkten för vätskenivån z är också höjden där röret befinner sig, det innebär att $z_2 = 0$. Vätskans hastighet i behållaren v_1 kan också anses vara lika med noll. Trycket som verkar på behållaren och trycket på andra sidan av utloppet vid ändan av röret är samma ($p_1 = p_2$), det vill säga de termerna kan negligeras. Av de termer som finns kvar kan det konstateras att den potentiella energin övergår till kinetisk energi.

$$m \cdot g \cdot z_1 = \frac{1}{2} \cdot m \cdot v_2^2 + P_{förlust} \quad (3\text{-}31)$$

$$v_2 = \sqrt{\frac{2(m \cdot g \cdot z_1 - P_{förlust})}{m}} \quad (3\text{-}32)$$

Vid behållarens utlopp sker det en tryckförlust som orsakar att all den potentiella energin inte övergår till kinetisk energi. Tryckförlusten som orsakas av förändringen i area mellan behållaren och röret, är beroende av hastigheten med vilken vätskan lämnar behållaren och Reynolds tal (Re). För att undvika de besvärliga beräkningarna och eftersom det här arbetet inte är en avhandling i flödesdynamik, förenklas beräkningen genom att ignorera faktumet att utloppets friktionsförlustfaktor ζ' kan vara flödesberoende. Förlustfaktorn blir flödesoberoende när flödet är turbulent och $Re > 10^5$.

När faktorn ζ' är oberoende har den värdet 0,5 för ett utlopp med skarpa kanter. Effektförlusten genom utloppet är dock hastighetsberoende.

$$P_{förlust} = \frac{1}{2} \cdot m \cdot v^2 \cdot (\zeta') \quad (3\text{-}33)$$

För att beräkna effektförlusten i utloppet måste ekvation (3-32) skrivas som ekvation (3-34).

$$v_2 = v_{ut} = \sqrt{\frac{2(g \cdot z_1)}{1 + \zeta'}} \quad (3\text{-}34)$$

När hastigheten fått sin ekvation kan ekvation (3-27) slutligen skrivas om, observera att $z = z_1$. För detaljbeskrivning av beräkningarna se bilaga 1 (Zevenhoven, 2013).

$$A\frac{\mathrm{d}z}{\mathrm{d}t} = \dot{V}_{in} - A_{rör} \cdot \sqrt{\frac{2 \cdot g}{1 + \zeta'}} \cdot \sqrt{z} \quad (3\text{-}35)$$

Differentialekvationen (3-35) är den matematiska modellen för behållarsystemet, med vilken simuleringsmodellen kan byggas.

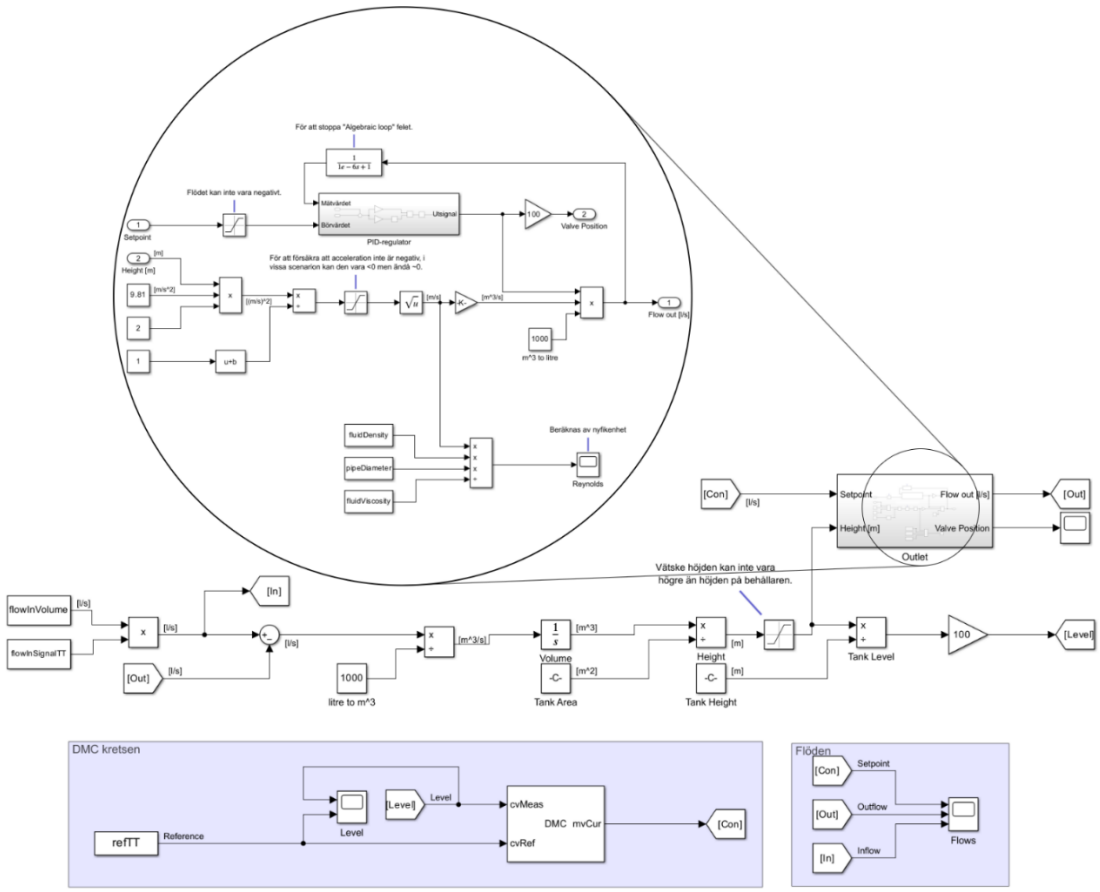
Vätskan i systemet antas vara vatten, vätskans egenskaper och andra parametrar kan avläsas från Tabell 1. Simuleringsscenariot önskas ha relativt snabb dynamik, därför

väljs behållarstorleken så att den var relativt liten, dessutom bestäms att behållaren ska ha en höjd till bredd relation på 2:1. I och med att behållaren är så liten väljs också storleken på inflödet att vara maximalt åtta liter i minuten. För att kunna balansera inflödet och minska på nivån även då inflödet är maximalt väljs utflödet att vara tolv liter i minuten. För rörstorlek ansågs en tum vara en passlig diameter. Värden på de olika parametrarna matades in ett skilt Matlab kodskript för att enkelt kunna ändra på de valda parametrarna. För simuleringsskriptet se bilaga 8 och för en skärmbild av simuleringsmodellen se Figur 5.

Tabell 1: Simuleringsparametrar (Engineering Toolbox)

Vätskan		
Densitet	ρ	$997{,}19\ \mathrm{kg/m^3}$
Viskositet	η	$1{,}0016\ \mathrm{Pa \cdot s}$
Behållaren		
Volym	V	$48\ \mathrm{l}$
Area	A	$7{,}68 \cdot 10^{-2}\ \mathrm{m^2}$
Höjd	z_{max}	$0.63\ \mathrm{m}$
Röret		
Diameter	d	$25{,}4\ \cdot 10^{-3}\ \mathrm{m}$
Area	$A_{rör}$	$5.07 \cdot 10^{-4}\ \mathrm{m^2}$

Simuleringsmodellen har en förenklad tolkning av flödesdynamiken, och ventilen som styrs kunde även utvecklas mera, men för syftet av arbetet är modellen tillräcklig.



Figur 5: En skärmbild av simuleringsmodellen.

3.2 Byggandet av en DMC algoritm

Med DMC algoritm beräknas den framtida systemresponsen med hjälp av stegsvaret, DMC algoritmen lär också vara relativt enkel att skapa. Det som behövs för att skapa algoritmen är prediktionsekvationer, ett prestationsindex och en optimeringsalgoritm för att beräkna de optimala stegförändringarna.

För att skapa en algoritm som är möjlig att simulera i Simulink skapas ett Matlab systemobjekt. Ett systemobjekt är en programklass där man kan definiera egenskaper och metoder som skapar en algoritm. Kraven för algoritmen är att användaren kan mata in en kontinuerlig överföringsfunktion som fungerar som prediktionsmodell, längden på horisonterna, värden på vikterna i prestationsindexet och eventuella villkor. Med de inmatade värdena kan algoritmen styra simuleringsmodellen.

```
classdef (StrictDefaults) DMC < matlab.System
% En enkel DMC algoritm.
% Author: Jonathan Häggvik
% Description: DMC algoritmen är skriven som en del av min
% magisteravhandlig i Kemi- och processteknik

        properties
                ...
        end

        methods
                ...
        end
```

Egenskaper (*properties*) är variabler eller parametrar som systemobjektet utnyttjar i metoderna som blivit skapade. Metoderna är beräkningar och andra funktioner som definierar algoritmen. Parametrar som är nödvändiga i arbetets modellprediktiva algoritm är de förutnämnda parametrarna som användaren matar in, prediktionsmatriserna och de tidigare styråtgärderna.

När prediktionsmatriserna blivit byggda och en prediktion kan beräknas kan optimeringen utföras. I detta fall är reglerproblemet ett SISO system men om problemet vore ett MIMO system måste en Hessematris definieras för de olika sambanden före optimeringen kan utföras. En Hessematris kan även effektivera optimeringsberäkningarna i ett SISO-system, eftersom Hessematrisen beskriver optimeringsproblemets gradient.

3.2.1 Stegsvarets prediktionsmatriser (Bilaga 7)

Prediktionsmatriserna skapas med den inmatade prediktionsmodellen. De nödvändiga stegsvarskoefficienterna fås genom att utföra ett stegtest på prediktionsmodellen, detta görs enkelt i matlab med kommandot ”step”.

```
            ...
            % Utför stegsförsök på prediktionsmodellen och sparar resultatet.
            this.H = step(this.predModel,timeSpan); % size(modlHor, 1)
            ...
```

Med stegssvarskoefficienterna kan prediktionsmatriser beräknas för att täcka prediktionshorisonten. För att utföra en prediktion med stegsvaret utgår beräkningen från ekvation (3-36), där *M* betecknar modellhorisonten.

$$y_{k+i} = \hat{y}_k + \underbrace{\sum_{j=1}^{i} h_i \Delta u_{k+i-j}}_{framtid} + \underbrace{\sum_{j=1}^{M} (h_{j+i} - h_j) \Delta u_{k-j}}_{förflutet} \qquad (3\text{-}36)$$

I algoritmen är modellhorisonten $M = P + 1$, men i regel bör den väljas så att processen hinner nå ett stationärt tillstånd. Härledning av prediktionsekvationen kan hittas i bilaga 3 (Garriga & Soroush, 2010).

Termen (3-37) i ekvation (3-36) representerar förändringen i utsignalen på basen av föregående tillämpade förändringar i insignalen. Från termen kan också sambandet mellan impulssvaret och stegsvaret observeras, subtraktionen bildar impulssvaret g_j, och summeringen av impulssvaret bildar stegsvaret (Ogunnaike, 1986).

$$\Delta y_k = \underbrace{\sum_{j=1}^{M} (h_{j+i} - h_j) \Delta u_{k-j}}_{förflutet} \qquad (3\text{-}37)$$

$$h_{j+i} - h_j \approx 0\,, \qquad j > M$$

För att härleda hur prediktionen ser ut skrivs ekvation (3-36) ut steg för steg.

$$y_{k+1} = \hat{y}_k + h_1 \Delta u_k + |h_2 - h_1| \Delta u_{k-1} + |h_3 - h_2| \Delta u_{k-2} + \cdots + |h_{m+1} - h_m| \Delta u_{k-m}$$

$$y_{k+2} = \hat{y}_k + h_1 \Delta u_{k+1} + h_2 \Delta u_k + |h_3 - h_1| \Delta u_{k-1} + |h_4 - h_2| \Delta u_{k-2} + \cdots + |h_{n+1} - h_n| \Delta u_{k-m}$$

$$y_{k+3} = \hat{y}_k + h_1 \Delta u_{k+2} + h_2 \Delta u_{k+1} + h_3 \Delta u_{k+1} + |h_4 - h_1| \Delta u_{k-1} + |h_5 - h_2| \Delta u_{k-2} + \cdots + |h_M - h_{M-1}| \Delta u_{k-M+1}$$

För att få en effektivare beräkning i Matlab skrivs prediktionsekvationerna i matrisform, från matriserna kan man identifiera ett mönster som liknar mönstren i CARIMA prediktionsmatriserna som behandlats i kapitel 2.1.3.2.

$$\begin{bmatrix} y_{k+1} \\ y_{k+2} \\ y_{k+3} \\ \vdots \\ y_{k+N} \end{bmatrix} = \begin{bmatrix} \hat{y}_k \\ \hat{y}_k \\ \hat{y}_k \\ \vdots \\ \hat{y}_k \end{bmatrix} + \begin{bmatrix} h_1 & 0 & 0 & & 0 \\ h_2 & h_1 & 0 & \cdots & 0 \\ h_3 & h_2 & h_1 & & 0 \\ & \vdots & & \ddots & \vdots \\ h_{P-S} & h_{P-S-1} & h_{P-S-2} & \cdots & h_1 \\ & \vdots & & & \vdots \\ h_P & h_{P-1} & h_{P-2} & \cdots & h_{P-S} \end{bmatrix} \begin{bmatrix} \Delta u_k \\ \Delta u_{k+1} \\ \Delta u_{k+2} \\ \vdots \\ \Delta u_{k+S-1} \end{bmatrix}$$

$$+ \begin{bmatrix} h_2 - h_1 & h_3 - h_2 & h_4 - h_3 & \cdots & h_M - h_{M-1} \\ h_3 - h_1 & h_4 - h_2 & h_5 - h_3 & \cdots & h_M - h_{M-1} \\ h_4 - h_1 & h_5 - h_2 & h_6 - h_3 & & h_M - h_{M-1} \\ & \vdots & & \ddots & \vdots \\ h_P - h_1 & h_{P+1} - h_2 & h_{P+2} - h_3 & \cdots & h_M - h_{M-1} \\ h_{P+1} - h_1 & h_{P+2} - h_2 & h_{p+3} - h_3 & & h_M - h_{M-1} \end{bmatrix} \begin{bmatrix} \Delta u_{k-1} \\ \Delta u_{k-2} \\ \Delta u_{k-3} \\ \vdots \\ \Delta u_{k-M+1} \end{bmatrix}$$

