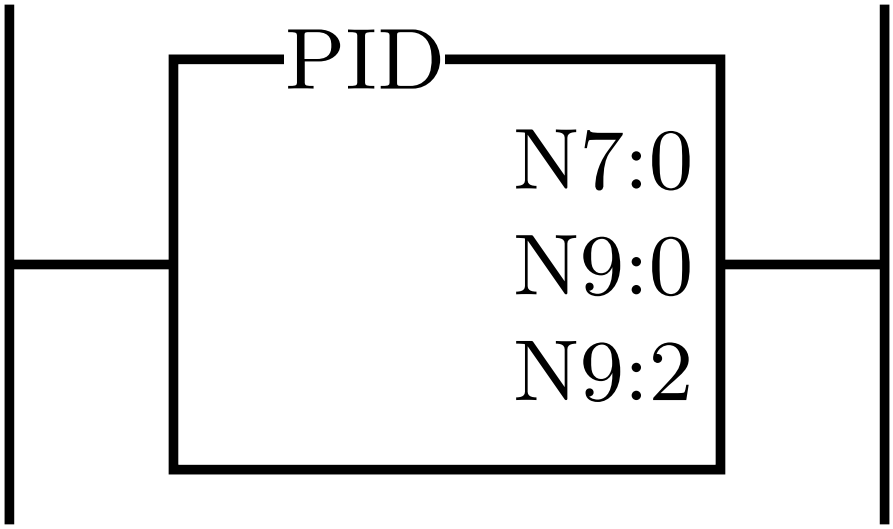
Laboration i industriella styrsystem

PID-implementering i PLC

Denna version: 31 januari 2018



REGLERTEKNIK
AUTOMATIC CONTROL
LINKÖPING

Namn:	
Personnr:	
Datum:	
Godkänd:	

Tabell 1. Godkännande – Förberedelseuppgifter

Uppgift	**Datum**	**Godkänd**
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		

Innehåll

Kapitel 1

Syfte och organisation

1.1 Laborationens syfte och mål

Syftet med denna laboration är att ge kunskaper i att implementera en PID-regulator i en PLC och att då särskilt hantera följande aspekter:

- Använda skalning av variabler för att hantera den avvägning mellan risk för överspill och upplösning som uppstår p g a heltalsberäkningar.
- Implementera någon metod som förhindrar integratoruppvridning.
- Implementera I- och D-del som exekveras med lägre samplingsfrekvens än P-delen.
- Skapa ett mera avancerat användargränssnitt jämfört med det som utvecklades i Lab 1.

Målen för laborationen, d v s vad som krävs för att laborationen ska bli godkänd, framgår av uppgiftsformuleringen i avsnitt 5.

Denna laboration är starkt kopplad till den den första laborationen i kursen, och i Lab 2 ska ni arbeta vidare på resultaten från Lab 1.

1.2 Organisation

Förberedelse

Laborationen kräver omfattande förberedelser enligt följande:

- Studera den teori och de metoder som laborationen bygger på, d v s främst avsnitt 3 i kurskompendiet om tidsdiskret implementering av PID-regulatorer, metoder för att hantera integratoruppvridning samt mera avancerad stegkodsprogrammering (ladderprogrammering), se Appendix till kurskompendiet.
- Gör förberedelseuppgifterna till laborationen. Dessa återfinns i avsnitt 6 av detta Lab-PM.
- Förberedelseuppgifterna ska redovisas och bli godkända vid ett s k Helddesk-tillfälle, vilket hålls några dagar före labtillfället. Information om tid och plats för dessa tillfällen ges via kursens hemsida. Helpdesktillfället måste vara genomfört och godkänt för att få genomföra laborationen.

Genomförande

Genomförandet av laborationen i kurslabbet sker på följande sätt:

- Laborationen inleds med ett schemalagt och handlett labpass om fyra timmar.
- Om de fyra timmarna vid det inledande labpasset inte räcker för att genomföra laborationen kan man boka ytterligare tid med hjälp av de bokningslistor som finns vid labplatsen. Notera dock att man inte får ha mer än sammanlagt fyra timmar bokade samtidigt.
- Utöver det inledande labpasset kan vid behov hjälp fås av handledarna efter överenskommelse.

Redovisning

- Laborationen redovisas för laborationsassistenten muntligt tillsammans med en genomgång av programmet. Båda personerna i labgruppen måste kunna redogöra för hur de olika laborationsuppgifterna lösts.

Kapitel 2

Tidsdiskret implementering av PID-regulator

2.1 Grundekvationer

Teorin för hur man tar fram en tidsdiskret approximation av en PID-regulator beskrivs tämligen utförligt i kurskompendiet, men för att underlätta förberedelserna för laborationen sammanfattas de viktigaste aspekterna nedan.

I kontinuerlig tid ges PID-regulatorn av ekvationen

$$u(t) = K(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\dot{e}(t)) \tag{2.1}$$

där $u(t)$ betecknar styrsignalen, och $e(t)$ betecknar reglerfelet, d v s skillanden mellan referens- och utsignal

$$e(t) = r(t) - y(t) \tag{2.2}$$

Genom att approximera I- och D-delarna enligt de metoder som presenteras i kurskompendiet fås den tidsdiskreta formen

$$u_k = K(e_k + \frac{T_s}{T_i}\sum_{l=1}^{k} e_l + \frac{T_d}{T_s}(e_k - e_{k-1})) \tag{2.3}$$

där u_k, e_k och y_k betecknar styrsignal, reglerfel och utsignal i samplingstidpunkten $t = k \cdot T_s$ där T_s betecknar tidsintervallet mellan samplingstidpunkterna. Av praktiska skäl kan man införa hjälpvariablerna

$$I_k = K\frac{T_s}{T_i}\sum_{l=1}^{k} e_l = R_i\sum_{l=1}^{k} e_l \tag{2.4}$$

respektive

$$D_k = -K\frac{T_d}{T_s}(y_k - y_{k-1}) = -R_d(y_k - y_{k-1}) \tag{2.5}$$

där

$$R_i = K\frac{T_s}{T_i} \qquad R_d = K\frac{T_d}{T_s} \tag{2.6}$$

Av praktiska skäl "deriveras" enbart utsignalen när den approximativa derivatan bildas i (2.5).

