CODEN: LUTFD2/(TFRT-5427)/1–93/(1990)

Jämförelse mellan en PID-regulator och Novatune

Marco Möhle

Institutionen för Reglerteknik
Tekniska Högskolan i Lund
December 1990

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

Document name
Master Thesis

Date of issue
December 1990

Document Number
CODEN: LUTFD2/(TFRT-5427)/1–93/(1990)

Author(s)
Marco Möhle

Supervisor
L-O Tjerngren and Björn Wittenmark

Sponsoring organisation

Title and subtitle
Comparison between a PID regulator and Novatune. (Jämförelse mellan en PID-regulator och Novatune.)

Abstract

The choice between a PID regulator and an adaptive regulator has been considered. The process to be controlled is assumed to be of first or second order. Guidelines have been produced, telling which regulator to prefer when the dynamics of the process changes in run-time or the process is perturbed by different kinds of noise. A PID regulator based on a dominant pole-placement algorithm has been compared with an adaptive regulator (Novatune) using minimum-variance control. Conclusions are drawn mainly from simulations done in Matlab and Simnon.

The consequences of deadtime variations have been investigated. It was shown that if the deadtime of the process varies too much, the PID regulator is to be preferred. The system response to changes is to be preferred. The system response to changes of the time-constant (s) of the process has also been checked. Discrete-time Nyquist curves have been analysed in order to compare the sensitivity against changes in process dynamics in general, and finally, step-disturbances are introduced to the system as well as white noise, in order to simulate real-time perturbations.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
Swedish

Number of pages
93

Recipient's notes

Security classification

INNEHÅLL

1 INLEDNING

ABB Automation har under ett antal år marknadsfört ett komplett styrsystem kallat ABB Master. I detta styrsystem har man möjlighet att välja mellan två olika typer av regulatorer. Antingen så kan man välja en konventionell PID-regulator, eller så kan man välja en adaptiv regulator.

Detta examensarbete har under sommaren 1990 utförts på ABB Automation i Västerås. Syftet har varit att ta fram riktlinjer för vilken av ovanstående regulatortyp man skall välja, då man endast har begränsad kännedom om processen. Ingående jämförelser mellan en PID-regulator, som designats enligt principen för dominerande polplacering, och en adaptiv regulator har gjorts för ett antal typprocesser. Processerna har valts godtyckligt, men ändå med tanke på egenskaper, som t ex dödtid, hos processer inom processindustrin. Kurvor och diagram har erhållits genom simuleringar i simulerings- och beräkningspaketen Matlab och Simnon, och utförts på en IBM PC.

Rapporten omfattar bland annat en beskrivning av de hjälpmedel som använts i detta exjobb samt allmäna beskrivningar av de teorier som ligger till grund för designen av de båda regulatortyper som testats. Dessutom redovisas hur val av viktiga designparametrar har gått till och hur själva simuleringarna har utförts. Slutligen visas och diskuteras simuleringsresultat och utifrån dessa dras vissa slutsatser.

2 RESULTAT

Examensarbetet har resulterat i riktlinjer för när man skall välja en adaptiv regulator, baserad på minimalvariansreglering, eller en PID-regulator, som har designats med hjälp av dominerande polplacering, då processens dynamik ändras under körning eller då systemet påverkas av olika typer av störningar. Framför allt har undersökningar av dödtidskänsligheten gjorts, vilket visade att den adaptiva regulatorn inte bör användas om dödtidsvariationerna är alltför stora. Men även dynamikändringar i form av ökningar och minskningar av tidskonstanter har testats. Nyquistkurvor, för att kunna kontrollera känsligheten mot dynamikförändringar i form av amplitud- och fasmarginal, har producerats, liksom ett antal diagram som visar utseendet på styr- och utsignal då systemet påverkats på ett eller annat sätt. Simuleringar av mindre omfattning har gjorts då styr- och mätsignaler påverkats av brus.

En mer utförlig genomgång av resultaten finns i kapitel 10, men i stort kan de sammanfattas i fördelar och nackdelar för respektive regulator enligt nedan:

+ PID : fungerar tillfredsställande för de allra flesta av de processtyper jag har testat och även då dynamikändringarna inte är alltför stora. Framför allt dödtidsökningar klarar PID:n bättre än Novatune.

+ Novatune: för processer med enbart reella poler blir adaptionen av regulatorparametrarna i någon mån optimal då tidskonstanterna ändras.

- PID : parametrarna måste justeras var gång processens dynamik ändras.

- Novatune: klarar ej dödtidsökningar och fungerar dåligt då vi har processer som är rent integrerande. Svår att ställa in pga många designparametrar och svårt att välja värden på dessa parametrar.

3 SIMULERINGSPAKETEN

Matlab och Simnon är två beräknings- och simuleringspaket som ofta används av reglertekniker vid design och simulering av reglersystem. Matlab är i första hand ett hjälpmedel vid avancerade matrisberäkningar, medan Simnon är speciellt uvecklat med tanke på simulering av olinjära system.