Matrisen med stegkoefficienterna som multipliceras med de framtida styråtgärderna har ett så kallat Toeplitzmönster, och matrisen med de föregående styråtgärderna har ett mönster som motsvarar en Hankel-matris. Storleken på de olika matriserna definieras av horisonterna, matrisen med Toeplitzmönstret är av storleken $P \times S$ och kallas Ct i matlabkoden. Matrisen med det avancerade Hankelmönstret är av storleken $P \times (M - 1)$ och kallas Mh i koden. För symboliska matrisberäkningar i Matlab se bilaga 9. I matlab finns det funktioner för att skapa Toeplitz- och Hankel-matriser utan en större beräkningsbörda (Kufoalor, et al., 2016).

```
...
 % Första raden i Toeplitz-matrisen.
rowsC = [H(1), zeros(1,S-1)];
% Första kolumnen i Toeplitz-matrisen.
columnsC = H(1:P)';

this.Ct = toeplitz(columnsC,rowsC);
...


...
% Första kolumnen i Hankel-matrisen.
columnsH = H(1+1:P+1)';
% Sista raden i Hankel-matrisen.
rowsH = [H(P+1:M), ones(1, P-1)*H(M)];
Hh = hankel(columnsH,rowsH);
...
```

3.2.2 Prestationsindex (Bilaga 4)

För att optimera styråtgärderna behövs ett prestationsindex, ett typiskt index behandlades i kapitel 2.2.1, det vill säga ekvation (2-23).

$$\min_{\Delta u} J = \sum_{j=1}^{p} e_{k+j}^{T} Q e_{k+j} + \sum_{j=0}^{m} \Delta u_{k+j}^{T} R\, \Delta u_{k+j} \qquad (2\text{-}23)$$

Prestationsindexet i arbetets algoritm är av formen,

$$\min_{\Delta u} J = [E - C\Delta u]^{T}[E - C\Delta u] + [\Delta u]^{T} R[\Delta u] \qquad (3\text{-}38)$$

där

$$E = e_{k+i} = r_{k+i} - \mathbf{1}\hat{y}_k - \underbrace{\sum_{j=1}^{M} (h_{j+i} - h_j)\Delta u_{k-j}}_{förflutet}\ , \qquad i = 1, 2, \dots, P \qquad (3\text{-}39)$$

Och **1** är en vektor med ettor av längden P. I koden är **1** representerat med namnet L och Ct är matrisen med stegsvarskoefficienter i Toeplitzmönster.

Prestationsindexet skrivs på detta sätt för att lätt byta vilka prediktionsberäkningar som används, prediktionsberäkningar som passa integrerande processer behandlas senare i den experimentella delen.

I prestationsindexet ingår endast en vikt för att straffa styråtgärder, eftersom systemet som styrs är ett SISO-system. Ifall prioriteten på att minimera felet måste ökas kan vikten som bestraffar styråtgärder minskas. I MISO- och MIMO-system kan felvikten vara nödvändig för att prioritera vilken av de styrda variablerna är av högre prioritet.

```
            ...
            %% Optimering

            % kommentera ut det e som inte används.
%             e = this.getE(cvMeas,cvRef);
            e = this.getIntegrE(cvMeas,cvRef);

            mv0 = zeros(this.ctrlHor,1);

            cost = @(mv)this.CostFunction(mv,e);
            ...
```

I Matlab kan optimeringen utföras med kommandot ”quadprog” som är en del av Matlabs Optimization Toolbox-verktyg. I kommandot kan man mata in likhets- och

olikhetsvillkor samt olinjära villkor. Optimeringsalgoritmen som ska lösa optimeringsproblemet kan även specificeras. Optimeringsalgoritmen som används i algoritmen i arbetet är interior-point, eftersom den är väl anpassad till MPC-optimeringsproblemen, ett annat alternativ vore active-set metoden. För att använda ”quadprog” kommandot måste förlustfunktionen formuleras som en kvadratisk objektfunktion (Xi & Li, 2019) (Nocedal & Wright, 2006).

$$\min f^T x + \frac{1}{2} x^T H x \tag{3-40}$$

$$f^T = -2E^T \cdot C$$

$$H = 2(C^T C + R) \tag{3-41}$$

$$x = \Delta u$$

```
...
H = 2 * (this.Ct'*this.Ct + this.moveWeight);

f = - 2 * (e)' * this.Ct;
% f' * x + 1/2 * x' * H * x
[delta_u, functionValue,~, output] =
quadprog(H,f,A,b,Aeq,beq,lb,ub,[],this.quadprogOpt);
...
```

Hårda villkor som användaren matar in löses av optimeringsalgoritmen, men om villkoren är så kallade mjuka villkor måste de vara en del av prestationsindexet. I detta arbete valdes det att lämna bort de mjuka villkoren, eftersom de inte är relevanta för syftet med arbetet.

Något som bör noteras är att optimeringsalgoritmen söker efter optimala stegförändringar, det vill säga Δu. Signalen som matas ur systemblocket i simuleringen är summan av den tidigare utmatade signaler och den nya förändringen.

```
...
this.upast = this.upast + delta_u(1);
u = this.upast;
...
```

På grund av att optimeringsalgoritmen söker de optimala stegförändringarna måste de hårda villkoren på den styrda och den manipulerade variabeln formuleras på ett speciellt

sätt. Villkoren matas in som matriser i ”quadprog” kommandot, uttrycket (3-42) beskriver formatet.

$$A \cdot \Delta u_{k+i} \leq b \tag{3-42}$$

För att kunna ställa villkor på till exempel den styrda variabeln måste $A \cdot \Delta u$ vara lika med y, det vill säga att matrisen A måste bestå av stegsvarskoefficienter. Ett problem uppstår på grund av att produkten av stegsvaret och stegförändringen bildar Δy under prediktionshorisonten, eftersom begynnelsetillståndet av y inte beaktas. Algoritmen mäter värdet på den styrda variabeln vid varje exekvering och med hjälp av mätvärdet $\hat{y}_k$ kan villkoret granskas. Villkoret tar då formen (3-43), där $y_{max} - \hat{y}_k$ bildar Δy. Nedre gränsen y_{min} ställs genom att multiplicera olikhetsvillkoret med minus ett. A är den sista raden i Toeplitz-matrisen, eftersom antagandet är att processen är närmast villkoret i slutet av prediktionshorisonten (Xi & Li, 2019).

$$\begin{aligned} A \cdot \Delta u_{k+i} &\leq y_{max} - \hat{y}_k \\ -A \cdot \Delta u_{k+i} &\leq -y_{min} + \hat{y}_k \end{aligned} \tag{3-43}$$

```
...
% Övre gränsen
A(1,:) = this.Ct(this.predHor, :);

b(1,1) = this.cvConHardUB - cvMeas;
...
% Nedre gränsen
A(2,:) = -this.Ct(this.predHor, :);

b(2) = -this.cvConHardLB + cvMeas;
...
```

För att ställa villkor på den manipulerade variabeln måste summan av de tidigare styråtgärderna beaktas i villkoret. Utöver det måste det granskas att inget av de beräknade styråtgärderna bryter villkoren. Detta görs genom att A bildar en Toeplitz-matris av ettor och nollor, sedan genom att ett för ett minska på de tidigare styråtgärderna och ersätta dem med nya styråtgärder. På detta sätt blir de nya styråtgärderna begränsade.

$$A = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & 1 & & 0 \\ \vdots & & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

$$A \cdot \Delta u_{k+i} \leq u_{max} - \sum_{j=1}^{M-i} u_{k-j} \qquad (3\text{-}44)$$

$$-A \cdot \Delta u_{k+i} \leq -u_{min} + \sum_{j=1}^{M-i} u_{k-j}$$

```
...
if ~isempty(this.mvConHardLB) % Nedre gränsen
        A(3:2+this.ctrlHor, :) = -toeplitz(ones(this.ctrlHor,1),[1,
        zeros(1,this.ctrlHor-1)]);

        b(3:2+this.ctrlHor) = -(this.mvConHardLB) + this.upast;

end

if ~isempty(this.mvConHardUB) % övre gränsen
        A(3+this.ctrlHor:2+2*this.ctrlHor, :) = toeplitz(ones(this.ctrlHor,1),[1,
        zeros(1,this.ctrlHor-1)]);

        b(3+this.ctrlHor:2+2*this.ctrlHor) = this.mvConHardUB - this.upast;

end
...
```

För att Interior-point-algoritmen ska fungera effektivare, rekommenderar matlab att man definierar en övre och nedre gräns för variablerna i optimeringsproblemet, i detta fall Δu. Genom att ställa begränsningar på Δu, begränsas hur stora styråtgärder som får implementeras vilket kan vara en eftertraktad egenskap i vissa scenarion.

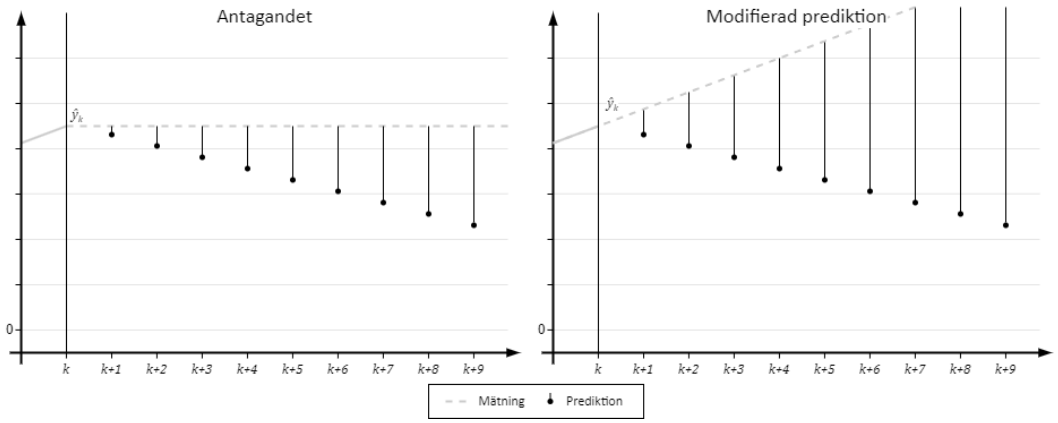
```
...
% Nedre och övre gräns för \delta u
lb = ones(this.ctrlHor,1) .* (-this.mvConHardRate);
ub = ones(this.ctrlHor,1) .* this.mvConHardRate;
...
```

3.2.3 Modifikationer för integrerande processer

De nuvarande prediktionsberäkningarna hanterar inte de kontinuerliga belastningsförhållandena vilka förekommer i det styrda behållarsystemet. I den interna

beräkningsmodellen är antagandet att mätningen $\hat{y}_k$ befinner sig i ett stationärt tillstånd och att processen söker sig till ett sådant tillstånd.

Så är inte fallet i det styrda systemet på grund av integrerande dynamik. Nivån ökar vid varje samplingstidpunkt på grund av inflödet som också skapar ett så kallat belastningsförhållande. En illustration på hur ett belastningsförhållande kan påverka en process kan hittas i Figur 6.



Figur 6: Exempel på belastningsförhållandet med integrerande process

Allmänt klarar DMC-algoritmen inte av integrerande processer, eftersom algoritmens interna beräkningar antar en självreglerande process. Med stegsvaret är antagandet att processen konvergerar.

$$\lim_{i\to\infty} h_i - h_{i-1} = 0 \qquad (3\text{-}45)$$

För att styra en integrerande process måste belastningsförhållandet läggas till i prediktionen. Detta görs genom att redigera prediktionsekvationerna enligt Gupta 1998. Förskjutningen som sker i stationärtillståndet kan beräknas ifrån regulatorns överföringsfunktion med hjälp av slutvärdessatsen. I detta fall är det dock inte nödvändigt att härleda överföringsfunktionen av regulatorn. För att undvika förskjutningen måste felet mellan den predikterade utsignalen och den verkliga utsignalen beräknas på ett annat sätt där lutningen mellan den tidigare och den nuvarande utsignalen beaktas (Gupta, 1998).

I en process med stationära tillstånd kan prediktionen beräknas enligt uttrycket,

$$y_{k+i} = y_k + \Delta y_{k+i} \tag{3-46}$$

där Δy_{k+i} är produkten av stegsvaret och de tidigare styråtgärderna som inte ännu börjat påverka processen. Tyvärr går det inte att använda samma ekvation med processer utan stationärtillstånd, eftersom den predikterade responsen är en ramp och inte begränsad. Rampen påverkas också endast av den senaste styråtgärden jämfört med de självreglerande processen där modellhorisonten beskriver antalet stegförändringar som ännu påverkar responsen. För den integrerande processen kan prediktionen beräknas med ekvation (3-47).

$$y_{k+i} = \hat{y}_k + i(\hat{y}_k - y_{k-1}), \qquad i = 1,2,\dots,N \tag{3-47}$$

Den slutliga prediktionsekvationen som används i prestationsindexet är ekvation (3-48), för detaljhärledning se bilaga 4.

$$E = \mathbf{1}e_k - i(e_{k-1} - e_k) \tag{3-48}$$

Nackdelen med prediktionsekvationen är att den är känslig för brus, eftersom prediktionen består endast av mätvärdet av processen. Enligt Gupta kan medelvärdet av mätvärdet i stället för att minska på påverkan av bruset. I praktiken vore detta ett problem men eftersom simuleringen inte har brus utnyttjas mätvärdet direkt (Dougherty & Cooper, 2003) (Gupta, 1998) (Shridhar & Cooper, 1997).

```
...
i = (1:this.predHor)';

if (cvMeas - this.cvPast == cvMeas)
    this.cvPast = cvMeas;
end

e = this.L * (cvRef - cvMeas)  + (cvMeas - this.cvPast) - i * (cvMeas - this.cvPast);
...
```

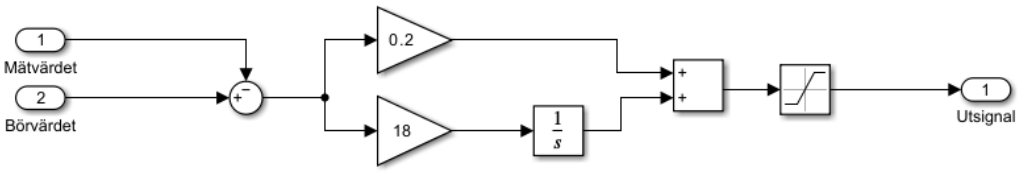
3.3 White-box identifieringstillvägagångssättet av prediktionsmodellen

När algoritmen väl blivit skriven (se bilaga 6 för hela algoritmen) kan prediktionsmodellen härledas. I detta arbete modelleras prediktionsmodellen enligt White-box metoden, beskriven i kapitel 2.1.1. Systemet kan modelleras enligt White-box metoden, eftersom dynamiken är känd och systemet är enkelt.