Genom att skriva den sista termen i integraldelen, d v s I_k, separat kan I_k uppdateras rekursivt enligt

$$I_k = R_i \sum_{l=1}^{k-1} e_l + R_i e_k = I_{k-1} + R_i e_k \tag{2.7}$$

Grundekvationerna för den tidsdiskreta PID-regulatorn kan därmed sammanfattas:

$$\begin{aligned} e_k &= r_k - y_k \\ I_k &= I_{k-1} + R_i e_k \\ D_k &= -R_d(y_k - y_{k-1}) \\ a_k &= I_k + D_k \\ u_k &= Ke_k + a_k \end{aligned} \tag{2.8}$$

2.2 Styrsignalbegränsning

PID-regulatorn i (2.8) tar inte hänsyn till att det i praktiken alltid finns begränsningar på hur stor styrsignalen, u_k, kan vara. I denna laboration är det pumpen som sätter begränsningarna, och den kan t ex inte suga ut vatten, vilket ger en nedre begränsning på noll. Vidare finns det en övre gräns för hur mycket vatten den kan pumpa. För att hantera dessa begränsningar är det lämpligt att införa ytterligare en signal v_k som bara används internt i regulatorn och att sedan jämföra denna med den övre respektive undre gränsen för den verkliga styrsignalen. Det är alltså en trunkerad version av v_k som skickas till pumpen. Regulatorn, kompletterad med denna funktion, ges därmed av

$$\begin{aligned} e_k &= r_k - y_k \\ I_k &= I_{k-1} + R_i e_k \\ D_k &= -R_d(y_k - y_{k-1}) \\ a_k &= I_k + D_k \\ v_k &= Ke_k + a_k \\ u_k &= \begin{cases} u_{\max}, & \text{om} \quad v_k > u_{\max}, \\ v_k, & \text{om} \quad u_{\min} \leq v_k \leq u_{\max}, \\ u_{\min}, & \text{om} \quad v_k < u_{\min}. \end{cases} \end{aligned} \tag{2.9}$$

Den diskreta PID-regulatorn (2.9) lämpar sig väl för implementering i ett sekventiellt system såsom PLC:n i labbet. Algoritm 1 beskriver en möjlig implementation.

Algoritm 1 Implementering av PID-regulator (2.9). Vid varje samplingstidpunkt k genomlöps stegen nedan.

1. Mät systemets utsignal, y_k
2. Beräkna felet, $e_k = r_k - y_k$
3. Uppdatera den approximerade integralen och den approximerade derivatan,
   $I_k = I_{k-1} + R_i e_k$
   $D_k = -R_d\left(y_k - y_{k-1}\right))$
4. Beräkna totala inverkan av I- och D-del,
   $a_k = I_k + D_k$
5. Beräkna den otrunkerade styrsignalen $v_k = Ke_k + a_k$
6. Beräkna den trunkerade styrsignalen u_k enligt (2.9)
7. Vänta tills nästa samplingsögonblick och börja sedan om på steg 1.

2.3 Långsammare exekvering av I- och D-del.

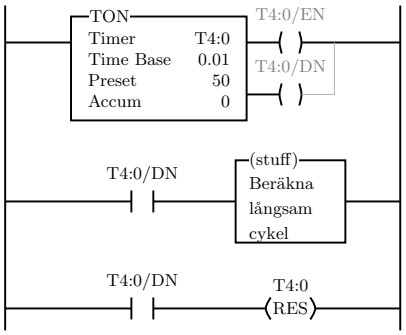
Den regulator som beskrivs i (2.9) uppdateras med tidsintervallet, T_s. Ett kort samplingsintervall är att föredra för att minimera tidsfördröjningen från en mätning till dess att utsignalen är uträknad och för att reagera snabbt på eventuella förändringar. Att köra PLC:n med kort sampeltid skapar problem för I- och D-delen eftersom PLC:n har begränsad upplösning. Lösningen är att låta I- och D-delarna, d v s variablerna I och D i ekvation (2.9), uppdateras mer sällan än v_k. Regulatorn kan därför modifieras enligt nedan där variablerna med index k uppdateras med tidsintervallet T_s och variablerna med index m uppdateras med (det längre) tidsintervallet $\tilde{T}_s$. I ekvationerna förutsätts, vilket också gäller i laborationen, att det längre intervallet $\tilde{T}_s$ är en multipel av det kortare intervallet T_s.

$$\begin{aligned} e_k &= r_k - y_k \\ I_m &= I_{m-1} + \tilde{R}_i e_m \\ D_m &= -\tilde{R}_d(y_m - y_{m-1}) \\ a_m &= I_m + D_m \\ v_k &= Ke_k + a_m \\ u_k &= \begin{cases} u_{\max}, & \text{om} \quad v_k > u_{\max}, \\ v_k, & \text{om} \quad u_{\min} \le v_k \le u_{\max}, \\ u_{\min}, & \text{om} \quad v_k < u_{\min}. \end{cases} \end{aligned} \tag{2.10}$$

där

$$\tilde{R}_i = K\frac{\tilde{T}_s}{T_i} \qquad \tilde{R}_d = K\frac{T_d}{\tilde{T}_s} \tag{2.11}$$

där $\tilde{T}_s$ betecknar det samplingsintervall som D- och I-delen uppdateras med. I laborationen uppdateras dessa med frekvensen 2 Hz, d v s $\tilde{T}_s = 0.5$ s, medan P-delen uppdateras med 100 Hz, d v s $T_s = 0.01$ s.