3.1 Matlab

Matlab är ett mycket kraftfullt programpaket vid matematiska beräkningar där stora matriser är inblandade. När man har system av högre ordning än två, blir det i allmänhet alltför tidsödande att räkna för hand. Det finns olika versioner av Matlab beroende på vilken dator det skall köras på. I mitt exjobb har jag använt mig av PC-Matlab, eftersom jag gör alla beräkningar och simuleringar på en IBM-PC.

Vanligtvis körs Matlab interaktivt, men man har också möjlighet att definiera egna nya funktioner vilka interpreteras vid exekveringen.

Utöver de allra vanligaste kommandona finns det ett antal kommandon grupperade i olika "toolbox"-ar. Till de vanligaste hör Control System Toolbox och System Identification Toolbox. I mitt exjobb har jag använt mig av båda dessa, men även ett antal funktioner speciellt framtagna på ABB Automation för deras produkter. Dessa funktioner kan man gruppera samman till en "inofficiell" toolbox, kallad ABB Automation Toolbox. Funktionerna är skrivna av Per Erik Modén. En lista med förklaringar på de funktioner jag har använt finns i appendix A.

Matlab kan även användas till simuleringar av linjära system, men skall man göra en olinjär simulering bör man helst använda sig av något annat simuleringspaket, t ex Simnon.

3.2
Simnon

Simnon är utvecklat på institutionen för reglerteknik i Lund och används vid simulering av olinjära system. Det finns, liksom Matlab, olika versioner för olika miljöer. Jag har använt mig av den så kallade MS-DOS versionen.

Egentligen omfattar Simnon två språk, ett för att beskriva dynamiska system i diskret eller kontinuerlig tid och ett för att styra simuleringar och grafisk presentation. Man har möjlighet att bygga upp ett reglersystem med hjälp av ett antal programmoduler, där varje modul beskriver en enhet. Man kan t ex ha en modul för regulatorn och en modul för processen. Dessa moduler knyts ihop med hjälp av ett så kallat "connecting system", som i själva verket är ett program där man anger vilka signaler som är gemensamma för olika block, dvs vilka signaler som skall knytas ihop.

Man har möjlighet att sampla olika moduler med olika samplingstid, vilket är speciellt användbart då man skall simulera en adaptiv regulator. Ofta sker ju adapteringen av regulatorparametrarna med en annan frekvens än den regulatorn samplar processen med. Ett litet problem uppstod dock här. Det visade sig att det inte alls gick att sampla med olika hastighet, utan den kortast angivna samplingstiden användes i alla moduler.

Med hjälp av så kallad "macro-kod" definieras slutligen systemet. Man kan göra simuleringar av önskad längd och titta på intressanta signaler i diagram, vars axlar kan graderas godtyckligt.

De moduler som beskriver regulatorn och som gör identifieringen av processen var redan konstruerade då jag började med mitt exjobb. Den Simnon-kod jag har producerat inskränker sig till ett "connecting system", en modul som genererar en speciell typ av brus, en macro-modul och slutligen alla processmoduler (vilket egentligen bara innebär att man anger processen på tillståndsform).

4
NOVATUNE

4.1
Allmänt

Novatune är det allmänt vedertagna namnet på den adaptiva regulator som marknadsförs av ABB Automation. Bland reglertekniker kallas den även för STAR-regulatorn (Self-Tuning Adaptive Regulator) och är designad redan någon gång i slutet på 70-talet. Den sitter numera även i ABB:s kompletta styrsystem kallad ABB Master, närmare bestämt i den del som kallas för MasterPiece. Den är realiserad i form av ett PC-element. En MasterPiece programmeras i ett språk som kallas AMPL (ABB MasterPiece Language). Detta är ett grafiskt funktionsblocksspråk, dvs man kan koppla ihop ett antal funktionsblock genom att rita linjer mellan in- och utgångar. Varje funktionsblock är ett PC-element. Det finns även ett PC-element som realiserar en PID-regulator. Man kan således i Mastern välja vilken typ av regulator man vill skall övervaka och styra en process.

4.2 Parametrar

Novatune är en avancerad regulator som kan vara ganska svår att ställa in, ty det finns ca. 25 parametrar som måste definieras. I detta examensarbete behöver vi dock endast ta hänsyn till några av designparametrarna (vilka i och för sig är ganska många). Här följer en lista på dessa med korta förklaringar och efterföljande kommentarer.

Parameter	Förklaring
NA	Antal parametrar för återkoppling
NB	Antal parametrar för styrvärde
NC	Antal parametrar i framkopplingspolynomet
KD	Dödtiden/samplingstiden + 1
PL	Placerad pol
PN	Styrsignalstraff
INT	Integralverkan
PY	Dödband för identifieringen
MAX	Övre begränsning på styrsignalen
MIN	Undre begränsning på styrsignalen
T_s	Samplingstiden