Den styrda variabeln är utflödet som är beroende av höjden z. Detta innebär att utflödet har ickelinjär dynamik, vilket betyder att termen måste linjäriseras innan Laplace-transformen kan utföras, detta kan göras genom att välja en lämplig referenspunkt som modellen linjäriseras kring, för beräkningen av linjäriseringen se bilaga 2.

$$A\frac{\mathrm{d}z}{\mathrm{d}t} = -A_{rör} \cdot \sqrt{\frac{2 \cdot g}{1+\zeta'}} \cdot \frac{z-\bar{z}}{2 \cdot \sqrt{\bar{z}}} + \sqrt{\bar{z}} \qquad (3\text{-}49)$$

Ett alternativ till att linjärisera utflödet är att lägga till en PI-regulator som kan användas för att linjärisera flödet, trots allt är flödet sällan oreglerat i praktiken. Användning en av en PI-regulator minskar också modellosäkerheten, då nivån befinner sig utanför den arbetspunkten där modellen blivit linjäriserad kan det verkliga flödet skilja från det predikterade flödet. Detta i sin tur kan bidra till instabilitet (Darby & Nikolaou, 2012).



Figur 7: PI-regulatorn i simuleringsmodellen.

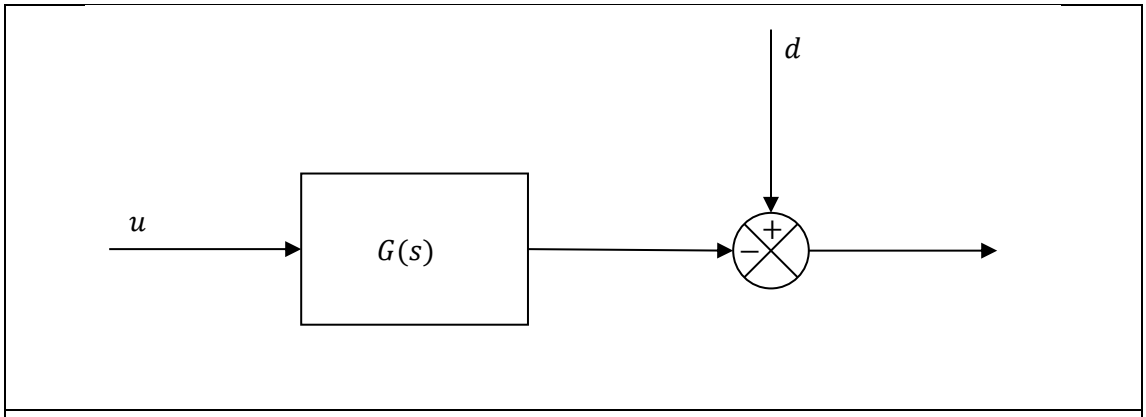
Valet att implementera en PI-regulator där börvärdet styrs av MPC-regulatorn förenklar ekvationen (3-49) till den ursprungliga ekvationen (3-27). Av praktiska skäl är sällan vätskehöjden av intresse, utan i stället hur många procent av maximala kapaciteten

behållaren innehåller. För att nivån i behållaren skall uttryckas i procent krävs det endast division med behållarvolymen.

$$A\frac{\mathrm{d}z}{\mathrm{d}t} = \dot{V}_{in} - \dot{V}_{ut} \tag{3-50}$$

$$\frac{\mathrm{d}l}{\mathrm{d}t} = \frac{\dot{V}_{in}}{V} - \frac{\dot{V}_{ut}}{V} \tag{3-51}$$

Från ekvationen kan termen för inflödet identifieras som en störning som är okänd, då är utflödet den reglerade termen.



Figur 8: Block diagram

$$\frac{\mathrm{d}l}{\mathrm{d}t} = -\frac{\dot{V}_{ut}}{V}(100\ \%) \tag{3-52}$$

Den modellprediktiva algoritmen anger börvärdet till PI-regulatorn som ligger mellan $0 - 12$ liter i minuten eller $0 - 0.2$ liter i sekunden, därför är den manipulerade variabeln u lika med $\dot{V}_{ut}$. Ekvation (3-53) är den slutliga matematiska modellen som kan transformeras till Laplace planet, för att användas som en prediktionsmodell.

$$\frac{\mathrm{d}l}{\mathrm{d}t} = -\frac{100}{V}u \tag{3-53}$$

Laplace-transformen beskrivs med uttrycket (3-54).

$$F(s) = \int_{0^-}^{\infty} f(t)e^{-s}\mathrm{d}t \tag{3-54}$$

Transformation av ekvation (3-53) när behållarvolymen är 48 liter, ger ekvation (3-55),

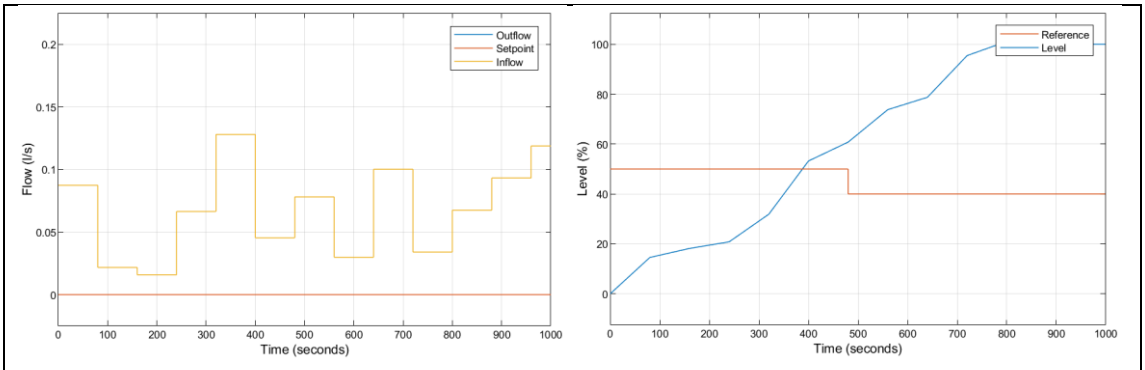
$$l = \frac{2{,}0833}{s} u \qquad (3\text{-}55)$$

som är prediktionsmodellen för systemet.

Ifall det fanns ett mätvärde på inflödet skulle det vara möjligt att framkoppla inflödet (kapitel 2.1.1). Inflödet skulle ha samma prediktionsmodell som utflödet, men skillnaden är att utflödet är beroende av nivån medan inflödet inte är det, det vill säga inflödet integreras i systemet oberoende var nivån befinner sig, medan utflödet integreras inte längre när nivån befinner sig nära noll (Nikus & Waller, 2002) (Böling, 2022).

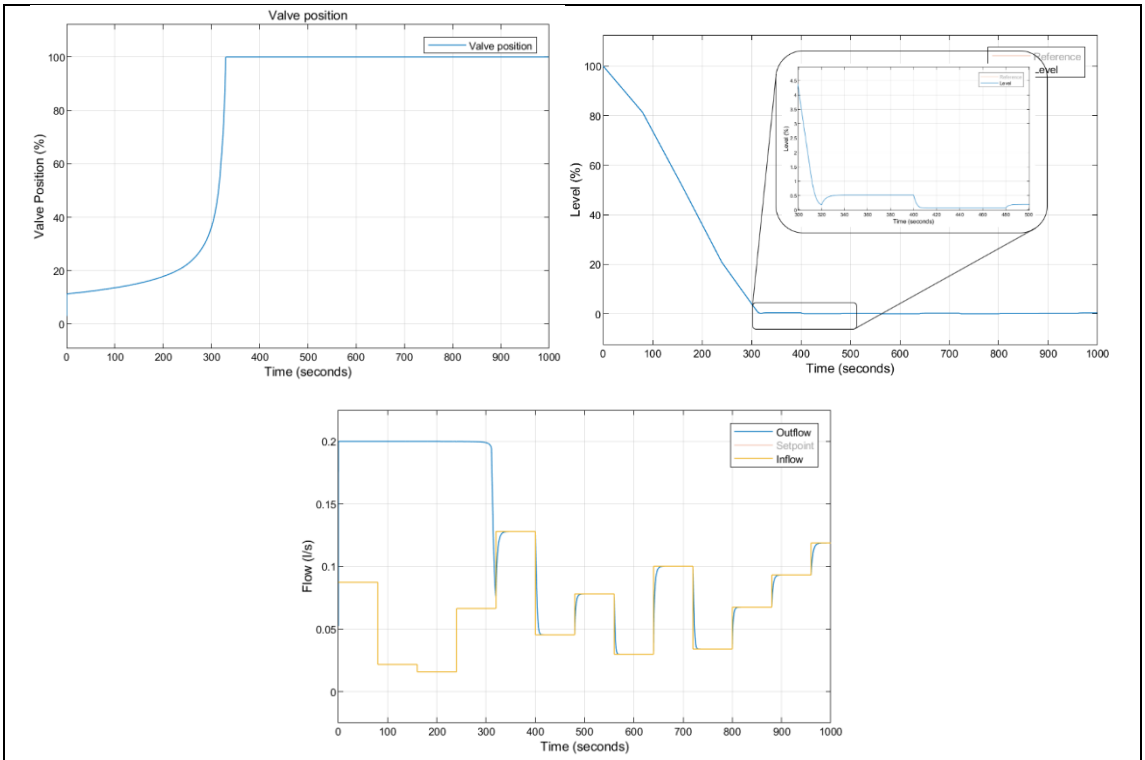
3.4 Simulering och test av algoritmen

I detta kapitel prövas den skapade simuleringsomgivningen och DMC algoritmen. I figurerna nedan är simuleringens dynamik upp ritad. Figur 9 visar hur inflödet integreras i systemet.



Figur 9: Inflödet och vätskenivån i behållaren.

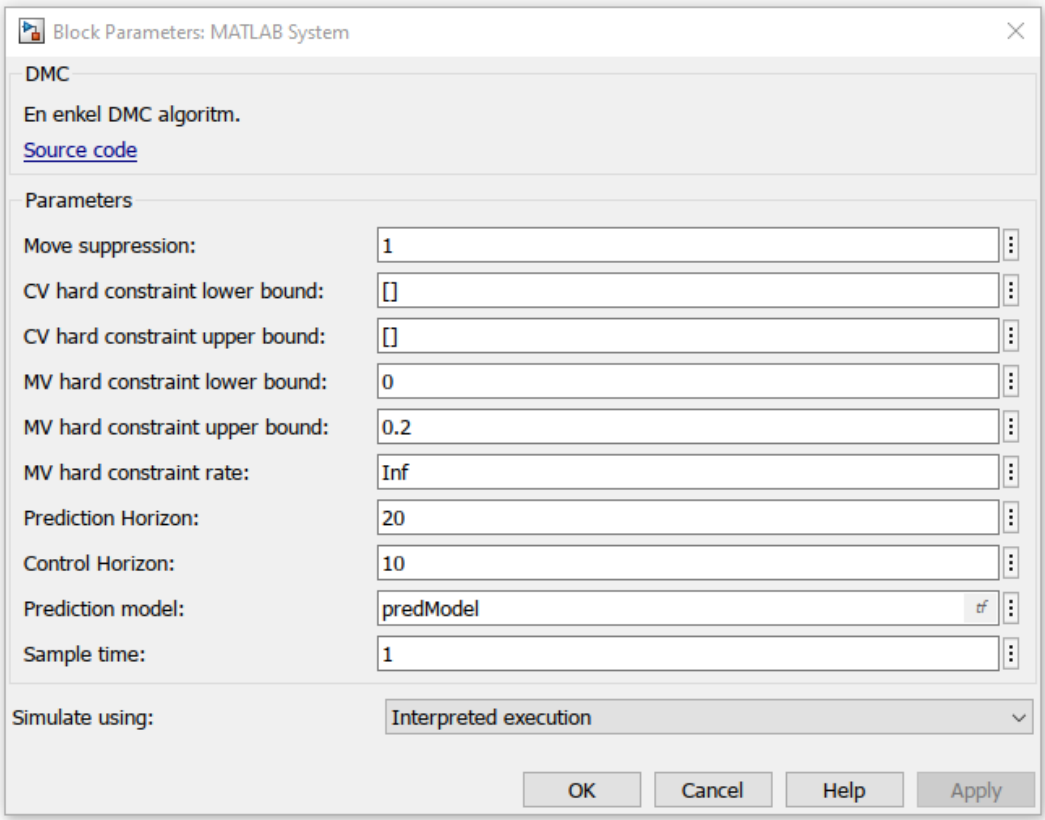
I Figur 10 har börvärdet för PI-regulatorn ändrats och ventilen öppnas för att hålla flödet vid 0.2 liter i sekunden, från grafen kan den ickelinjära dynamiken av utflödet noteras. Dynamiken är inte längre integrerande när nivån närmar sig noll, eftersom utflödet är beroende av höjden på vätskan i behållaren.



Figur 10: Utflödets verkan på vätskenivån, och ventilposition.

Parametrarna av arbetets DMC-algoritm kan ställas in i simulink tack vare att algoritmen är skriven som ett matlabsystemobjekt. Parametrarna som kan justeras är styråtgärdsvikten, samtliga villkor och horisonterna. Prediktionsmodellen definieras i Matlabs kommandotolk som en överföringsfunktion med hjälp av ”tf” kommandot. Fönstret för att ställa in parametrarna är demonstrerat i Figur 11.

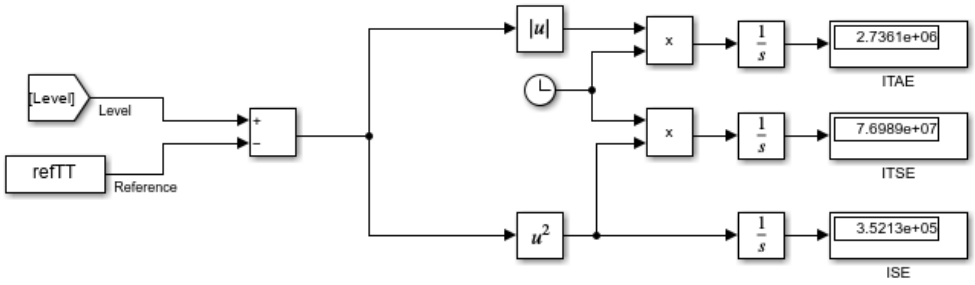
```
...
gain = 1/(tankVolume*1e3)*100
predModel = tf(-gain,[1, 0],'iodelay',0);
...
```



Figur 11: En skärmbild på DMC-systemblockets inställningar.