Figur 2.1. Kod för att skapa långsam cykel. Instruktionerna i den mellersta raden (som illustreras med blocket (stuff)) körs två gånger i sekunden.

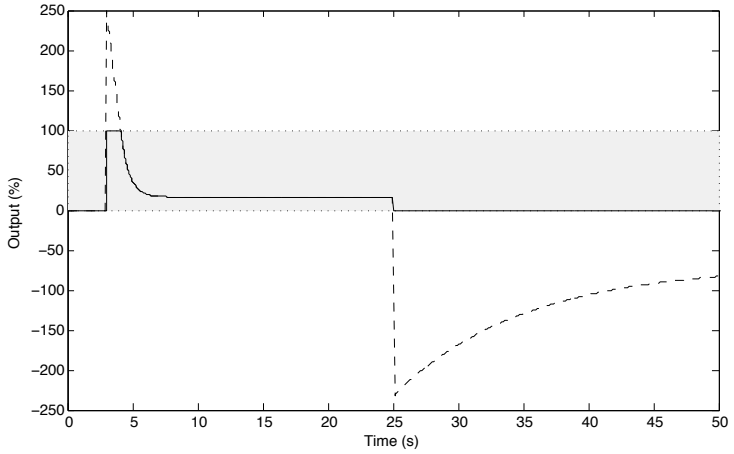
Ett sätt att åstadkomma en längre sampeltid är att använda en timer. Den timer vi föreslår att man använder heter `TON`. Figur 2.1 visar hur timern kan användas för att köra integral- och derivatadelarna med en längre sampeltid. Då timern har räknat upp till Time base $\cdot$ Preset $= 50 \cdot 0.01 = 0.5s$ sätts `T4:0/DN` till 1. Om `T4:0/DN` är 1 så beräknas I_m och D_m enligt (2.10) och timerblocket `T4:0` nollställs. Algoritm 2 är en utökning av algoritm 1 med den långsamma cykeln. För mer information om timern, se Appendix i kurskompendiet.

Algoritm 2 Implementering av PID-regulator (2.10) med långsam I- och D-del.

1. Mät systemets utsignal, y_k
2. Beräkna felet, $e_k = r_k - y_k$
3. Med tidsintervallet $\tilde{T}_s$, d v s då timern räknat klart, beräkna den approximerade integralen och derivatan, d v s
   Om `T4:0/DN` $==$ 1 beräkna
   $I_m = I_{m-1} + \tilde{R}_i e_m$
   $D_m = -\tilde{R}_d(y_m - y_{m-1})$
4. Beräkna totala inverkan av I- och D-del och starta om räknaren
   $a_m = I_m + D_m$
   Nollställ timern
5. Beräkna den otrunkerade styrsignalen $v_k = Ke_k + a_m$
6. Beräkna den trunkerade styrsignalen u_k enligt (2.10)
7. Vänta tills nästa samplingsögonblick och börja sedan om på steg 1.

2.4 Skydd mot integratoruppvridning

En negativ konsekvens av att styrsignalen är begränsad är att s k integratoruppvridning kan inträffa, vilket kan leda till försämrade egenskaper hos reglersystemet. För att motverka integratoruppvridning skall villkorlig integration implementeras, vilket innebär att uppdateringen av integralen upphör om styrsignalen är mättad. I kurskompendiet beskrivs denna och alternativa metoder i mera detalj. Styrsignalen är omättad, d v s är inom det tillåtna intervallet, om $u_{min} \leq v_k \leq u_{max}$. Med andra ord skall alltså I_m inte uppdateras om $u_k \neq v_k$. Se figur 2.2 för ett exempel på en mättad styrsignal.



Figur 2.2. Ett exempel där den beräknade styrsignalen v_k (streckad) är större (och mindre) än det tillåtna värdet på utsignalen u_k (heldragen). Notera att v_k sammanfaller med u_k när signalen är omättad, det vill säga $v_k = u_k$ då signalen är omättad.

Algoritm 3 är en utökad version av algoritm 2 där villkorlig integration är implementerad. Mer information om detta samt andra uppvridningsskydd och en mer detaljerad förklaring finns i kurskompendiet.

Algoritm 3 Algoritm för implementering av PID-regulator (2.10) med villkorlig integration och långsam integral- och derivatadel.

1. Mät systemets utsignal, y_k.
2. Beräkna felet, $e_k = r_k - y_k$
3. Med tidsintervallet $\tilde{T}_s$, d v s då timern räknat klart, beräkna den approximerade integralen och derivatan, d v s
   Om T4:0/DN == 1 beräkna
   $D_m = -\tilde{R}_d(y_m - y_{m-1})$
   Om $v == u$
   $I_m = I_{m-1} + \tilde{R}_i e_m$
   Annars
   $I_m = I_{m-1}$
4. Beräkna totala inverkan av I- och D-del och starta om räknaren
   $a_m = I_m + D_m$
   Nollställ timern
5. Beräkna den otrunkerade styrsignalen $v_k = Ke_k + a_m$
6. Beräkna den trunkerade styrsignalen u_k enligt (2.10)
7. Vänta tills nästa samplingsögonblick och börja sedan om på steg 1.