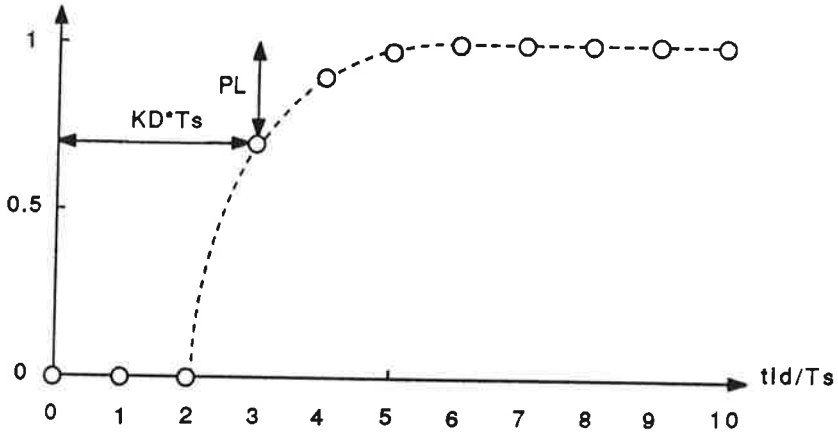
Novatune försöker själv komma fram till rätt modell för processen

$$Ay = Bu$$

men man måste ange ordningstalen på A- och B-polynomen, så att den vet hur många parametrar den skall estimera. Kombinationen NA=2 och NB=3 betyder t ex att 2 parametrar i A-polynomet och 3 parametrar i B-polynomet skall skattas, dvs A och B är båda av andra ordningen.

Om man kan mäta störningar kan dessa bäst elimineras om man använder sig av så kallad framkoppling. I detta exjobb har jag antagit att så inte är fallet och använder således inte framkoppling. Därför är NC=0 i alla simuleringar.

KD bestämmer hur långt framåt i tiden regulatorn tittar. Man kallar därför samplingstiden multiplicerat med KD för prediktionshorisonten. Ett av de allra viktigaste kraven vid val av parametervärden är att prediktionshorisonten är större än dödtiden och att ett stegsvar skall ha gjort ett ordentligt utslag vid den tiden (se figur 1).



Figur 1 Stegsvar för slutna systemet.

PL bestämmer hur stort detta utslag skall vara. Rent teoretiskt sett är PL polen i det önskade slutna systemet. En minskning av PL ger således ett snabbare stegsvar. Jag har satt PL=0.3.

Om styrsignalen svänger upp och ner för varje sampel kan man erhålla en något lugnare reglering, om man sätter PN>0. Dock börjar man alltid inställningen med PN=0 och försöker göra denna så bra som möjligt. I de fall jag har ansett det nödvändigt har jag satt ett värde på PN skiljt ifrån 0.

Integralverkan, dvs INT=1, används normalt i Novatune. Det innebär att vi vid prediktionen tar hänsyn till skillnaden mellan två på varandra följande sampel av mätsignalerna. Fördelen med detta är att vi kan undvika stationära fel vid konstanta börvärden.

Adaptionen av regulatorparametrarna stängs av om inte utsignalen ändras tillräckligt mycket mellan vart sampel, dvs excitationen är för dålig. PY anger gränsen för när man skall anse att så är fallet. Jag har satt PY=0.01.

Amplituden på styrsignalen är begränsad uppåt och nedåt för att undvika alltför kraftiga styrsignaler. I mina simuleringar har jag satt MAX=10 och MIN=-10.

Samplingstiden kräver ingen ytterligare kommentar. Den väljes enligt typkurvorna på sidan 4:59 i installationsguiden för MasterPiece 200/1. Finns det ingen typkurva som passar, väljer man den samplingstid som man tycker ger bäst uppförande (se även under punkten 8.3 - Val av samplingstid).

5
DESIGNMETODERNA

De båda designmetoder som tas upp i denna rapport är dominerande polplacering och självinställande reglering s.k. adaptiv reglering. Den grundläggande idén bakom dominerande polplacering är att man placerar ett antal poler så att det slutna systemet uppför sig på ett sätt som är önskvärt. Övriga poler placeras så att de inte påverkar resten av systemet, dvs de görs så snabba som möjligt. Till den adaptiva regulatorns fördel hör att den själv anpassar sig till förändringar i processen så kallade dynamikändringar, dvs den ställer in sig själv så att regleringen i viss mån blir optimal.

5.1
Dominerande polplacering

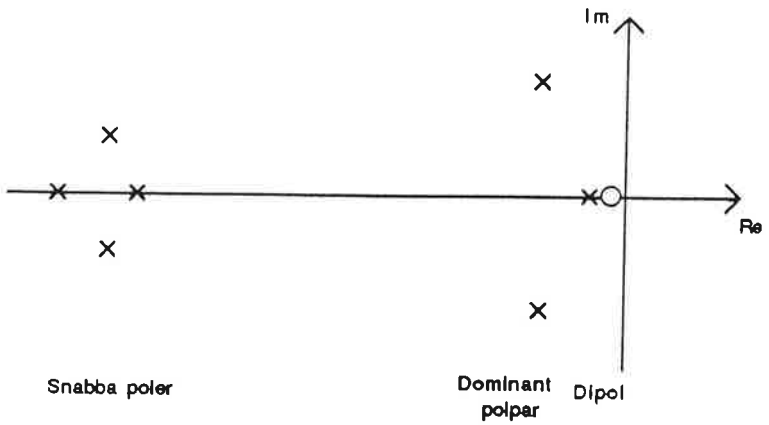
Den variant av dominerande polplacering som jag har utnyttjat i detta examensarbete, bygger på att man vill att det slutna systemet skall uppföra sig som ett andra ordningens system.

$$G(s) = \frac{\omega^2}{s^2 + 2\zeta\omega s + \omega^2}$$

Man placerar då två poler som ett par i närheten av den imaginära axeln på den negativa sidan. De övriga hamnar förhoppningsvis någonstans längs den negativa reella axeln. Det är önskvärt att dessa poler blir så snabba som möjligt, dvs hamnar så långt bort från den imaginära axeln som möjligt. Ju snabbare polerna är desto mindre inverkan har de på utsignalen.