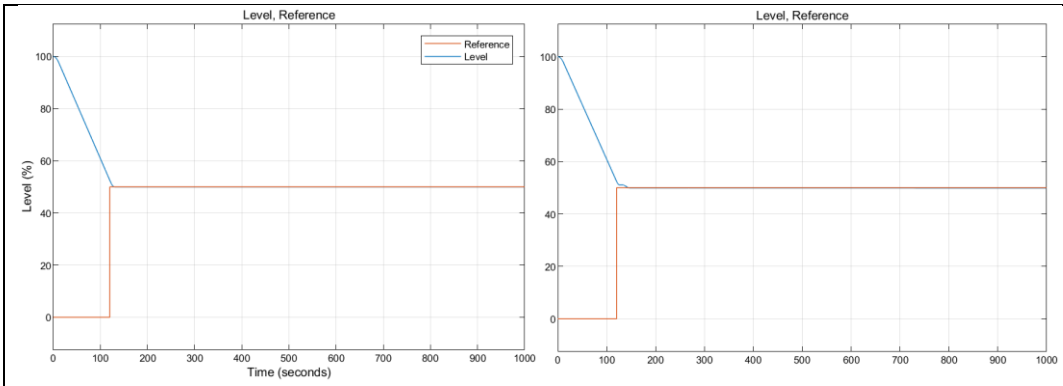
För att mäta och jämföra prestationen av de olika inställningarna som prövas på DMC-algoritmen läggs följande beräkningar till. ITAE står för det engelska *Integral of time multiplied by absolute error*, på svenska integralen av produkten av tid och det absoluta felet. Felet i detta fall är skillnaden mellan börvärdet och nivån. ITAE måttet är viktat mot slutet av försöket, och mäter hur fort algoritmen når börvärde. ITSE (*Integral of time multiplied by squared error*) är integralen av tiden multiplicerat med kvadraten av felet och är också viktat mot slutet av försöket. ITSE kan jämföras med ISE som är integralen av kvadraten av felet, för att avgöra när det största felet sker under försöket. I praktiken borde prestationsmåttet mätas från och med att en störning eller en börvärdesförändring introducerats, men eftersom scenariot är lika för varje försök och målet är inte att jämföra med en regulator utanför arbetet, mäts prestationsmåtten för hela simuleringen. Figur 12

är en skärmbild av de förutnämnda beräkningarna i simulink (Hakala, 2022) (Schultz & Rideout, 1961)



Figur 12

DMC-algoritmen med de båda metoderna för att beräkna prediktionen prövas utan inflödet och de visar önskad prestation.



Figur 13: Prestationen av DMC-algoritmen utan störning. (Vänster) Utan modifierade prediktionsekvationer. (Höger) Med modifierade prediktionsekvationer.

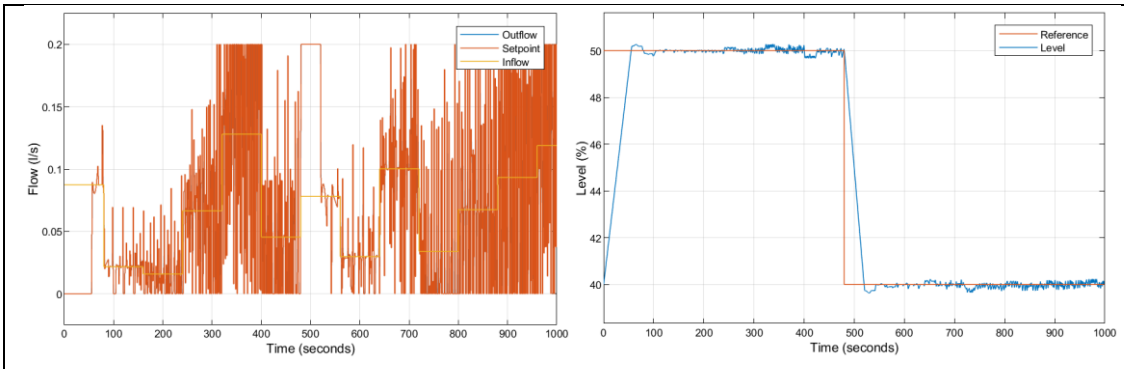
4 Resultat och diskussion

I detta kapitel ställs DMC-algoritmen in, först utan de modifierade prediktionsekvationerna och sedan med prediktionsekvationerna för processer med integrerande dynamik. Målet för inställningen av algoritmerna är ett lugnt och målmedvetet reglerande av nivån i behållaren. En sådan regulatorn kan bland annat användas i nivåreglering av en buffertbehållare.

De inställda algoritmerna jämförs numeriskt med de tidigare nämnda måtten och grafiskt. För att jämföra inställningar och prediktionsekvationer genereras en signal som varierar på inflödet och en börvärdesförändring för att se hur algoritmen presterar. Simuleringslängden är 1000 sekunder. Först prövas den vanliga DMC-algoritmen utan prediktionsekvationer som blivit anpassade för integrerande dynamik.

4.1 DMC utan integrerande prediktionsekvationer

Den vanliga DMC-algoritmen tillämpar en mycket aggressiv reglermetod för att hålla den styrda nivån vid börvärdet när algoritmen inte har blivit inställd. Det första försöket utfördes med några slumpmässiga inställningsvärden.



Figur 14: Algoritmens prestation utan inställningar.

Parameter inställningar		
Prediktionshorisonten	P	20

Styrhorisonten	S	10
Samplingstid	T_s	1
Styråtgärdsvikt	R	1
CV nedre gräns	y_{min}	$-$
CV övre gräns	y_{max}	$-$
MV nedre gräns	u_{min}	0
MV övre gräns	u_{max}	$0.2\,\mathrm{l/s} = 12\,\mathrm{l/min}$
Övre och nedre gräns för styråtgärder	$-\infty \leq \Delta u \leq \infty$	
Prestation		
ITAE	$1.460 \cdot 10^5$	
ITSE	$6.842 \cdot 10^5$	
ISE	$3.171 \cdot 10^3$	

För att få en regulator som beter sig lugnare tillämpas inställningslagar från ett antal artiklar. Shirdhar och Cooper har utvecklat inställningslagar som är byggda från en FOPDT-modell, men eftersom den nuvarande prediktionsmodellen som utnyttjas inte passar inställningslagarna så måste improviserade inställningar tillämpas.

$$\frac{y(s)}{u(s)} = \frac{K_p e^{-\theta_p s}}{\tau_p s + 1} \text{ jämfört med } \frac{y(s)}{u(s)} = \frac{K_p e^{-\theta_p s}}{s} \tag{4-56}$$

För samplingstiden rekommenderas det att välja det större värdet av en tiondel av tidskonstanten τ_p eller hälften av dödtiden θ_p. Det finns ingen approximation av dödtiden i simuleringen, men från Figur 14 kan det observeras att dödtiden är så gott som negligerbar, eftersom utflödet följer MV börvärdet nästan exakt. Tidskonstanten i prediktionsmodellen är oändlig, eftersom det är fråga om ett integrerande system. En annan begränsning är att samplingstiden i algoritmen inte kan vara mindre än ett. Ett värde mindre än ett orsakar syntaxfel i koden. På grund av dessa orsaker är samplingstiden lika med 1.

Längden på horisonterna ställs in genom att beräkna den diskreta dödtiden. Som redan konstaterats var dödtiden negligerbar och samplingstiden 1, men eftersom regleralgoritmen noterar den tillämpade styråtgärden först vid följande samplingstidpunkt är den diskreta dödtiden 1. Om man följer Cooper och Shirdhars inställningsregler blir längden på prediktionshorisonten 2, även Alhajeri och Soroush inställningsrekommendationer ger en prediktionshorisont av längden 2 och 3. Konsensusen är att prediktionshorisonten ska vara så lång att processen når ett stationärt tillstånd, därför är det svårt att uppskatta en prediktionshorisont för processer utan stationärtillstånd. I regel presterar modellprediktiva regleralgoritmer bättre med en längre horisont. För att ha någon riktlinje hur lång prediktionshorisonten måste vara, sökes svar annanstans. Genceli och Nikolaou har utvecklat en inställningsrekommendation för prediktionshorisonten som utnyttjar begränsningarna på den manipulerade variabeln (Genceli & Nikolaou, 1993) (Alhajeri & Soroush, 2020) (Manuel Lopez-Guede, et al., 2013) (Garriga & Soroush, 2010).

$$P - 1 \geq \frac{u_{max} - u_{min}}{\Delta u_{max}} \tag{4-57}$$

En annan orsak att lägga till begränsningen är att om det vore möjligt att stänga ventilen oändligt snabbt skulle det orsaka en vattenhammareffekt vilket skulle slita på rör och utrustning. En vattenhammare är effekten som kan åstadkommas i vardagen genom att ha kranen öppen och slå igen den snabbt, då detta sker smäller det till i rören. I det simulerade systemet måste ventilen stängas snabbare än 14 millisekunder för att effekten ska ske, för beräkning se bilaga 5. I bilaga 5 kan det även observeras att tack vare PI-regulatorn sker det ingen vattenhammare. Det är dock inte önskvärt att flödet tar slut tvärt och därför begränsas styrförändringarna till en liter i minuten, eller 0,01667 liter i sekunden.

Då storleken på styråtgärdsförändringarna blivit begränsad går det att föreslå en lägre gräns för prediktionshorisonten. Regulatorn måste se så långt in i framtiden att den hinner börja justera flödet i tid innan responsen når börvärdet. För att regulatorn ska gå

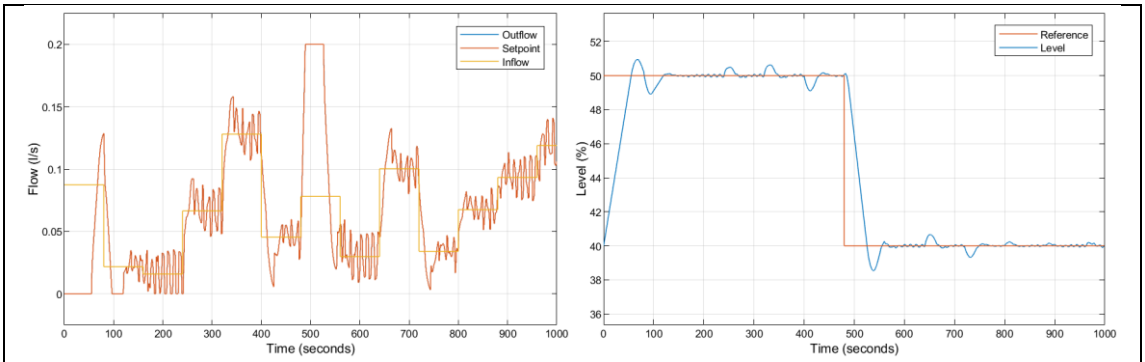
från övre gränsen till nedre gränsen, det vill säga från noll till tolv liter i minuten krävs det tolv exekveringscyklar, det betyder att prediktionshorisonten måste åtminstone vara lika med tolv, men med tanke på Genceli och Nikolaous inställningsrekommendation väljs en prediktionshorisont av längden 14.

Längden av styrhorisonten rekommenderas vara kort av Alhajeri och Soroush, också med Garriga och Soroush rekommendationer är styrhorisonten av längden ett, två eller tre, beroende på vilken tolkning som används. Enligt Shirdhar och Cooper ska man välja längden mellan ett och sex beroende på syftet. Styrhorisonten ger regulatorn flera frihetsgrader och möjligheten att tillämpa ett mera aggressivt reglersätt. Till att börja med väljs en styrhorisont som är av storleken två.

Enligt Shirdhar och Cooper ska värdet på styråtgärdsvikten beräknas med uttrycket (4-58) , då är vikten lika med ~0,087.

$$f = \begin{cases} 0\,, & S = 1 \\ \dfrac{S}{500}\left(\dfrac{3{,}5\tau_{CL}}{T_s} + 2 - \dfrac{(S-1)}{2}\right), & S > 1 \end{cases} \qquad (4\text{-}58)$$

$$R = fK_p^2$$



Figur 15: Algoritmens prestation med bättre inställningar.