2.5 Filtrering av D-del

En begränsning när man använder D-delen, d v s derivatan av reglerfelet, i PID-regulatorn är att den blir känslig för mätstörningar, d v s osäkerheter i mätsignalen y_k. Ett sätt att hantera detta är att kombinera D-delen med ett s k lågpassfilter, som till viss del tar bort denna effekt. I kurskompendiet visas hur kombinationen av D-verkan och lågpassfiltrering kan approximeras i tidsdiskret form via ekvationen

$$D_k = \frac{N}{N + T_s} D_{k-1} - \frac{KT_d}{N + T_s}(y_k - y_{k-1}) \tag{2.12}$$

Denna ekvation skiljer sig från motsvarande ekvation i kurskompendiet på två sätt. Dels verkar, på samma sätt som ovan, D-delen enbart på utsignalen y, d v s vi avstår av från att derivera referenssignalen r, och dels har koefficienterna framför derivatan i PID-regulatorn, d v s K och T_d tagits med i ekvationen.

I analogi med behandlingen ovan kan man sedan införa hjälpvariablerna

$$R_d = K\frac{T_d}{T_s + N} \quad R_N = \frac{N}{N + T_s} \tag{2.13}$$

vilket ger

$$D_k = R_N D_{k-1} - R_d(y_k - y_{k-1}) \tag{2.14}$$

PID-regulatorn, kompletterad med filtrerad D-del, ges av algoritm 4 nedan.

Algoritm 4 Algoritm för implementering av PID-regulator (2.10) med villkorlig integration, långsam integral- och derivatadel samt dervatafilter.

1. Mät systemets utsignal, y_k.
2. Beräkna felet, $e_k = r_k - y_k$
3. Med tidsintervallet $\tilde{T}_s$, d v s då timern räknat klart, beräkna den approximerade integralen och derivatan, d v s
   Om T4:0/DN == 1 beräkna
   $D_m = \tilde{R}_N D_{m-1} - \tilde{R}_d(y_m - y_{m-1})$
   Om $v == u$
   $I_m = I_{m-1} + \tilde{R}_i e_m$
   Annars
   $I_m = I_{m-1}$
4. Beräkna totala inverkan av I- och D-del och starta om räknaren
   $a_m = I_m + D_m$
   Nollställ timern
5. Beräkna den otrunkerade styrsignalen $v_k = Ke_k + a_m$
6. Beräkna den trunkerade styrsignalen u_k enligt (2.10)
7. Vänta tills nästa samplingsögonblick och börja sedan om på steg 1.

2.6 Beräkningar med heltal

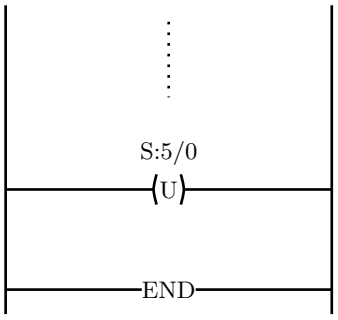
Den PLC som används i den här laborationen använder endast med 16-bitars heltal med en teckenbit. Det är en stor begränsning och här ges ledning till hur det kan hanteras.

2.6.1 Överspill

Begränsningen till 16 bitar för heltal med tecken innebär att alla tal är begränsade till intervallet $[-32768, 32767]$. Beräkningar som leder till tal utanför det här intervallet ger ett ogiltigt resultat och måste hanteras. Problemet kallas *överspill* (på engelska: *overflow*). PLC:n som används i laborationen hanterar överspill genom att:

- Trunkera svaret till tillåtet intervall. Exempelvis blir $400 \cdot 100 = 40000$ trunkerat till 32767.
- Meddela att överspill har inträffat genom att sätta biten `S:5/0` till 1. Denna bit kallas för *minor error bit*.

För att hantera överspill behöver programmet på något sätt ta hand om felet och återställa `S:5/0`. Eftersom man i det stegkodsprogram som tas fram i laborationen kommer att ta hand om eventuellt överspill kan man nollställa biten `S:5/0` innan slutet av programmet med operationen `OTU`(*unlatch*), se figur 2.3. Om `S:5/0` inte nollställs innan programmets slut kommer PLC:n att avbryta exekveringen och FLT-lampan kommer att blinka rött. Om det händer kan man ofta få vägledning om var det blivit fel genom att leta efter register som fått värdet 32767 eller -32768.



Figur 2.3. Genom att utföra en unlatch (nollställning) av `S:5/0` sist i programmet fångas eventuellt överspill upp och programmet kan fortsätta att exekvera.

2.6.2 Skalning

På grund av att tal i PLC:n representeras med heltal måste de skalas för att kunna representera tal som inte är heltal mellan -32768 och 32767. Vid beräkningar är det därför viktigt att ta hänsyn till skalningen.

- Addition (subtraktion): Talen som adderas (subtraheras) måste ha *samma* skalfaktor.
- Multiplikation (division): Talen behöver inte ha samma skall faktor men det är viktigt att vara medveten om att den resulterande skalfaktorn är produkten (kvoten) av skalfaktorerna för de båda talen. Vid multiplikation är det viktigt att se till att resultatet inte blir för stort.

Vilket av talen som skall skalas upp/ner innan beräkning beror på hur man på bästa sätt behåller god upplösning samtidigt som man undviker överspill.