När är ett polpar dominant i förhållande till övriga poler? Ett mått på dominans är kvoten mellan realdelen för den långsammaste av de snabba polerna och realdelen för det dominerande paret. Detta mått kan man kalla för polmarginal.

När man designar en PID-regulator får man ett nollställe i $-1/T_i$. Detta är oftast ett långsamt nollställe, vilket resulterar i stora överslängar om det inte motverkas på något sätt. Vanligast är att man placerar en pol alldeles intill och till vänster om nollstället (se figur 2). Denna pol-nollställeskonfiguration kallas för en dipol.

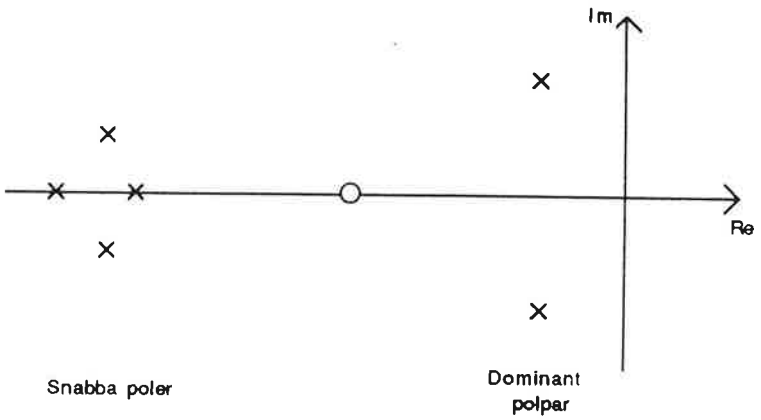


Figur 2 Polkonfiguration med dipol.

Genom att i den proportionella delen av regulatorekvationen multiplicera börvärdet med en skalfaktor β,

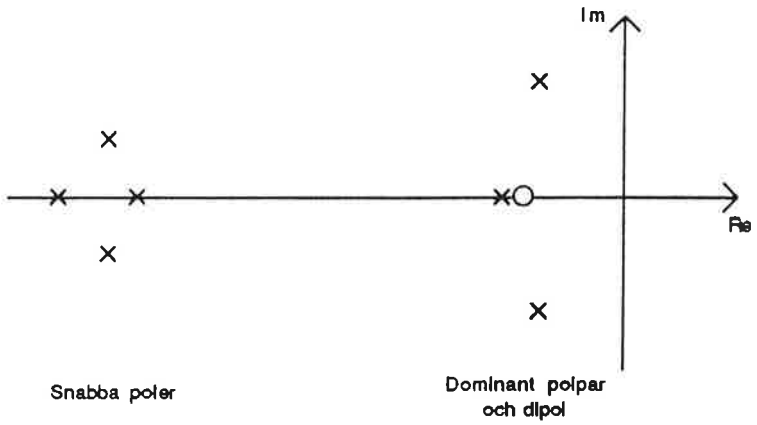
$$u = K(\beta r - y + \frac{1}{T_i s}(r-y) - T_d s y)$$

placerar man nollstället i $-1/\beta T_i$ i stället. Om β är mindre än 1 får vi ett snabbare nollställe. Hägglund och Åström (Automatic tuning of PID controllers based on dominant pole design) rekommenderar att man väljer $\beta=1/(3\sigma T_i)$, där $-\sigma$ är det dominerande polparets realdel. Då erhålles ett nollställe som är 3 gånger så snabbt som det dominerande polparet och har då relativt liten inverkan på ett stegsvar. Således behöver man inte införa en extra pol (se figur 3).



Figur 3 Polkonfiguration utan dipol.

Ett tredje alternativ är att man specificerar 3 poler så att den tredje polen har samma realdel eller samma avstånd till origo som det dominerande polparet. Nollstället placeras sedan i anslutning till denna tredje pol (se figur 4).



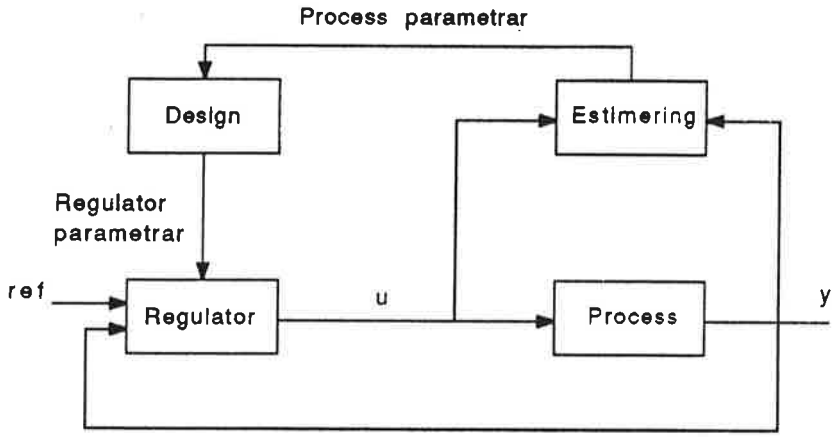
Figur 4 Dipolen något till vänster om polparet.