Parameter inställningar		
Prediktionshorisonten	P	14

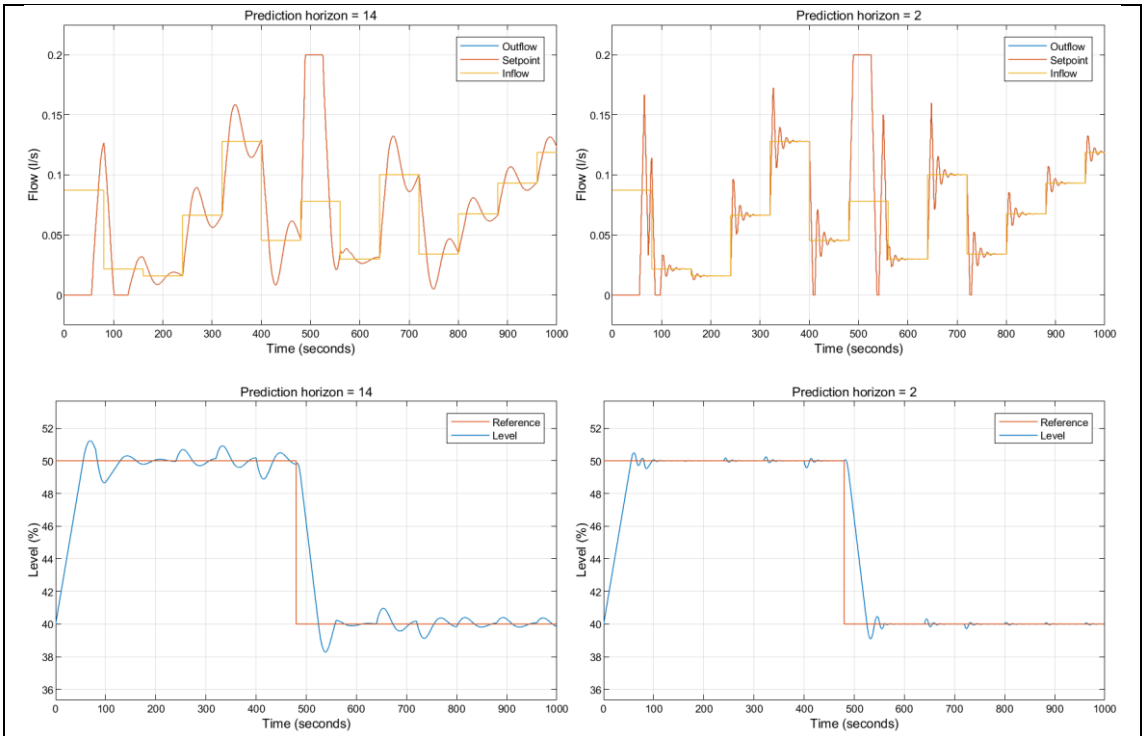
Styrhorisonten	S	2
Samplingstid	T_s	1
Styråtgärdsvikt	R	0.087
CV nedre gräns	y_{min}	−
CV övre gräns	y_{max}	−
MV nedre gräns	u_{min}	0
MV övre gräns	u_{max}	$0.2\,\mathrm{l/s} = 12\,\mathrm{l/min}$
Övre och nedre gräns för styråtgärder	$-0{,}01\overline{666} \leq \Delta u \leq 0.01\overline{666}$	
Prestation		
ITAE	$1.995 \cdot 10^5$	
ITSE	$1.008 \cdot 10^6$	
ISE	$3.846 \cdot 10^3$	

De nya parameterinställningarna resulterade i att regulatorn tillämpar mindre styråtgärder och oscillerar mindre. Däremot visar prestationsmåtten en sämre prestation vilket i och för sig är väntat, eftersom regulatorn har en begränsning på storleken av styråtgärderna vilket begränsar hur fort regulatorn kan anpassa sig till en förändring.

Prestationen av den inställda algoritmen oscillerar mer än önskat, i försök att minska på svängningarna i styrsignalen prövas olika parameterkombinationer. Alhajeri och Soroush, Garriga och Soroush samt Emhemed och Mamat har samlat ett antal uttryck för att beräkna de olika parametrarna, men på grund av den styrda processens karakteristiska beteende väljs ett värde genom test, vilket inte är en ovanlig metod i praktiken (Emhemed & Mamat, 2020).

I samband med testandet observerades att en längre prediktionshorisont resulterade i en långsammare reglerprestation. En kort styrhorisont försämrade förmågan att nå börvärdet, men ökade på stabiliteten. Som tidigare nämnts är målet för regulatorn i scenariot att ha minimal mängd styråtgärder, men ändå hålla vätskenivån nära

börvärdet. Testandet av olika parameterkombinationer producerade två alternativ, båda alternativen hade en kort styrhorisont av längden ett men olika långa prediktionshorisonter. Styråtgärdsvikten valdes att vara lika med noll enligt Emhemed och Mamats rekommendation, det vill säga när styrhorisonten är lika med ett ska vikten vara lika med noll. Ingendera av parameterkombinationerna producerar en nöjaktigt stabil prestation, men prestationerna var de bästa av de testade kombinationerna. Ifall målet vore att hålla nivån vid börvärdet, skulle alternativet med den kortare prediktionshorisonten vara bättre, men eftersom den längre prediktionshorisonten producerar färre styråtgärder anses den vara bättre i det här scenariot. I följande kapitel prövas de modifierade prediktionsberäkningarna härledda i kapitel 3.2.3.



Figur 16: Algoritmens prestation utan inställningar.

Parameter inställningar			
Prediktionshorisonten	P	14	2
Styrhorisonten	S	1	

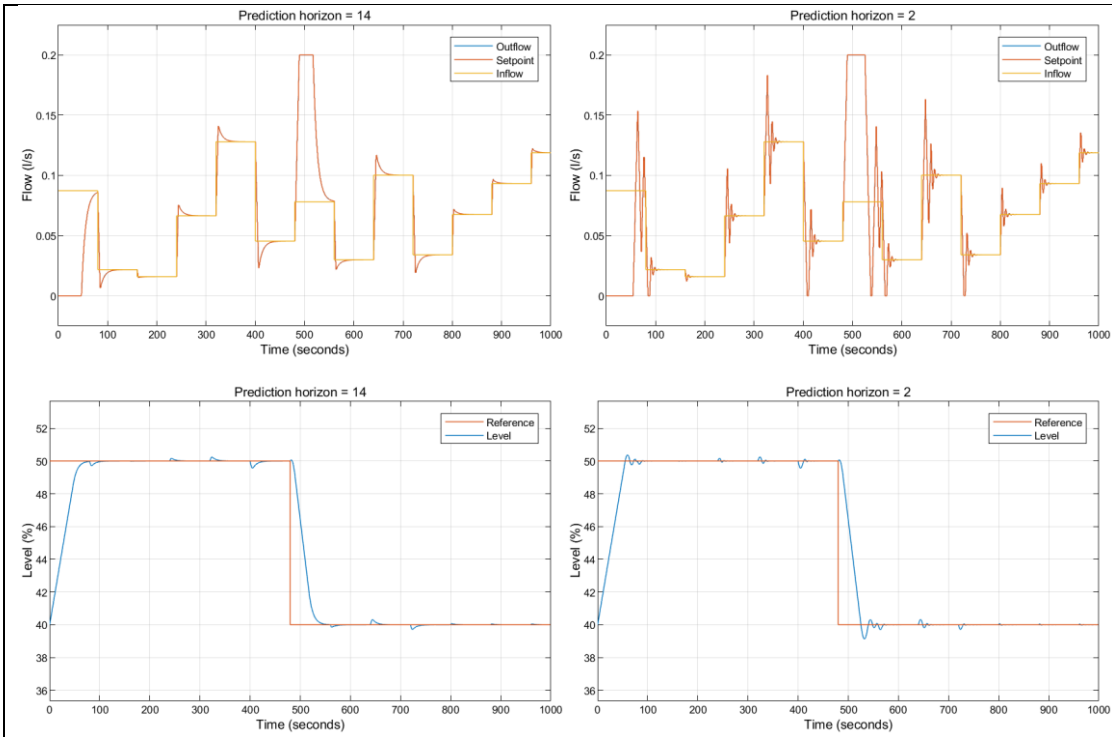
Samplingstid	T_s	1	
Styråtgärdsvikt	R	0	
CV nedre gräns	y_{min}	−	
CV övre gräns	y_{max}	−	
MV nedre gräns	u_{min}	0	
MV övre gräns	u_{max}	$0.2\ \mathrm{l/s} = 12\ \mathrm{l/min}$	
Övre och nedre gräns för styråtgärder	$-0{,}01\overline{666} \leq \Delta u \leq 0.01\overline{666}$		
Prestation	**14**	**2**	
ITAE	$2.589 \cdot 10^5$	$1.491 \cdot 10^5$	
ITSE	$9.819 \cdot 10^5$	$9.640 \cdot 10^5$	
ISE	$3.808 \cdot 10^3$	$3.739 \cdot 10^3$	

4.2 DMC med prediktionsekvationer anpassade för integrerande processer

För att använda prediktionsekvationerna passande för integrerande dynamik måste algoritmens kod redigeras genom att ta i bruk de modifierade prediktionsberäkningarna för integrerande processer som behandlats i kapitel 3.2.3. Detta görs genom att kommentera ut metoden som beräknar prediktionsberäkningar för självreglerande processer genom att lägga till en procentsymbol. Motsvarande tas de andra prediktionsberäkningarna i bruk genom att och radera procentsymbolen, andra modifikationer behövs inte, se kodsnutten nedan.

```
...
% kommentera ut det e som inte används.
e = this.getE(cvMeas,cvRef);
% e = this.getIntegrE(cvMeas,cvRef);
...
```

Med de modifierade prediktionsekvationerna prövas samma parametrar som producerat de bästa resultaten med de vanliga prediktionsekvationerna. Regulatorn med den längre prediktionshorisonten presterar mycket bra med tanke på vad målet var för regulatorn i arbetet. Om enbart prestationsmåtten jämförs presterar algoritmen med de slumpmässiga parametrarna bäst, detta beror på att algoritmen inte hade en begränsning på förändringhastigheten av den manipulerade variabeln. Orsaken till att prestationsmåtten är sämre även med algoritmen (prediktionshorisont 14 i Figur 17) som visuellt sett presterar väl, är att algoritmen söker sig långsammare till börvärdet än den första parameterkombinationen med de vanliga prediktionsekvationerna.



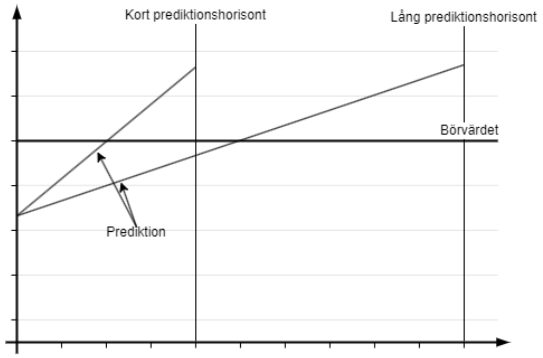
Figur 17: Prestationen med de anpassade prediktionsekvationerna.

Parameter inställningar (anpassade prediktionsekvationer)			
Prediktionshorisonten	P	14	2
Styrhorisonten	S	1	
Samplingstid	T_s	1	

Styråtgärdsvikt	R	0	
CV nedre gräns	y_{min}	−	
CV övre gräns	y_{max}	−	
MV nedre gräns	u_{min}	0	
MV övre gräns	u_{max}	$0{,}2\ \mathrm{l/s} = 12\ \mathrm{l/min}$	
Övre och nedre gräns för styråtgärder	$-0{,}01\overline{666} \leq \Delta u \leq 0.01\overline{666}$		
Prestation	$\boldsymbol{P = 14}$		$\boldsymbol{P = 2}$
ITAE	$1{,}471 \cdot 10^5$		$1{,}452 \cdot 10^5$
ITSE	$9{,}628 \cdot 10^5$		$9{,}627 \cdot 10^5$
ISE	$3{,}734 \cdot 10^3$		$3{,}733 \cdot 10^3$

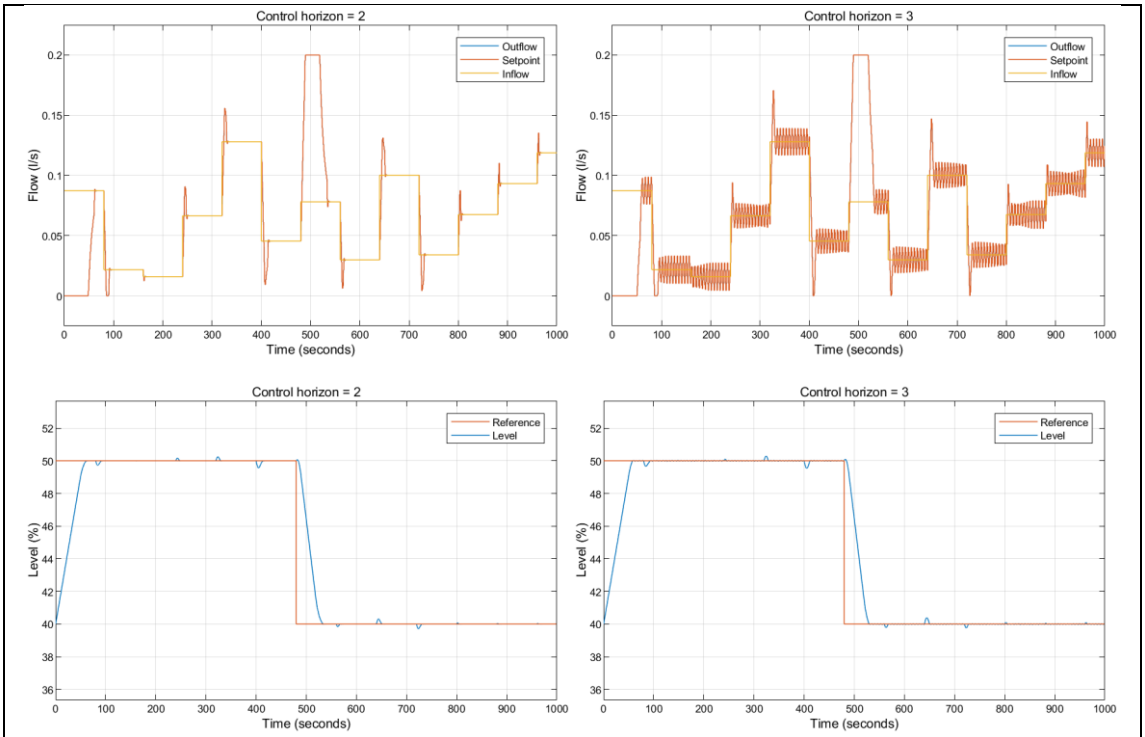
Fortsatta försök för att hitta parameterkombinationer som förbättrar prestationen av algoritmen producerade motsvarande resultat som med algoritmen utan prediktionsekvationer som blivit anpassade för integrerande processer. En längre prediktionshorisont ger en långsammare respons, och i samband med en längre prediktionshorisont kan längden på styrhorisonten ökas.

Varför responsen blir långsammare när längden på prediktionshorisonten ökas kan förklaras med den integrerande dynamiken och hur felet beräknas mellan prediktion och börvärdet, se Figur 18. Den längre prediktionshorisonten orsakar att felet blir större om aggressivare styråtgärder tillämpas.



Figur 18: Hur prediktionshorisonten påverkar de valda styråtgärder.

De fortsatta försöken producerade marginellt bättre resultat än algoritmen med anpassade prediktionsekvationer och prediktionshorisonten 14. En kortare horisont nådde börvärdet snabbare, men teoretiskt sätt kan inte prediktionshorisonten vara mycket kortare. Om förändringarna i störningen vore större skulle den orsaka svängningar i den manipulerade variabeln, eftersom regulatorn inte hinner anpassa sig till förändringen på grund av begränsningen på Δu. En styrhorisont på två gav bättre resultat på prestationsmåtten, men har större styråtgärder jämfört med när styrhorisonten var lika med ett. En längre styrhorisont större än två orsakade instabilitet i den manipulerade variabeln och styråtgärdsvikten bidrog inte till stabiliteten även om flera värden i olika storlek prövades.