Kapitel 3

Utrustning och programvara

3.1 Datorer och programvara

I laborationen används samma utrustning och programvara som i Lab 1, d v s en PC med programvarorna RSLogix och InTouch samt PLC:n. För att lösa uppgifterna i denna laboration bygger man vidare på resultaten från Lab 1 i form av PLC-programmet och operatörsgränssnittet. För frågor kring handhavandet av utrustning och programvaror hänvisas till PM för Lab 1 samt den Lathund som finns vid labplatsen. En viktig skillnad jämfört med den första laborationen är att Lab 2 kommer att ställa avsevärt större krav på hur man använder datorns minne, d v s i vilka register man lagrar de variabler och delresultat som är aktuella vid implementeringen av PID-regulatorn. Därför finns en föreslagen plan för hur detta ska göras, och den presenteras i nästa avsnitt. **Ta för vana att skriva ner era variabelnamn i tabellen då en ny tag definieras i In Touch.**

3.2 Plan för minnesanvändning

Här presenteras ett förslag på hur minnet i PLCn kan användas. Först påminner vi om några intressanta värden som är givna av sammanhanget. Förslaget bygger på att allt minne som hör till den nya regulatorn ska samlas i datafilen N10. Registrena N10:14 - N10:16 kan användas för att lagra mellanresultat. **Var dock noga med att skriva ner var ni lägger era olika mellanresultat och med vilken skalfaktor de sparas.** Utöver registren som nämns ovan läggs flaggor och heltalsvariabler enligt tabell 3.2. Att en flagga är satt

Tabell 3.1. Skalfaktorer för de variabler som används i programmet. Notera att flaggan RG måste vara satt till 1 för att skalfaktorn 100 skall gälla för K och T_i. Samtliga register i denna tabell är av datatypen I/O real. Notera att skalfaktorerna i flera fall skiljer sig från de som användes i laboration 1.

Reg.	**Uttryck**	**Faktor**	**Kommentar**
I:3.0		1	Utsignal från övre tank
I:3.1		1	Utsignal från undre tank
O:3.0		1	Styrsignal till pump
N7:2	r_k	1/2	Referenssignal
N7:3	K	100	Parameter för inbyggda regulatorn
N7:4	T_i	100	Parameter för inbyggda regulatorn
N7:5	T_d	100	Parameter för inbyggda regulatorn
N9:0	$y_{\text{upper},k}$	1/2	Tanknivå övre tanken
N9:1	$y_{\text{lower},k}$	1/2	Tanknivå undre tanken
N9:2	$u_{\text{pump},k}$	1/2	Styrsignal till pumpen
N9:3	$u_{\text{pump,manuell},k}$	1/2	Manuell styrsignal till pumpen
N9:4	y_k	1/2	Ärvärde
N10:1	K	4	Parameter för egen regulator
N10:2	$\tilde{R}_i = K\frac{\tilde{T}_s}{T_i}$	256	Parameter för egen regulator
N10:3	$\tilde{R}_d = K\frac{T_d}{N+\tilde{T}_s}$	1	Parameter för egen regulator
N10:4	$\tilde{R}_N = \frac{N}{N+\tilde{T}_s}$	256	Parameter för egen regulator
N10:5	$e_k = r_k - y_k$	1/2	Reglerfelet
N10:6	$\tilde{R}_i e_m$	1	Tillskottet till integralen vid tidpunkten $m \cdot \tilde{T}_s$
N10:7	I_m	1	Integraldelen
N10:8	D_m	1/4	Derivatan i tidpunkten $m \cdot \tilde{T}_s$
N10:9	Ke_k	1/4	Mellanlagring av P-del
N10:10	$a_m = I_m + D_m$	1/4	Mellanlagring av D- och I-del
N10:11	v_k	1/4	Uträknad styrsignal för pump
N10:12	u_k	1/4	Trunkerad styrsignal för pump
N10:13	y_{m-1}	1/2	Utsignal i tidpunkten $(m-1) \cdot \tilde{T}_s$
N10:14			Fritt för mellanresultat
N10:15			Fritt för mellanresultat
N10:16			Fritt för mellanresultat

(lika med 1) ska tolkas som att motsvarande funktion är aktiv.

Tabell 3.2. Flaggor och heltalsvariabler. Att flaggan är satt (biten är $= 1$*) tolkas som att funktionen är aktiv. Samtliga register i denna tabell är av datatypen I/O discrete.*

Register	**Kommentar**
N7:0/1	Manuellt läge.
N9:5/0	Val av tank som skall regleras.
N9:5/1	Överfyllnadssskydd aktiverat.
N10:0/0	Egna PID-regulatorn inkopplad
N10:0/1	Integratoruppvridningsskydd aktivt

Kapitel 4

Arbetsgång

OBS: Allt som hanterar D-delen av regulatorn gäller endast för kursen TSIU04.

Även om uppgiften lätt kan formuleras som att en PID-regulator med derivatafilter och skydd mot integratoruppvridning ska implementeras, kan det visa sig oväntat svårt att genomföra uppgiften med den enkla hårdvara som används i laborationen.

Beskrivningarna av de olika varianterna av PID-regulatorn kan vid en första anblick kännas omöjliga att hantera, men man har stor hjälp av att bryta ner problemet i hanterbara bitar. Det första naturliga steget är att göra en ”inventering” av problemet och dess delar.

- Är det möjligt att implementera delar av algoritmen?
- I vilken ordning måste saker utföras?
- Vad finns tillgängligt och vad måste beräknas?

Dessa ”regler” kan appliceras på (2.10) och den kan delas upp i beräkning av felet, beräkning av den proportionella delen, beräkning av integraldelen, beräkning av derivatadelen, summering av alla bidrag och trunkering av den beräknade styrsignalen. Dessa delar kan implementeras var för sig och det är därmed också möjligt att testa dem på ett lättare sätt.

För att underlätta problemet är det rekommenderat att först implementera en P-regulator med trunkering.

1. Implementera en P-regulator med trunkering av styrsignalen, dvs. punkt $1, 2, 5, 6$ och 7 i algoritm 1.

2. Utöka regulatorn med integral- och derivatadel som uppdateras långsammare, d v s lägg till punkt 3 och 4 i algoritm 2.

3. Utöka PID-regulatorn med integratoruppvridningsskydd enligt kapitel 2.4, dvs gör ändringarna enligt algoritm 3.

Kapitel 5

Laborationsuppgifter

OBS: Allt som hanterar D-delen av regulatorn gäller endast för kursen TSIU04.