Först specificerar man ett värde på dämpningsfaktorn ζ i sin andragradsfunktion. Därefter designas en PI- eller PID-regulator enligt en metod för dominerande polplacering, som beskrivs av Åström/Hägglund (Automatic tuning of PID controllers) för kontinuerlig tid. Alla beräkningar utförs dock för diskret tid, sedan polerna, givna av ζ och ω, transformerats. Kriteriet för val av ω är att ω skall vara så stort som möjligt så att dominansen bibehålls. Kravet på dominans översätts från att ligga till vänster om en viss linje i kontinuerlig tid, till att ligga innanför motsvarande cirkel i diskret tid. Problemet är olinjärt i ω och löses därför iterativt, med ökande ω så länge de ansatta polerna blir dominerande, och minskande ω så länge de inte blir det. Stegen i ω minskas successivt till önskad upplösning, här ca 1%.

Denna variant av dominerande polplacering är speciellt intressant därför att man bara behöver ange en designparameter, nämligen ζ. Jag har genomgående valt att sätta ζ=0.7.

5.2 Adaptiv design

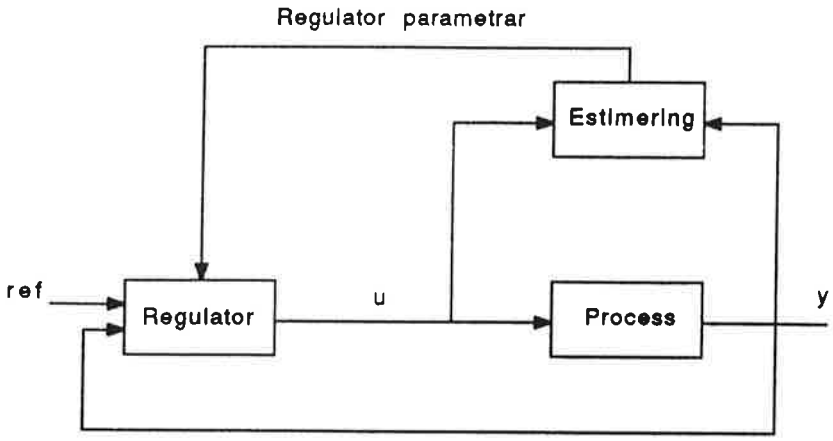
Den typ av adaptiv regulator som ofta används idag beskrevs för första gången av Kalman 1958, men gick då inte att realisera på grund av vissa hårdvaruproblem. Den byggde bland annat på minsta-kvadrat anpassning och deadbeat-reglering. Wieslander/Wittenmark föreslog i början på 70-talet att man skulle använda minmalvarians-reglering i stället, och det är enligt den principen som Novatune är designad. Själva strukturen för en adaptiv regulator kan ses i figur 5.



Figur 5 Blockdiagram för en adaptiv regulator.

I figuren ovan har man således två stycken loopar, dels den vanliga återkopplingen, dels uppdateringen av regulatorparametrarna. Dessa loopar kan ha olika hastighet, med den förstnämnda som den snabbare av de båda.

En regulator av den här typen kallas för en STR-regulator ("Self-Tuning" Regulator). Denna kan delas in i två klasser, indirekt och direkt. I en indirekt STR (se figur 5) sker det först en estimering av processmodellen och därefter uppdateras regulatorparametrarna enligt någon algoritm. En direkt STR däremot estimerar regulatorparametrarna direkt. Novatune är en baserad på en direkt algoritm. Således ser blockstrukturen ut så här i stället:



Figur 6 Blockdiagram för Novatune.

5.2.1
Identifiering

Estimeringsmodulen har till uppgift att försöka göra en identifikation av processen, dvs den skall göra en modell av denna. Modellen uppdateras om processens dynamik ändrar sig.

Den typ av parameterskattning som beskrivs i detta avsnitt kallas för RLS (Recursive Least Square). Själva estimeringen går till som så att ett kommande ärvärde predikteras med hjälp av ett antal tidigare värden enligt

$$\hat{y}(t+k) = \phi^T(t)\ \theta$$

Hur många steg framåt i tiden värdet skall predikteras anges av prediktionshorisonten k. θ är en vektor som innehåller de okända parametrarna och ϕ består av gamla insignaler och styrsignaler.

$$\phi^T(t) = [\phi_1(t)\ \phi_2(t)\ \ldots\ \phi_n(t)]$$

$$\theta = \begin{bmatrix} \theta_1 \\ : \\ \theta_n \end{bmatrix}$$

$\hat{y}$ är det predikterade ärvärdet. Skillnaden mellan detta värde och det verkliga ärvärdet kallas för

prediktionsfelet. Detta fel vill man naturligtvis skall vara så litet som möjligt. Därför gör man en så kallad minsta-kvadrat anpassning, dvs man beräknar summan av kvadraten på alla prediktionsfel och därefter väljer man parametervärden så att denna summa minimeras.