Figur 19: Jämförelse av styrhorisontslängd.

Parameter inställningar (anpassade prediktionsekvationer)			
Prediktionshorisonten	P	13	
Styrhorisonten	S	2	3
Samplingstid	T_s	1	
Styråtgärdsvikt	R	178	
CV nedre gräns	y_{min}	–	
CV övre gräns	y_{max}	–	
MV nedre gräns	u_{min}	0	
MV övre gräns	u_{max}	$0{,}2$ l/s $= 12$ l/min	
Övre och nedre gräns för styråtgärder	$-0{,}01\overline{666} \leq \Delta u \leq 0.01\overline{666}$		
Prestation	$\boldsymbol{S = 2}$	$\boldsymbol{S = 3}$	
ITAE	$1{,}385 \cdot 10^5$	$1{,}460 \cdot 10^5$	

ITSE	$9{,}606 \cdot 10^5$	$9{,}852 \cdot 10^5$
ISE	$3{,}728 \cdot 10^3$	$3{,}778 \cdot 10^3$

Algoritmen med de modifierade prediktionsekvationerna presterade som önskat. Algoritmen med de vanliga prediktionsekvationerna kunde inte styra responsen till börvärdet, vilket den modifierade algoritmen kunde. Algoritmen med styrhorisonten 2 i Figur 19 presterade bäst enligt prestationsmåtten jämfört med de andra begränsade parameterkombinationerna både med och utan anpassade prediktionsekvationerna. I ett annat scenario vore den parameterkombinationen den bästa, eftersom responsen söker sig fort till börvärdet, men målet för reglerproblemet var lugna regleråtgärder därför anses algoritmen med prediktionshorisonten 14 och styrhorisonten 1 i Figur 17 bättre för att reglera bufferttanken i simuleringen.

Den skrivna DMC-algoritmen är tillgänglig i bilaga 6 och författaren av arbetet förser läsaren med simuleringsmodellen på begäran.

5 Sammanfattning

Modellprediktiv reglering är en attraktiv metod för att reglera begränsade multivariabel problem. En modellprediktiv regleralgoritm beräknar den framtida responsen med hjälp av en matematisk modell. Responsen jämförs sedan med börvärdet i en förlustfunktion som sedan minimeras genom optimering. Lösningen ger den optimala styråtgärden.

Det finns flera modeller som kan användas för att uppskatta det framtida beteendet av ett dynamiskt system. En vanlig och enkel metod är stegsvaret, som endast består av koefficienter som blivit uppmätta från ett stegförsök som utförts för att identifiera systemets dynamik. Stegsvaret är en så kallad icke-parametrisk modell. Det betyder att modellen består av en oändlig mängd parametrar, men ofta blir modellerna förkortade till ett hanterligt antal. Parametriska modeller kan härledas från de icke-parametriska modellerna, alternativt kan modellens parameter identifieras med en identifieringsmetod som till exempel minstakvadratmetoden. Till de parametriska modellerna hör bland annat tillståndsmodellen och CARIMA-modellen.

Modellerna presterar olika när de tillämpas i modellprediktiv reglering. DMC är en av de äldre tolkningarna av modellprediktiv reglering och använder stegsvaret som prediktionsmodell. En utmaning för DMC-algoritmen är integrerande eller icke-stationära störningar, eftersom beräkningarna med stegsvars modellen antar att den styrda processen söker sig till ett stationärt tillstånd.

I arbetet prövades reglering av ett behållarsystem med integrerande dynamik med hjälp av en enkel DMC-algoritm med typiska stegsvarsprediktionsekvationer och med prediktionsekvationer anpassade specifikt till integrerande processer. Olika längder på horisonter och värden på styråtgärdsvikten prövades på de två metoderna. Metoderna jämfördes visuellt och numeriskt, för att avgöra vilken metod som fungerade bättre med en integrerad störning. Metoden som inte hade anpassats för integrerande processer klarade inte av att driva nivån av den simulerade behållaren till börvärdet, eftersom inflödet av vätskan skapade en förskjutning mellan börvärdet och nivån. Den anpassade

algoritmen eliminerade förskjutningen och drabbades inte av belastningsförhållandet som orsakades av störningen. Däremot saknade simuleringsmodellen brus som ofta förekommer i industriella omgivningar, men det lär vara möjligt att använda de anpassade prediktionsekvationerna även i förhållanden med brus, om man använder medelvärdet av mätningen i stället för mätvärdet.

Arbetet skrapar endast ytan av modellprediktiv reglering, men läsaren torde få en allmän uppfattning om vad modellprediktiv reglering är.

6 Referenser

Alhajeri, M. & Soroush, M., 2020. Tuning Guidelines for Model-Predictive Control. *Industrial and Engineering Chemistry Research,* 3, 59(10), pp. 4177-4191.

Beall, J., 2022. *Loop tuning basics: Self-regulating processes.* [Online] Available at: https://www.isa.org/intech-home/2016/may-june/departments/loop-tuning-basics-self-regulating-processes

Bittanti, S., 2019. *Model Identification and Data Analysis.* Newark: John Wiley & Sons, Incorporated.

Böling, J., 2022. *E-post discussion* [Intervju] 2022.

Clarke, D. W., 1988. Application of Generalized Predictive Control to Industrial Processes. *IEEE Control Systems Magazine,* 8(2), pp. 49-55.

Clarke, D. W. & Mohtadi, C., 1989. Properties of Generalized Predictive Control. *Automatica,* 25(6), pp. 859-875.

Darby, M. L. & Nikolaou, M., 2012. MPC: Current practice and challenges. *Control Engineering Practice,* 4, 20(4), pp. 328-342.

Dougherty, D. & Cooper, D. J., 2003. Tuning guidelines of a dynamic matrix controller for integrating (non-self-regulating) processes. *Industrial and Engineering Chemistry Research,* 4, 42(8), pp. 1739-1752.

Emhemed, A. A. A. & Mamat, R., 2020. Model Predictive Control: A Summary of Industrial Challenges and Tuning Techniques. *International Journal of Mechatronics, Electrical and COmputer Technology,* 10(35), pp. 4441-4459.

Frank, P., 2016. *Optimering,* Åbo: u.n.

Garriga, J. L. & Soroush, M., 2010. Model predictive control tuning methods: A review. *Industrial and Engineering Chemistry Research,* 4, 49(8), pp. 3505-3515.

Genceli, H. & Nikolaou, M., 1993. Robust stability analysis of constrained l1-norm model predictive control. *AlChe journal,* 39(12), pp. 1954-1965.

Gupta, Y. P., 1998. Control of integrating processes using Dynamic Matrix Control. *Chemical Engineering Research and Design,* 76(4 A4), pp. 465-470.

Haber, R., Bars, R. & Schmitz, U., 2011. *Predictive Control in Process Engineering : From the Basics to the Applications.* 1 red. u.o.:John Wiley & Sons, Incorporated.

Häggblom, K.-E., 2015. *REGLERTEKNIK I: Grundkurs,* u.o.: u.n.

Häggblom, K.-E. & Böling, J., 2013. *REGLERTEKNIK II: Tillståndsmetoder,* u.o.: u.n.

Hakala, E., 2022. *Tuning strategies for control of integrating systems,* Åbo: u.n.

Holkar, K. S., Wagh, K. K. & Waghmare, L. M., 2010. An Overview of Model Predictive Control. *International Journal of Control and Automation International Journal of Control and Automation,* 3(4), pp. 47-63.

Kufoalor, D. K. M., Imsland, L. & Johansen, T. A., 2016. Efficient Implementation of Step Response Prediction Models for Embedded Model Predictive Control. *Computers & Chemical Engineering,* Volym 90, pp. 121-135.

Kurula, M., 2021. *400212.0 Försöksplanering.* u.o.:u.n.

Lee, J. H. & Xiao, J., 2000. Use of two-stage optimization in model predictive control of stable and integrating systems. *Computers & Chemical Engineering Computers and Chemical Engineering,* Volym 24.

Lundström, P., Lee, J. H., Morari, M. & Skogestad, S., 1995. Limitations of dynamic matrix control. *Computers and Chemical Engineering,* 19(4), pp. 409-421.

Manuel Lopez-Guede, J., Fernandez-Gauna, B., Graña, M. & Oterino, F., 2013. On the Influence of the Prediction Horizon in Dynamic Matrix Control. *International Journal of Control Science and Engineering,* 2013(1), pp. 22-30.

MathWorks, 2023. *System Identification Overview.* [Online] Available at: https://se.mathworks.com/help/ident/gs/about-system-identification.html [Använd 2023].

McMillan, G., 2023. *How to Better Understand Integrating and Runaway Process Dynamics.* [Online]
Available at: https://blog.isa.org/improve-industrial-automation-understand-integrating-runaway-process-dynamics
[Använd 2023].

Nebeluk, R. & Ławryńczuk, M., 2021. Tuning of multivariable model predictive control for industrial tasks. *Algorithms,* 1.14(1).

Nikus, M. & Waller, M., 2002. *Matematisk Modellering.* u.o.:u.n.

Nocedal, J. & Wright, S. J., 2006. *Numerical Optimization.* 2nd red. New York: Springer.

Ogunnaike, B. A., 1986. Dynamic Matrix Control: A Nonstochastic, Industrial Process Control Technique with Parallels in Applied Statistics. *Industrial & Engineering Chemistry Fundamentals,* 25(4), pp. 712-718.

Rossiter, J. A., 2018. *A First Course in Predictive Control, Second Edition.* 2nd red. Boca Raton: CRC Press.

Rossiter, J. A. & Kouvaritakis, B., 2001. Modelling and implicit modelling for predictive control. *International Journal of Control,* 7, 74(11), pp. 1085-1095.

Santoro, B. F. & Odloak, D., 2012. Closed-loop stable model predictive control of integrating systems with dead time. *Journal of Process Control,* 8, 22(7), pp. 1209-1218.

Schultz, W. & Rideout, V., 1961. Control System Performance Measures: Past, Present, and Future. *IRE Transactions on automatic control,* pp. 22-35.

Shridhar, R. & Cooper, D. J., 1997. A Tuning Strategy for Unconstrained SISO Model Predictive Control. *Ind. Eng. Chem. Res,* Volym 36, pp. 729-746.

Sorensen, R. & Cutler, C., 1998. LP Integrates Economics into Dynamic Matrix Control. *Hydrocarbon Processing*, 77(9).

Tangirala, A. K., 2017. *Principles of System Identification.* 1st red. Boca Raton: CRC Press.

Tötterman, S., 2019. *Aspects on Robust Control and Identification,* u.o.: u.n.

Völker, M., Sonntag, C. & Engell, S., 2007. Control of integrated processes: A case study on reactive distillation in a medium-scale pilot plant. *Control Engineering Practice*, 7, 15(7), pp. 863-881.

Weisstein, E. W., 2022. *Diophantine Equation.* [Online] Available at: https://mathworld.wolfram.com/DiophantineEquation.html [Använd 2022].

Weisstein, E. W., 2023. *Hankel Matrix.* [Online] Available at: https://mathworld.wolfram.com/HankelMatrix.html [Använd 2023].

Weisstein, E. W., 2023. *Toeplitz Matrix.* [Online] Available at: https://mathworld.wolfram.com/ToeplitzMatrix.html [Använd 2023].

Xi, Y. & Li, D., 2019. *Predictive Control : Fundamentals and Developments.* Newark, SINGAPORE: John Wiley & Sons, Incorporated.

Zevenhoven, R., 2013. *PS00CD62 Fluids, particles and CFD.* u.o.:u.n.

Zhu, Y., 2001. *Multivariable System Identification for Process Control.* Amsterdam; New York: Pergamon.

7 Bilaga 1: Detaljhärledning av simuleringsmodellen

Fortsätter från ekvation (3-31).

$$m \cdot g \cdot z_1 = \frac{1}{2} \cdot m \cdot v_2^2 + \zeta' \cdot \frac{1}{2} \cdot m \cdot v_2^2 \tag{1}$$

Flyttar bryter ut lika termer.

$$m \cdot g \cdot z_1 = \left(\frac{1}{2} \cdot m \cdot v_2^2\right)(1 + \zeta') \tag{2}$$

Dividerar med $(1 + \zeta')$.

$$\frac{1}{2} \cdot m \cdot v_2^2 = \frac{m \cdot g \cdot z_1}{1 + \zeta'} \tag{3}$$

Kan förkorta lika termer.

$$v_2^2 = \frac{2 \cdot \cancel{m} \cdot g \cdot z_1}{\cancel{m}(1 + \zeta')} \tag{4}$$

Hastigheten ur utloppet v_2 är lika med:

$$v_2 = \sqrt{\frac{2 \cdot g \cdot z_1}{1 + \zeta'}} \tag{5}$$

Balansförhållandet för simuleringsmodellen.

$$\frac{\mathrm{d}V}{\mathrm{d}t} = \dot{V}_{in} - \dot{V}_{ut} \tag{6}$$

Utflödet $\dot{V}_{ut}$ är lika med arean av röret och hastigheten av vätskan i utloppet v_2.

$$A\frac{\mathrm{d}z}{\mathrm{d}t} = \dot{V}_{in} - A_{rör} v_{ut}\,, \qquad v_{ut} = v_2 \tag{7}$$

Substituerar v_2 med (5)

$$A\frac{\mathrm{d}z}{\mathrm{d}t} = -A_{rör} \cdot \sqrt{\frac{2 \cdot g}{1 + \zeta'}} \cdot \sqrt{z} \tag{8}$$

8 Bilaga 2: Linjäisering av modellen kring en arbetspunkt

Fortsätter från förra bilagan (8) eller ekvation (3-35)

$$A\frac{\mathrm{d}z}{\mathrm{d}t} = -A_{rör} \cdot \sqrt{\frac{2 \cdot g}{1+\zeta'}} \cdot \sqrt{z} \tag{1}$$

För linjäriseringen måste en lämplig arbetspunkt $\bar{z}$ väljas. Uttrycket för linjäriseringen:

$$y = f(\bar{z}) + f'(\bar{z})(z-\bar{z}) = \sqrt{\bar{z}} + \frac{1}{2}\bar{z}^{-\frac{1}{2}}(z-\bar{z}) \tag{2}$$

Substituerar $\sqrt{z}$ med den linjäriserade uttrycket.