Laborationens huvudsakliga målsättning är att implementera en egen diskret PID-regulator i PLC:n. En ytterligare målsättning är att integrera denna regulator i det grafiska gränssnitt som utvecklades under Lab 1 samt att studera skillnader och likheter mellan dessa regulatorer. Centrala problem och begrepp i denna laboration är skalning, överspill, logiska villkor samt integratoruppvridning.

Laborationen består av förberedelseuppgifter, där ni själva skapar ett färdigt stegkodsprogram på papper innan laborationen. Under den schemalagda tiden ska stegkodsprogrammet implementeras och provköras i PLC:n samt det grafiska gränssnittet uppdateras med några ytterligare funktioner. Som tidigare genomförs laborationen till stor del på egen hand men en laborationshandledare kommer att vara närvarande för att svara på frågor och hjälpa till med praktiska problem.

Utgå från de RSLogix- och InTouch-program som utvecklades under Lab 1 samt förberedelseuppgifterna för att lösa problemen i detta avsnitt.

5.1 Krav på den egna PID-regulatorn

Utgå från stegkodsprogrammet som skapades under Lab 1 med de två utökningarna. Utöver de krav som ställdes i Lab 1, ställer vi nu ett antal ytterligare krav på er implementation av PID-regulatorn. Stegkodsprogrammet skall innehålla följande ytterligare funktionalitet:

1. Det ska vara möjligt att välja mellan den egen-implementerade och den inbyggda regulatorn genom flaggan N10:0/0.
2. Den egen-implementerade regulatorn ska:
   (a) kunna styra antingen den övre eller den undre tanken. Vilken tank som styrs ska kontrolleras av flaggan N9:5/0 på samma sätt som för den inbyggda regulatorn i Lab 1.
   (b) styra pumpen med hjälp av en PID-återkoppling.
   (c) använda en långsam cykel för att uppdatera den integrerande och deriverande återkopplingen samt en snabb cykel för den proportionella återkopplingen.
   (d) hantera överspill i alla multiplikationer.
   (e) trunkera styrsignalen så att den hamnar inom rätt intervall.
   (f) implementera ett skydd mot integratoruppvridning, som kan inaktiveras genom flaggan N10:0/1.
3. **Endast TSIU04**: Den egen-implementerade regulatorn ska:
   (a) Implementera ett derivatafilter på PID-regulatorn för att minska regulatorns bruskänslighet, som kan inaktiveras genom att sätta $N = 0$.

5.2 Krav på det grafiska gränsnittet

Utgå från InTouch-gränsnittet som utvecklades under Lab 1. Under denna laboration ska ni vidareutveckla detta gränssnitt så att det utöver kraven från Lab 1 även uppfyller ett antal ytterligare krav. Det ska vara möjligt att:

4. välja om det är den egen-implementerade regulatorn eller den inbyggda som används för att styra tanken, alltså styra flaggan N10:0/0.
5. följa den egenimplementerade PID-regulatorns I-del i en realtidstrend- och historiktrendkurva. Detta för att senare kunna studera hur väl integratoruppvridningsskyddet fungerar.
6. **Endast TSIU04**: följa den egenimplementerade PID-regulatorns D-del i en realtidstrend- och historiktrendkurva. Detta för att senare kunna studera derivatafiltrets funktion.

7. nollställa I-delen, det vill säga kunna nollsätta N10:7.
8. stänga av skyddet mot integratoruppvridning från processdatorn, alltså styra flaggan N10:0/1.
9. **Endast TSIU04:** stänga av derivatafiltret från operatörsgränssnittet.
10. ställa in regulatorparametrar för den egna regulatorn via gränssnittet. Notera att det är K och T_i som skall ställas in via gränssnittet men att det är K och R_i som lagras i PLC:n. Se även beskrivningen av *data change script* i avsnitt 6.2.
11. **Endast TSIU04:** utöver kravet i uppgiften ovan kunna ställa in T_d och N för den egna regulatorn via gränssnittet. Notera att det är T_d och N som skall ställas in från gränssnittet men det är R_d och R_N som lagras i PLC:n.

5.3 Analys av implementationen

Som avslutning på laborationen ska ni lösa följande uppgifter genom att använda er av stegkodsprogrammet och gränsnittet som utvecklas enligt ovanstående krav.

12. För övre tanken:
    (a) jämför prestanda för den egenimplementerade- och inbyggda PID-regulatorn. Beter de sig på samma sätt?
    (b) utför ett antal stegsvarsexperiment med och utan integratoruppvridningsskydd. Förklara beteendet i de båda fallen.
    (c) på vilket sätt påverkar parametrarna K och T_i hur lång överslängen blir när integratoruppvridningsskydd är inaktiverat?
13. **Endast TSIU04:** För undre tanken, jämför regleringen med och utan D-del.

Kapitel 6

Förberedelseuppgifter

Detta avsnitt innehåller de uppgifter som ska genomföras som förberedelse för laborationen. För att få påbörja laborationen krävs att uppgifterna är genomförda och godkända samt att Lab 1 är godkänd. Svaren på uppgifterna kommer användas under laborationen och examinationen. Tänk därför på hur uppgifterna kopplar till laborationsuppgifterna.

6.1 Integratoruppvridning

Denna laboration behandlar bland annat *integratoruppvridning*, som är ett praktiskt problem som man ofta stöter på när man implementerar en PID-regulator. Läs därför genom materialet från föreläsningar och lektioner samt läs i kurskompendiet för att besvara dessa frågor:

Uppgifter:

1. Läs avsnitt 3.4 i kurskompendiet om integratoruppvridning och olika metoder att hantera detta.
2. Vilken nytta har man av en I-del (integrerande del) i ett reglersystem?
3. När kan integratoruppvridning inträffa, och vilka konsekvensen riskerar man?
4. Förklara hur villkorlig uppdatering fungerar och hur den skyddar mot integratoruppvridning.