I verkligheten skulle detta innebära att man är tvungen att spara alla gamla data för att identifieringen skall fungera. Därför använder man sig i stället av en rekursiv algoritm,

$$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)\varepsilon(t) \qquad (5.2.1.1)$$

där ε är prediktionsfelet vid tiden t och K är en skalfaktor som bland annat beror på en kovariansmatris P(t), som också uppdateras rekursivt enligt en Riccati-ekvation.

För att inte gamla värden skall inverka lika mycket som de allra senaste, inför man en så kallad glömskefaktor λ. Den är satt till 0.98 vilket innebär att värdenas betydelse avklingar exponentiellt.

Följer man denna rutin erhåller man skattningar på processparametrarna, dvs vi har en indirekt regulator. Eftersom Novatune är en direkt regulator kan man i det här fallet även se dessa parametrar som regulator-parametrar, eftersom styrlagen i Novatune är formulerad så att skattningarna direkt påverkar styrsignalen (se 5.3 - Minamalvariansreglering).

5.3 Minimalvariansreglering

Minimalvariansreglering går, precis som namnet antyder, ut på att man skall försöka minimera utsignalens varians, dvs man vill minimera

$$J_{mv} = E\{ y^2 \}$$

En nackdel med denna form av reglering är att man ofta får stora amplituder på styrsignalen. Detta kan åtgärdas på två sätt:

1. Man kan specificera en pol, så att man får en insvängning enligt en första ordningens överföringsfunktion. Detta åstadkommer man genom att låta modellen prediktera ett filtrerat ärvärde,

$$f_t = A_m * y_t$$

och välja styrvärden så att detta predikterade ärvärde blir lika med börvärdet. I Novatune ser filtret ut på följande sätt:

$$A_m = (1 - PL*q^{-1})$$

Speciellt i Novatune sätts dessutom ärvärdet lika med $(1-PL)r_t$, i stället för bara r_t (r_t är börvärdet), eftersom vi stationärt har att

$$f_{stat} = (1-PL)y_{stat}$$

2. Man lägger på ett straff på styrsignalen, dvs man multiplicerar den egentliga styrsignalen med en faktor. Detta kallas för generaliserad minimalvarians-reglering och presenterades av Clark/Gawthrop i mitten på 70-talet. I Novatune har straffet följande utseende:

$$\frac{b_0^2}{b_0^2 + \rho}$$

Desto större värde på ρ desto mer begränsas styrsignalen. Värdet på ρ bestäms av parametern PN.

$$\rho = \rho_0 * 2^{PN}$$

$$\rho_0 = 4*PY^2/(MAX-MIN)$$

Parametrarna PY, MAX och MIN förklaras under punkten 4.2 - Parametrar.

I Novatune försöker man således i princip minimera

$$J_{mv} = E\{y(t+k)^2 + \rho\Delta u(t)^2 | t\}$$

och detta går till på följande sätt (jag förutsätter här att man har valt INT=1, dvs integralverkan):

Man försöker prediktera ett ärvärde k steg framåt i tiden enligt

$$f_{t+k} = (1 - PLq^{-1})y_{t+k} - (1-PL)y_t \qquad (5.3.1)$$

Eftersom man inte har tillgång till framtida ärvärden sätter man

$$\hat{f}_{t+k} = B(q^{-1})\Delta u_t + A(q^{-1})\Delta y_t \qquad (5.3.2)$$

B- och A-polynomen innehåller okända parametrar. Den sista termen i uttryck 5.3.1 är nödvändig för att ovanstående parametrisering i Δ-variabler skall vara giltig. Vid tiden t har man att

$$\hat{f}_t = B(q^{-1})\Delta u_{t-k} + A(q^{-1})\Delta y_{t-k}$$

Nu kan parametrarna i A- och B-polynomen estimeras genom att man minimerar

$$\Sigma\{(1-PLq^{-1})y_t - (1-PL)y_{t-k} - \hat{f}_t\}^2$$

Detta göres rekursivt enligt RLS-metoden (se 5.2.1 - Identifiering). Hela uttrycket innanför parantesen i summan ovan motsvaras av ε i formel 5.2.1.1. Vi vill nu sätta

$$(1-PLq^{-1})y_{t+k} = (1-PL)r_{t+k} - (\rho/b_0)\Delta u_t$$

där vi således har lagt på ett straff på styrsignalen. b_0 är en av de skattade processparametrarna. Detta ger

$$f_{t+k} = (1-PL)(r_{t+k} - y_t) - (\rho/b_0)(u_t - u_{t-1}) \qquad (5.3.3)$$

Eftersom r(t+k) inte är känt vid tiden t används oftast r(t) i stället. Formel 5.3.2 kan skrivas om på följande sätt

$$\hat{f}_{t+k} = \phi_t^T \theta$$

där θ innehåller de okända parametrarna och φ har följande utseende (för INT=1):

$$\phi_t = \begin{bmatrix} u_t - u_{t-1} \\ \vdots \\ u_{t-NB+1} - u_{t-NB} \\ y_t - y_{t-1} \\ \vdots \\ y_{t-NA+1} - y_{t-NA} \end{bmatrix}$$