$$\frac{d}{dh} = -A_{rör} \cdot \sqrt{\frac{2 \cdot g}{1+\zeta'}}\left(\sqrt{\bar{z}} + \frac{1}{2}\bar{z}^{-\frac{1}{2}}(z-\bar{z})\right) \tag{3}$$

Skriver om uttrycket.

$$A\frac{\mathrm{d}z}{\mathrm{d}t} = -A_{rör} \cdot \sqrt{\frac{2 \cdot g}{1+\zeta'}}\left(\frac{z-\bar{z}}{2 \cdot \sqrt{\bar{z}}} + \sqrt{\bar{z}}\right) \tag{4}$$

9 Bilaga 3: Härledning av prediktionsekvationen

$$y_{k+i} = \underbrace{\sum_{j=1}^{i} h_j \Delta u_{k+i-j}}_{framtid} + \underbrace{\sum_{j=i+1}^{M-1} h_j \Delta u_{k+i-j}}_{förflutet} + d_k$$

$$d_k = \hat{y}_k - \sum_{j=1}^{M-1} h_j \Delta u_{k-j}$$

Substitution av d_k:

$$y_{k+i} = \underbrace{\sum_{j=1}^{i} h_j \Delta u_{k+i-j}}_{framtid} + \underbrace{\sum_{j=i+1}^{M-1} h_j \Delta u_{k+i-j}}_{förflutet} + \hat{y}_k - \sum_{j=1}^{M-1} h_j \Delta u_{k-j}$$

Om ordnande av termer:

$$y_{k+i} = \hat{y}_k + \sum_{j=1}^{i} h_j \Delta u_{k+i-j} + \sum_{j=i+1}^{M-1} h_j \Delta u_{k+i-j} - \sum_{j=1}^{M-1} h_j \Delta u_{k-j}$$

Kan skrivas om som:

$$y_{k+i} = \hat{y}_k + \underbrace{\sum_{j=1}^{i} h_i \Delta u_{k+i-j}}_{framtid} + \underbrace{\sum_{j=1}^{M-1} \left| h_{j+i} - h_j \right| \Delta u_{k-j}}_{förflutet}$$

Numeriskt bevis:

Antar att $M = 3$ och beräknar för $i = 1, 2, 3$.

$$\sum_{j=i+1}^{M-1} h_j \Delta u_{k+i-j} - \sum_{j=1}^{M-1} h_j \Delta u_{k-j}$$

$$\begin{aligned} &(i = 1 \,|h_2 \Delta u_{k-1} + h_3 \Delta u_{k-2} + h_4 \Delta u_{k-3}) \\ =\ &(i = 2 \,|h_3 \Delta u_{k-1} + h_4 \Delta u_{k-2} + h_5 \Delta u_{k-3}) \\ &(i = 3 \,|h_4 \Delta u_{k-1} + h_5 \Delta u_{k-2} + h_6 \Delta u_{k-3}) \end{aligned}$$

$$\begin{aligned} &(i = 1 \,|h_1 \Delta u_{k-1} + h_2 \Delta u_{k-2} + h_3 \Delta u_{k-3}) \\ -\ &(i = 2 \,|h_1 \Delta u_{k-1} + h_2 \Delta u_{k-2} + h_3 \Delta u_{k-3}) \\ &(i = 3 \,|h_1 \Delta u_{k-1} + h_2 \Delta u_{k-2} + h_3 \Delta u_{k-3}) \end{aligned}$$

$$(i = 1 \,|h_2 \Delta u_{k-1} + h_3 \Delta u_{k-2} + h_4 \Delta u_{k-3} - h_1 \Delta u_{k-1} - h_2 \Delta u_{k-2} - h_3 \Delta u_{k-3})$$

Flyttar på termer:

$$(i = 1 \,|h_2 \Delta u_{k-1} - h_1 \Delta u_{k-1} + h_3 \Delta u_{k-2} - h_2 \Delta u_{k-2} + h_4 \Delta u_{k-3} - h_3 \Delta u_{k-3})$$

Bryter ut likadana termer:

$$(i = 1\,|(h_2 - h_1)\Delta u_{k-1} + (h_3 - h_2)\Delta u_{k-2} + (h_4 - h_1\Delta)u_{k-3})$$

$$\sum_{j=1}^{M-1} |h_{j+1} - h_j|\Delta\, u_{k-j}$$

$$(i = 2\,|h_2\Delta u_{k-1} + h_3\Delta u_{k-2} + h_5\Delta u_{k-3} - h_1\Delta u_{k-1} - h_2\Delta u_{k-2} - h_3\Delta u_{k-3})$$
$$(i = 2\,|h_3\Delta u_{k-1} - h_1\Delta u_{k-1} + h_4\Delta u_{k-2} - h_2\Delta u_{k-2} + h_5\Delta u_{k-3} - h_3\Delta u_{k-3})$$
$$(i = 2\,|(h_3 - h_1)\Delta u_{k-1} + (h_4 - h_2)\Delta u_{k-2} + (h_5 - h_1\Delta)u_{k-3})$$

$$\sum_{j=1}^{M-1} |h_{j+2} - h_j|\Delta\, u_{k-j}$$

$$(i = 3\,|h_4\Delta u_{k-1} + h_5\Delta u_{k-2} + h_6\Delta u_{k-3} - h_1\Delta u_{k-1} - h_2\Delta u_{k-2} - h_3\Delta u_{k-3})$$
$$(i = 3|h_4\Delta u_{k-1} - h_1\Delta u_{k-1} + h_5\Delta u_{k-2} - h_2\Delta u_{k-2} + h_6\Delta u_{k-3} - h_3\Delta u_{k-3})$$
$$(i = 3|(h_4 - h_1)\Delta u_{k-1} + (h_5 - h_2)\Delta u_{k-2} + (h_6 - h_1\Delta)u_{k-3})$$

$$\sum_{j=1}^{M-1} |h_{j+3} - h_j|\Delta\, u_{k-j}$$

$$\sum_{j=1}^{M-1} |h_{j+i} - h_j|\Delta\, u_{k-j}$$

$$y_{k+i} = \hat{y}_k + \underbrace{\sum_{j=1}^{i} h_i\Delta u_{k+i-j}}_{framtid} + \underbrace{\sum_{j=1}^{M-1} |h_{j+i} - h_j|\Delta\, u_{k-j}}_{förflutet}$$

(Shridhar & Cooper, 1997) (Kufoalor, et al., 2016)

10 Bilaga 4: Härledning av prestationsindexet och anpassade prediktionsekvationer för integrerande dynamik

$$y_{k+i} = y_0 + \sum_{j=1}^{i} h_j \Delta u_{k+i-j} + \sum_{j=i+1}^{M-1} h_j \Delta u_{k+i-j} + d_k \quad (1)$$

Före optimeringen är utförd är den nuvarande och de framtida styråtgärderna okända därför kan y_{k+i} kan förenklas till y_k, y_0 är stationära tillståndet.

$$y_{k+i} = y_k = y_0 + \sum_{j=i+1}^{M-1} h_j \Delta u_{k+i-j} + d_k \quad (2)$$

$$d_k = \hat{y}_k - y_0 - \sum_{i=1}^{M-1} h_i \Delta u_{k-i} \quad (3)$$

För att den framtida prediktionen ska följa börvärdet gäller:

$$e = r - y_{k+i} = 0\,, \qquad i = 1,2,3,\dots,P \quad (4)$$

y_{k+1} i (4) substitueras med (1), då fås:

$$\underbrace{r - y_0 - \sum_{j=1}^{i} h_j \Delta u_{k+i-j} - d_k}_{Predikterade\ felet\ med\ tidigare\ insignaler\ (e_{k+i})} = \underbrace{\sum_{j=i+1}^{M-1} h_j \Delta u_{k+i-j}}_{Rådande\ och\ framtida\ insignaler} \quad (5)$$

(5) ger uttrycket (6) som kan minimeras, A är stegsvarets Toeplitz-matris.

$$e = A\Delta u \quad (6)$$

$$\min_{\Delta u} J = (e - A\Delta u)^T (e - A\Delta u) \quad (7)$$

Åtgärdsdämpandefaktorn R läggs till för att bestraffa onödiga styråtgärder.

$$\min_{\Delta u} J = (e - A\Delta u)^T (e - A\Delta u) + \Delta u^T R \Delta u \quad (8)$$

Stegsvaret för en integrerande process är en rak linje från senaste mätvärdet till det nuvarande mätvärdet. Då gäller inte längre (1), ekvationen modifieras till:

$$y_{k+i} = y_0 + i(\hat{y}_k - \hat{y}_{k-1}) + d_k \quad (10)$$

$$d_k = \hat{y}_k - y_0 - (\hat{y}_k - \hat{y}_{k-1}) \quad (11)$$

Genom substitution av d_k i (10) med (11) fås (12), y_0 elimineras.

$$y_{k+i} = y_k = i(\hat{y}_k - \hat{y}_{k-1}) + \hat{y}_k - (\hat{y}_k - \hat{y}_{k-1}) \quad (12)$$

(10) insättas i (13).

$$r - y_k = 0 \quad (13)$$

$$r - y_0 - i(\hat{y}_k - \hat{y}_{k-1}) - d_k = 0 \quad (14)$$

d_k kan substitueras med (11).

$$r - i(\hat{y}_k - \hat{y}_{k-1}) - \hat{y}_k + (\hat{y}_k - \hat{y}_{k-1}) = 0 \quad (15)$$

Felet minimeras med $A\Delta u$, (16) kan användas med det tidigare härledda prestationsindexet.

$$e = A\Delta u \quad (16)$$

(Dougherty & Cooper, 2003) (Shridhar & Cooper, 1997) (Gupta, 1998)

11 Bilaga 5: Beräkningar av vattenhammare effekten

Formel för att beräkna tryck förändring på grund av vattenhammare.

$$\Delta p = \frac{v \cdot \rho}{\sqrt{\rho\left(\kappa + \left(\frac{d}{E \cdot \delta}\right)\right)}} = v \cdot c \cdot \rho \tag{1}$$

v, Vätskans hastighet innan ventilen stängs

ρ, Vätskans densitet, för vatten i 20 °C: $\rho = 998{,}21$ kg/m^3 (Engineering toolbox, 2023)

κ, Vätskans kompressabilitet, för vatten i 20 °C: $\kappa = 4{,}8 \cdot 10^{-10}$ Pa^{-1}

d rör diameter, $d = 2{,}54 \cdot 10^{-2}$ m

l rörlängd före ventilen, $l = 10$ m

E Elastisitets koefficient, för stål rör: $E = 2{,}1 \cdot 10^{11}$ Pa

δ rörets väggtjocklek, för 1″ rör: $\delta = 4{,}55 \cdot 10^{-3}$ m (Engineering toolbox, 2023)

Formeln för att beräkna den kritiska stängningstiden.

$$t_{critical} = \frac{2 \cdot l}{c} \tag{2}$$

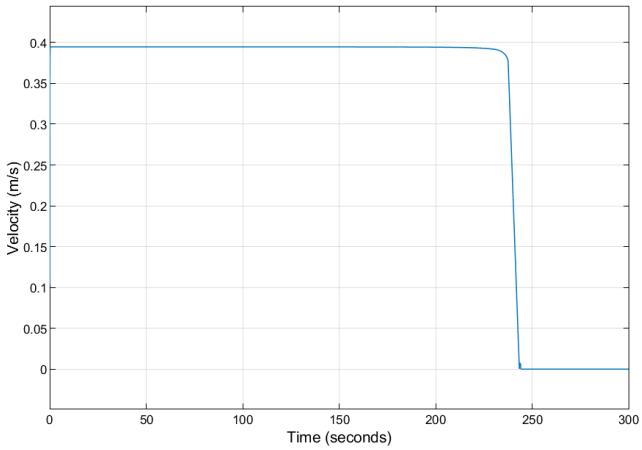
Ljudets hastighet i ett rör.

$$c = \frac{1}{\sqrt{\rho\left(\kappa + \left(\frac{d}{E \cdot \delta}\right)\right)}} \tag{3}$$

Med de förutnämda värden blir:

$$c = 1406{,}25 \text{ m/s}$$

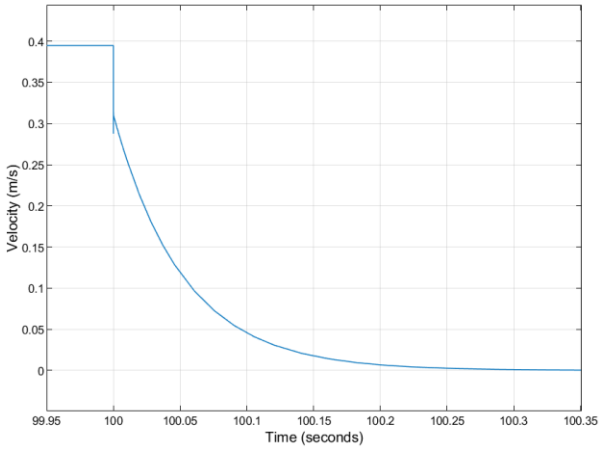
$$t_{critical} = 0.014222 \text{ s}$$



Hastigheten avläses från grafen som fåtts från simuleringen. Tryckförändringen beräknas med (1) med $v \approx 0.4$

$$\Delta p = 561.5 \text{ kPa}$$

PI-regulatorn begränsar stängningen av ventilen tillräckligt för att det inte ska ske en vattenhammare, tiden för att stänga ventilen är $\Delta T = 0.3\ s$