6.2 Regulatorparametrar och skalning

Er implementering av PID-regulatorn kommer att använda sig av andra skalfaktorer än den inbyggda för de olika parameterarna (K, T_i och T_d). Vi behöver därför beräkna nya skalningar på ett liknande sätt som i Lab 1. Lägg märke till att vi använder en annan *parametrisering* av PID-regulatorn än den inbyggda. Detta betyder främst att vi har $\tilde{R}_i, \tilde{R}_d$ och $\tilde{R}_N$ som konstanter i ekvation (2.11).

Uppgift:

5. Använd avsnitt 2.3 för att hitta hur man beräknar $\tilde{R}_i$, $\tilde{R}_d$ och $\tilde{R}_N$.

I InTouch vill vi kunna ställa in parametrarna K, T_i och T_d för båda regulatorerna samt N för vår egna. Eftersom vi i implementeringen använder $\tilde{R}_i$, $\tilde{R}_d$ och $\tilde{R}_N$ som beror på K, T_i, T_d och N, måste vi räkna ut $\tilde{R}_i$, $\tilde{R}_d$ och $\tilde{R}_N$ givet dessa parametrar och sedan skicka resultatet till rätt register. Detta kommer vi att lösa genom ett s k *data change script* i In Touch. Ett *data change script* utför en beräkning när en viss *tag* har förändrats i InTouch. Skriptet ser ut precis som i Matlab. Antag att man t ex har kopplat en slider som ska styra förstärkningen till ett *tagname* som heter *Kegen* och på samma sätt skapat *Tiegen* för integreringstiden. Antag vidare att vi har skapat en *tag* som är kopplat till det register i PLC:n som ska lagra $\tilde{R}_i$ och har döpt denna tag till *Riegen*, samt har skalat den på rätt sätt (se nästa förberedelseuppgift). Då kommer vårt skript att bli `Riegen=Kegen*0.5/Tiegen`, eftersom samplingstiden är 0.5 sekunder. Notera att *Kegen* och *Riegen* är kopplade till respektive register i PLC:n som vanligt, dock finns *Tiegen* endast i InTouch.

Uppgifter:

6. Det är fördelaktigt att genomföra denna beräkning i InTouch istället för i stegkodsprogrammet. Vad är anledningen till detta?

7. Beräkna en ny skalning för $\tilde{R}_i$, $\tilde{R}_d$ och $\tilde{R}_N$ som vi ska använda i InTouch. Antag att förstärkningen $K \in [0, 200]$, att integreringstiden $T_i \in [0.1, 10]$ minuter, att deriveringstiden $T_d \in [0, 1]$ minuter och att tidskonstanten för lågpassfiltret på derivatadelen $N \in [0, 30]$ s.

   (a) Vad är det största respektive minsta värdet som $\tilde{R}_i$, $\tilde{R}_d$ och $\tilde{R}_N$ kan anta som oskalad? (Observera att $\tilde{R}_i$, $\tilde{R}_d$ och $\tilde{R}_N$ ska vara dimensionslösa, alltså sakna enheter.)

   (b) Vilket intervall blir detta om man skalar $\tilde{R}_i$, $\tilde{R}_d$ och $\tilde{R}_N$ med den skalfaktor som föreslagits i detta labb-pm?

Tabell 6.1. Skalfaktorer för de variabler som används i programmet. Notera att flaggan RG måste vara satt till 1 för att skalfaktorn 100 skall gälla för K och T_i. Samtliga register i denna tabell är av datatypen I/O real. Notera att skalfaktorerna i flera fall skiljer sig från de som användes i laboration 1.

Reg.	**Uttryck**	**Faktor**	**Kommentar**
I:3.0		1	Utsignal från övre tank
I:3.1		1	Utsignal från undre tank
O:3.0		1	Styrsignal till pump
N7:2	r_k	1/2	Referenssignal
N7:3	K	100	Parameter för inbyggda regulatorn
N7:4	T_i	100	Parameter för inbyggda regulatorn
N7:5	T_d	100	Parameter för inbyggda regulatorn
N9:0	$y_{\text{upper},k}$	1/2	Tanknivå övre tanken
N9:1	$y_{\text{lower},k}$	1/2	Tanknivå undre tanken
N9:2	$u_{\text{pump},k}$	1/2	Styrsignal till pumpen
N9:3	$u_{\text{pump,manuell},k}$	1/2	Manuell styrsignal till pumpen
N9:4	y_k	1/2	Ärvärde
N10:1	K	4	Parameter för egen regulator
N10:2	$\tilde{R}_i = K\frac{\tilde{T}_s}{T_i}$	256	Parameter för egen regulator
N10:3	$\tilde{R}_d = K\frac{T_d}{N+\tilde{T}_s}$	1	Parameter för egen regulator
N10:4	$\tilde{R}_N = \frac{N}{N+\tilde{T}_s}$	256	Parameter för egen regulator
N10:5	$e_k = r_k - y_k$	1/2	Reglerfelet
N10:6	$\tilde{R}_i e_m$	1	Tillskottet till integralen vid tidpunkten $m \cdot \tilde{T}_s$
N10:7	I_m	1	Integraldelen
N10:8	D_m	1/4	Derivatan i tidpunkten $m \cdot \tilde{T}_s$
N10:9	Ke_k	1/4	Mellanlagring av P-del
N10:10	$a_m = I_m + D_m$	1/4	Mellanlagring av D- och I-del
N10:11	v_k	1/4	Uträknad styrsignal för pump
N10:12	u_k	1/4	Trunkerad styrsignal för pump
N10:13	y_{m-1}	1/2	Utsignal i tidpunkten $(m-1) \cdot \tilde{T}_s$
N10:14			Fritt för mellanresultat
N10:15			Fritt för mellanresultat
N10:16			Fritt för mellanresultat