Byter man ut f_{t+k} mot $\hat{f}_{t+k}$ i formel 5.3.3, kan man lösa ut u_t och på så sätt erhålla styrlagen

$$u_t = u_{t-1} + \frac{b_0}{b_0^2+\rho} \Big\{ (1-PL)(r_t - y_t) - b_1(u_{t-1} - u_{t-2}) - \ldots - b_{NB-1}(u_{t-NB+1} - u_{t-NB}) - a_0(y_t - y_{t-1}) - \ldots - a_{NA-1}(y_{t-NA+1} - y_{t-NA}) \Big\}$$

En annan nackdel med minimalvariansstyrning är att alla processens nollställen förkortas bort. Detta innebär att det finns risk för "hidden oscillations" i styrsignalen då nollställena är dåligt dämpade, dvs ligger nära, på eller rent av utanför enhetscirkeln. Denna typ av svängning syns inte på ärvärdet i samplingsögonblicken utan endast på styrsignalen, som slår upp och ner i varje sampel. Risken ökar också om man samplar snabbare.

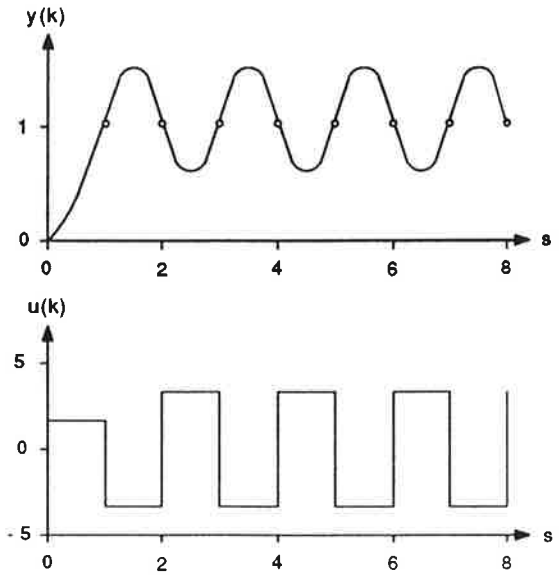
6
MEDELVARIANSBERÄKNING

När man samplar ett kontinuerligt system, kan det uppstå ett fenomen som kallas för "hidden oscillations". Detta innebär att utsignalen ser helt normal ut i samplingsögonblicken, men däremellan svänger upp och ner (ripple). Det finns två anledningar till att sådana svängningar kan förekomma,

1) observerbarheten av det öppna samplade systemet går förlorad på grund av felaktigt vald samplingstid.

2) dåligt dämpade nollställen i det öppna systemet förkortas bort av regulatorn.

Det första fallet kan upptäckas om man kontrollerar observerbarheten hos det samplade systemet. Dock är det ganska liten risk att "hidden oscillations" uppkommer av denna anledning. Vad som krävs är nämligen att man samplar med en oscillativ mod av systemet. Vanligast är i stället att ripple uppkommer på grund av bortförkortning av nollställen. Oscillationerna upptäcks genom att man tittar på styrsignalen vilken, till skillnad från det första fallet, också oscillerar (se figur 7).



Figur 7 Stegsvar och styrsignal.

Anta att man har en stabil process. Kan det uppstå "hidden oscillations" om det kommer in stokastiska störningar? Bengt Lennartsson har presenterat en metod där detta kan kontrolleras genom att jämföra utsignalens varians i samplingsögonblicket med

medelvariansen under en samplingsperiod. Medelvariansen beräknas på följande sätt: variansen i samplings-ögonblicken kan skrivas

$$E\{ y^2(nT_s) \} = \sigma^2$$

Anta att man är intresserad av variansen mellan samplingsögonblicken. Detta kan skrivas som

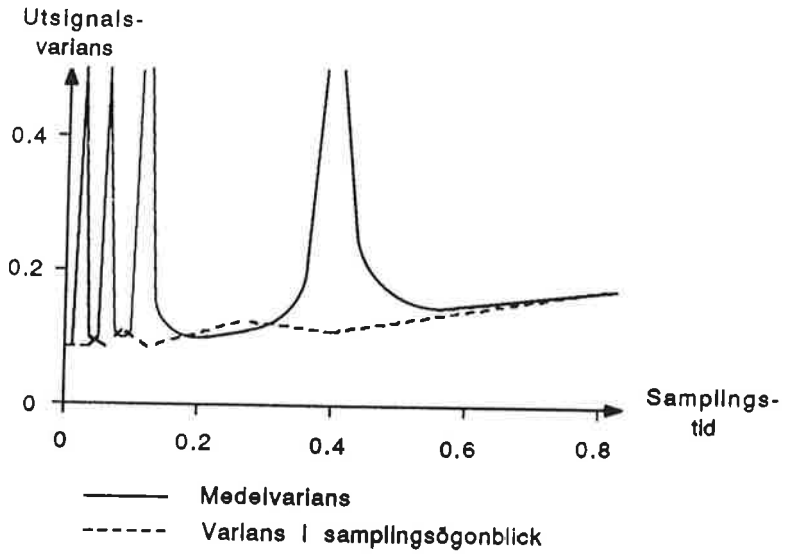
$$E\{ y^2(nT_s+\tau) \} = \sigma^2(\tau)$$