(Zevenhoven, 2013)

https://www.engineeringtoolbox.com/nominal-wall-thickness-pipe-d_1337.html

https://www.engineeringtoolbox.com/water-density-specific-weight-d_595.html

12 Bilaga 6: DMC Algoritmen

```
classdef (StrictDefaults) DMC < matlab.System
% En enkel DMC algoritm.
% Author: Jonathan Häggvik
% Description: DMC algoritmen är skriven som en del av min
% magisteravhandlig i Kemi- och processteknik

    properties (Nontunable) %% Del av prestationsindexet

        moveWeight {mustBeNonnegative} = 1 % Move suppression

        cvConHardLB {mustBeNumeric} = [] % CV hard constraint lower bound
        cvConHardUB {mustBeNumeric} = [] % CV hard constraint upper bound

        mvConHardLB {mustBeNumeric} = [] % MV hard constraint lower bound
        mvConHardUB {mustBeNumeric} = [] % MV hard constraint upper bound
        mvConHardRate {mustBeNumeric, mustBePositive} = 1 % MV hard constraint rate

        predHor {mustBeInteger, mustBePositive} = 20 % Prediction Horizon
        ctrlHor {mustBeInteger, mustBePositive} = 10 % Control Horizon
        predModel % Prediction model
        sampTime {mustBeInteger, mustBePositive} = 1 % Sample time

    end % Nontunable properties

    properties(Access = private)
        % Prediction matriser
        modlHor % Modell horisont
        H % Stegsvar
        L % Matris med ettor
        Ct % Toeplitz
        Mh % Hankel
        conStruct % villkor

        fminconOpt % Optimerings inställningar
        quadprogOpt
        upast
        mvPast % Föregående styråtgärder
        cvPast % Föregående mätvärden
    end % private properties

    methods
        ...
        % Konstruktor
        function this = DMC(varargin)
            % Support name-value pair arguments when constructing object
            setProperties(this,nargin,varargin{:})
        end

    end % methods (public)

    methods (Access = private)

        function setCt(this,S,P,H)
            % P Prediktions horisont
            % S Styr horisont
            % H Stegsvars koefficienter

            % Första raden i Toeplitz matrisen.
            rowsC = [H(1), zeros(1,S-1)];
            % Första kolumnen i Toeplitz matrisen.
            columnsC = H(1:P)';

            this.Ct = toeplitz(columnsC,rowsC);
        end % function getCt
```

```
        function setMh(this,P,M,H)
            % M Modell horisont
            % P Prediktions horisont
            % H Stegsvars koefficienter

            % Första kolumnen i Hankel matrisen.
            columnsH = H(1+1:P+1)';
            % Sista raden i Hankel matrisen.
            rowsH = [H(P+1:M), ones(1, P-1)*H(M)];
            Hh = hankel(columnsH,rowsH);

            rows = ones(P,1);
            columns = H(1:M-1)';
            sub = rows*columns;

            this.Mh = Hh-sub;
        end % function getMh


        function e = getIntegrE(this,cvMeas,cvRef)
            % e för integrerande processer

            i = (1:this.predHor)';

            if (cvMeas - this.cvPast == cvMeas)
                this.cvPast = cvMeas;
            end

            e = this.L * (cvRef - cvMeas)  + (cvMeas - this.cvPast) - i * (cvMeas -
this.cvPast);

        end

        function e = getE(this,cvMeas,cvRef)
            % e för självreglerande processer

            e = this.L * (cvRef - cvMeas) - this.Mh * this.mvPast;

        end

        % Prestationsindex
        function cost = CostFunction(this,mv,e)


            cost = (e - this.Ct * mv)' * (e - this.Ct * mv) + mv' * this.moveWeight * mv;

        end


        function u = optimera(this, cvMeas, cvRef)

            %% VILLKOR
            % Nedre och övre gräns för \delta u

            % Likhetsvillkor
            Aeq = [];
            beq = [];

            % Olikhetsvillkor A*x <= b
            % a11 * x(1) + a12 * x(2) + ... + a1N * x(N) <= b1
            % a21 * x(1) + a22 * x(2) + ... + a2N * x(N) <= b2
            % Initierar matriserna för villkoren
            A = zeros(2+2*this.ctrlHor, this.ctrlHor);
            b = zeros(2+2*this.ctrlHor, 1);

            % STYRDA VARIABLER
```

```
            if ~isempty(this.cvConHardLB) % Nedre gränsen
                A(2,:) = -this.Ct(this.predHor, :);
                % cvMeas < cvConHard
                b(2) = -this.cvConHardLB + cvMeas;

            end

            if ~isempty(this.cvConHardUB) % Övre gränsen
                % Ct(P)*x(1) + H(P-1)*x(2) + ... + H(P-S+1)x(S) <= CVUB
                % behöver inte kolla alla endast att de sista inte bryter
                % mot villkoren.
                A(1,:) = this.Ct(this.predHor, :);

                b(1,1) = this.cvConHardUB - cvMeas;

            end

            % MANIPULERADE VARIABLER

            % Nedre och övre gräns för \delta u
            lb = ones(this.ctrlHor,1) .* (-this.mvConHardRate);
            ub = ones(this.ctrlHor,1) .* this.mvConHardRate;

            if ~isempty(this.mvConHardLB) % Nedre gränsen
                A(3:2+this.ctrlHor, :) = -toeplitz(ones(this.ctrlHor,1),[1,
zeros(1,this.ctrlHor-1)]);

                b(3:2+this.ctrlHor) = -(this.mvConHardLB) + this.upast;

            end

            if ~isempty(this.mvConHardUB) % övre gränsen
                A(3+this.ctrlHor:2+2*this.ctrlHor, :) = toeplitz(ones(this.ctrlHor,1),[1,
zeros(1,this.ctrlHor-1)]);

                b(3+this.ctrlHor:2+2*this.ctrlHor) = this.mvConHardUB - this.upast;

            end

            if isempty(this.cvConHardLB) && ...
               isempty(this.cvConHardUB) && ...
               isempty(this.cvConHardLB) && ...
               isempty(this.mvConHardUB)

                A = [];
                b = [];

            end

            ...
            %% Optimering

            % kommentera ut det e som inte används.
            e = this.getE(cvMeas,cvRef);
%             e = this.getIntegrE(cvMeas,cvRef);

            mv0 = zeros(this.ctrlHor,1);

            cost = @(mv)this.CostFunction(mv,e);

%             [delta_u, functionValue,~, output] =
fmincon(cost,mv0,A,b,Aeq,beq,lb,ub,[],this.fminconOpt);

            % Xi & Li 2019 p129

            H = 2 * (this.Ct'*this.Ct + this.moveWeight);

            f = - 2 * (e)' * this.Ct;
```

```
            % f' * x + 1/2 * x' * H * x
            [delta_u, functionValue,~, output] =
quadprog(H,f,A,b,Aeq,beq,lb,ub,[],this.quadprogOpt);

            disp(output);
            disp(functionValue);

            this.upast = this.upast + delta_u(1);
            u = this.upast;

            this.mvPast = circshift(this.mvPast,1);
            this.mvPast(1) = delta_u(1);

            this.cvPast = cvMeas;

        end % function optimera

    end % methods (Access = private)

    % Metoder som simulink använder.
    methods (Access = protected)
       % Funktioenen setupImpl kallas en gång i början av simuleringen.
        function setupImpl(this)

            % Granskar ifall den inmatade modellen inte fyller kraven.
            if ~(isa(this.predModel,'lti'))
                ME = MException('DMC_build6:UnsupportedModelType',...
                    'Prediction model must be of lti type.',this.predmodel);
                throw(ME);
            end

            % Granskar längden på horisonter.
            if (this.predHor < this.ctrlHor)
                ME = MException('DMC_build6:ValueError',...
                    'predHor must be greater than ctrlHor.');
                throw(ME);
            end

            %
            this.modlHor = this.predHor + 1;

            % h0 = 0, Tidsrymd för stegförsöket.
            timeSpan = 1 : this.sampTime : this.modlHor*this.sampTime;

            % Utför stegsförsök på prediktionsmodellen och sparar resultat.
            this.H = step(this.predModel,timeSpan); % size(modlHor, 1)

            % Skapar och sparar matris med ettor
            this.L = ones(this.predHor,1);

            % Skapar och sparar Toeplitz matris.
            this.setCt(this.ctrlHor,this.predHor,this.H);

            % Skapar och sparar Hankel matris.
            this.setMh(this.predHor,this.modlHor,this.H);

            % Andra variabler som initieras.
            this.fminconOpt = optimoptions('fmincon', 'Algorithm', 'interior-point');
            this.quadprogOpt = optimoptions('quadprog','Algorithm','interior-point-convex');

            this.mvPast = zeros(this.modlHor-1,1);

            this.upast = 0;
            this.cvPast = 0;

        end % function setupImpl

        % Simulink skickar en förfrågan om algoritmens sampeltid.
```

```
        function sts = getSampleTimeImpl(this)
            sts = createSampleTime(this,'Type','Discrete',...
            'SampleTime',this.sampTime,'OffsetTime',0.5);
        end

        % Denna funktion utförs varje sampeltid.
        % Namnet stepImpl hör till de funktioner simulink kallar.
        function mvCur = stepImpl(this,cvMeas,cvRef)
            % Implement algorithm. Calculate y as a function of input u and
            % discrete states.
            mvCur = this.optimera(cvMeas,cvRef);

        end % stepImpl

        %% Andra funktioner
        % Nödvändiga funktioner som definerar egenskaper på
        % utsignalen av det definierade system blocket

        function resetImpl(~)
            % Initialize / reset discrete-state properties
        end

        function s1 = getOutputSizeImpl(~)
            s1 = [1 1];
        end

        function d1 = getOutputDataTypeImpl(~)
            d1 = 'double';
        end

        function c1 = isOutputComplexImpl(~)
            c1 = false;
        end

        function c1 = isOutputFixedSizeImpl(~)
            c1 = true;
        end

    end % methods (Access = protected)

end
```

13 Bilaga 7: Symboliska stegsvarsprediktionsmatriser i matlab

```
clear; %clc;
```

Prediction med Stegsvar

$y_k = h(q^{-1})\Delta u_k$

```
M = 6; % Modell horisont.
P = 5; % Prediktions horisont (P < M).
S = 4; % Styr horisont (S <= P).
```

$h_0 = 0$, för strikt propera system, antagandet är implicit i beräkningen.

$\Delta = (1 - q^{-1})$

```
syms h [1, M]
% h = [0, h]
d = [1, -1];
```

Delta matriser

Detta stycke finns endast för att bevisa att L matrisen är en kolumn med ettor, i praktiken kan L matrisen formas som det.

Toeplitz ($N \times N$)

```
% Första kolumnen i Toeplitz-matrisen.
cd = [d(1:min([P,length(d)])), ...
      zeros(1,P-length(d))];
% Första raden i Toeplitz-matrisen.
rd = [d(1), zeros(1,P-1)];
Cd = toeplitz(cd,rd);
```

Hankel ($N \times N$)

```
% Första columnen i Hankel-matrisen.
cd = [d(2:min(length(d),P+1)),...
      zeros(1, P-length(d)+1)];
% Sista raden i Hankel-matrisen.
if (P < length(d))
    rd = [d(2:length(d)), zeros(1, P-1)];
```

```
else
    rd = [zeros(1, P)];
end
Hd = hankel(cd,rd);


L = -inv(Cd)*Hd
```

Toeplitz ($P \times S$)

```
% Första raden i Toeplitz-matrisen.
rowsC = [h(1), zeros(1,S-1)]
% Första kolumnen i Toeplitz-matrisen.
columnsC = h(1:P).'
Ct = toeplitz(columnsC,rowsC)
```

Hankel ($M \times P - 1$)

För att bygga prediktionen skiljer sig matrisen från de andra modellerna.

```
% Första kolumnen i Hankel-matrisen.
columnsH = h(1+1:P+1).'
% Sista raden i Hankel-matrisen.
rowsH = [h(P+1:M), ones(1, P-1)*h(M)]
Hh = hankel(columnsH,rowsH)
```

Matrisen som subtraheras från Hankel-matrisen för att ta hänsyn till de föregående insignalerna, skapas genom att kombinera en Hankel-matris av ettor och stegssvar koefficienterna.

```
columns = h(1:M-1)
rows = ones(P,1)
sub = rows*columns


Mh = Hh-sub
```

Prediktionen

$y_{k+1} = C_H \Delta u_k + L y_k + M \Delta u_{k-1}$

```
syms y_future [P, 1]    % y-k+1 y_k+2 ... y_k+P
syms u_future [S-1, 1]  % u_k u_k+1 ... u_k+S-1
syms u_past [M-1, 1]   % u_k-1 u_k-2 ... u_k-(M+1)
```

```
syms y_meas
syms Delta

u_future = ['u_k'; u_future].*Delta
u_past = u_past.*Delta
y_future

y_future = Ct*u_future + L*y_meas + Mh*u_past
```

Alternativ metod

```
% Första kolumnen i Hankel-matrisen.
ch = [h(2:P+1)];
% Sista raden i Hankel-matrisen.
rh = [h(P+1:M), zeros(1, P-1)];
Hh = hankel(ch,rh)

y_future = Ct*u_future + Hh*u_past - L * (h(1:M-1)*u_past)
```

Referenser

(Kufoalor, et al., 2016)

(Rossiter & Kouvaritakis, 2001)

(Xi & Li, 2019)

(Haber, et al., 2011)

14 Bilaga 8: Simuleringsmodellens skript

```
clc; clear;
%% Parameters

% Vätskan: Vatten
fluidDensity = 997.19; % [kg/m^3]
fluidViscosity = 1.0016e3; % [Pa*s]
frictionFactor = 0;
gravityConstant = 9.81; % [m/s^2]

% Tank (1:2 - diameter to height ratio)
tankVolume = 48/1e3; % [m^3] V = r^2 * pi * 4 * r
tankRadius = (tankVolume/(4*pi()))^(1/3); % [m]
tankHeight = tankRadius*4; % [m]
tankArea = pi() * tankRadius^2; % [m^2]

% Rör
pipeDiameter = 25.4e-3; % [m]
pipeArea = pi() * (pipeDiameter/2)^2; % [m^2]

% Inflöde
flowInVolume = 8/60; % [l/s]
flowInVelocity = flowInVolume/(1000*pipeArea); % [m/s]
levelIncrementMin = (flowInVolume/tankVolume); % [%/min]

%% Simulation
model = 'SimuleringsModel';
open_system(model);

% Length of simulation
simulationLength = 60*16 %60*60; % [min] Time unit of model is seconds

% Sample rate for massFlowIn
flowInMeasSampleRate = 80; %  % [min]
samplingTime = seconds(0:flowInMeasSampleRate:simulationLength)';

flowInSignal = rand(length(samplingTime/2),1);
% flowInSignal = ones(size(samplingTime));
% flowInSignal = zeros(size(samplingTime));
flowInSignalTT = timetable(samplingTime,flowInSignal);

tankInitialVolume = 0 * tankVolume;

%% Model predictive control

refValue = [zeros(120,1);  50 * ones(simulationLength-120 + 1,1)];
refDuration = seconds(0:simulationLength)';
refTT = timetable(refDuration,refValue);

gain = -1/(tankVolume*1e3)*100
predModel = tf(gain,[1, 0],'iodelay',0);

sampTime = 1;
predHor = 20;
ctrlHor = 10;
```