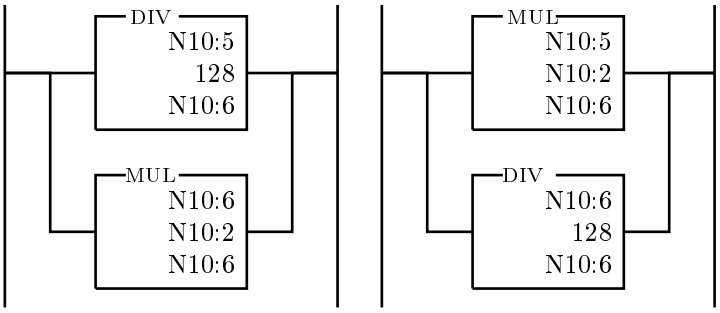
6.3 Heltal och överspill

När man som i denna laboration använder en PLC som endast kan hantera heltal riskerar man att få problem med överspill eftersom stora heltal kommer att multipliceras och adderas. Uppgiften är nu att skapa några stegkodsstrukturer som kommer att hjälpa oss att hantera detta problem på bästa sätt.

Antag att man vill genomföra följande beräkning: multiplicera registren N10:2 och N10:5 samt placera den skalade produkten i N10:6. Skalningen kommer att bli en division med 128 eftersom målregistret har skalfaktor 1 och källregistren har skalfaktorerna 256 respektive 1/2. Detta kan genomföra på två olika sätt: att först multiplicera källregistrer och sedan skala med 128 eller att först skala ett källregister och sedan multiplicera källregistren. Detta kan genomföras i stegkodsprogrammering genom strukturerna som visas i figur 6.1.

Uppgifter:

8. Vad händer i de olika stegen? Vad kan man uppnå med att göra på detta sätt?
9. Om skalningen görs innan multiplikationen måste ett register väljas att skala. Är det lämpligast att välja N10:2 eller N10:5? Motivera.
10. Vad är för- och nackdelarna med de två lösningarna i figur 6.1? Vilket alternativ vill man använda i första hand?



Figur 6.1. Två olika sätt att lösa problemet $N10{:}6 = \frac{N10{:}2 \cdot N10{:}5}{128}$.

En metod för att lösa detta problem på ett sätt som balanserar problemet med överspill och förlust av precision är att använda sig av *minor error bit* S:5/0.

Uppgifter:

11. Hur kan `S:5/0` användas för att avgöra vilket av alternativen i figur 6.1 som är bäst?

12. Använd de två strukturerna i figur 6.1 samt `S:5/0` för att konstruera ett stegkodsprogram som utför multiplikationen med största möjliga precision men som samtidigt undviker överspill. Använd följande steg:

    (a) Utgå från att nollsätta `S:5/0`.

    (b) Prova att utföra det bästa alternativet från ovan.

    (c) Om överspill inträffar, utför det andra alternativet.

13. Skapa ytterligare en byggsten till det framtida programmet genom att förändra registren och skalfaktorerna (se tabell 3.2) i det föregående stegkodsprogrammet för att istället utföra operationen $K \cdot e_k$.

6.4 Programmets struktur

Vi är nu redo att kombinera kunskapen om diskretiserade PID-regulatorer samt överspillshantering i ett stegkodsprogram. Målsättningen är att skapa ett komplett stegkodsprogram som hanterar skalning, överspill och integratoruppvridning genom att i ordning lösa uppgifterna nedan. Utgå från lösningarna till föregående uppgifter, algoritmerna tidigare i detta Lab-PM och stegkodsprogrammet som utvecklades i Lab 1.

Uppgifter:

14. Utgå från stegkodsprogrammet från Lab 1. Var skall den egen-implementerade regulatorn placeras? Notera att det skall vara möjligt att välja vilken regulator som är aktiv.

15. Skapa en enkel P-regulator genom att använda algoritm 2, överspillshanteringen från föregående avsnitt samt kapitel 3.2.

    (a) Bryt ned punkt 1 i kapitel 4 till hanterbara bitar.

    (b) Vad finns tillgängligt och vad behöver beräknas?

    (c) Vilka skalfaktorer blir det efter varje uträkning och stämmer det med registren de skall lagras i?

    (d) Behöver överspill hanteras?

    (e) Skriv ner lösningen som pseudokod.

16. Inför nu en långsam cykel i ert program där I- och D-delen ska placeras, utgå från lösningen i föregående uppgift, ladder-programmet i figur 2.1 och algoritm 2.
    (a) Bryt ned punkt 2 i kapitel 4 till hanterbara bitar.
    (b) Vad finns tillgängligt och vad behöver beräknas?
    (c) Vilka skalfaktorer blir det efter varje uträkning och stämmer det med registren de skall lagras i?
    (d) Behöver överspill hanteras?
    (e) Skriv ner lösningen som pseudokod.
17. Utöka lösningen med villkorlig integration genom att utgå från lösningen i föregående uppgift och algoritm 3. Skriv ner lösningen som pseudokod.
18. **Endast TSIU04:** Utöka lösningen med derivatafilter genom att utgå från lösningen i föregående uppgift. Skriv ner lösningen som pseudokod. Notera att det skall gå att stänga av derivata-filtret genom att sätta $N = 0$.
19. Skapa ett komplett stegkodsprogram genom att utgå från psuedokoden från ovanstående uppgifter.