där τ antar värden mellan 0 och T_s. Ur detta kan medelvariansen erhållas genom att beräkna integralen över en samplingsperiod

$$\bar{\sigma}^2 = \frac{1}{T_s} \int \sigma^2(\tau)\, d\tau$$

Eftersom integralen är svår att beräkna, approximerar man ovanstående uttryck med medelvärdet av varianserna i ett antal punkter mellan samplingstillfällena. Det visar sig att medelvariansen ökar drastiskt då processens nollställen närmar sig -1, medan variansen i samplingsögonblicken ligger kvar på en låg nivå. Detta illustreras i figur 8. Eftersom regulatorn kompenserar bort processens nollställen, kan detta ge upphov till instabila moder om nollställena ligger nära eller utanför enhetscirkeln. Medelvariansen skjuter då i höjden. Anledningen till att variansen i samplingsögonblicken är fortsatt låg beror på pol-nollställesförkortningen som verkar i själva samplingsögonblicken. Vi har således en metod för att upptäcka "hidden oscillations" i utsignalen genom att titta på medelvariansens storlek och jämföra den med variansen i samplingstillfället.

Här är en bild på hur medelvariansen kan variera som funktion av samplingstiden i förhållande till variansen i samplingsögonblicket:



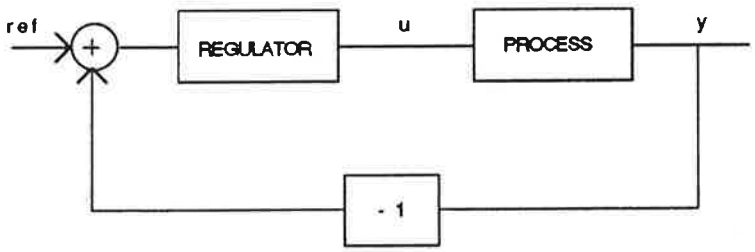
Figur 8 Medelvarians och varians i samplingsögonblick.

7
KORT OM MATLAB-RUTINERNA

Den programkod som har skrivits för detta exjobb består huvudsakligen av Matlab-rutiner, närmare bestämt fyra stycken. Två av dessa, en för PID:n och en för Novatune, ger en grafisk presentation av till exempel ett stegsvar. Dessa har jag kallat dompp.m respektive nova.m. De två övriga (stoc.m och stova.m) beräknar varianser på utsignalen för respektive regulator. Detta för att se om de samplingstider jag har valt ger så kallde "hidden oscillations" i utgsignalen (se kapitel 6 - Medelvariansberäkning).

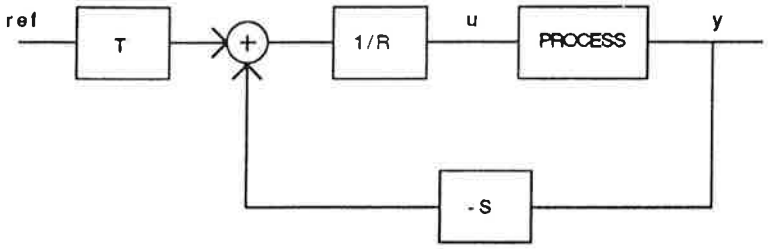
Gemensamt för både dompp.m och nova.m är att man på ett enkelt sätt kan välja hur insignalen till systemet skall se ut, dvs man kan välja mellan ett steg, en impuls eller ingen insignal alls. Dessutom kan man lägga på stegstörningar, antingen som ett steg som kommer in precis före processen, precis efter eller både och. Slutligen kan man på ett enkelt sätt välja hur axlarna skall graderas i de diagram som visar hur utsignalen respektive styrsignalen ser ut för ett givet system med de insignaler man har valt.

Ett reglersystem beskriver man ofta med ett enkelt blockschema:



Figur 9 Slutet system.

Många gånger vill man beskriva regulatorn på SRT-form i stället, och då ser blockschemat ut så här:



Figur 10 Slutet system med regulator på SRT-form.

Scriptet close, som jag har använt i alla mina rutiner för att sluta reglerkretsen, förutsätter att regulatorn är skriven på SRT-form.

dompp.m

Designar en PID-regulator enligt principen för dominerande polplacering. Processen samplas av scriptet procdef, som ger en modell på tillståndsform. Men även överföringsfunktionen erhålles. Själva polplaceringen görs av ett script som heter dompidz, som enligt gällande krav och specifikationer beräknar regulatorparametrarna k, T_i och T_d. Dock kan man inte sluta reglerkretsen med dessa parametrar, utan de måste först konverteras till S-, R- och T-polynom, vilket görs av pid2srt. Kretsen sluts med hjälp av scriptet close. Som resultat erhålles två diagram, ett för utsignalen och ett för styrsignalen. Därefter beräknas och presenteras stegsvarsspecifikationerna stigtid, insvängningstid och översläng.

stoc.m

Beräknar medelvarians och varians i samplingsögonblick för PID-regulatorn. Till den ursprungliga processen adderar man brus. Processen samplas först, av scriptet processd, med en kortare samplingstid än den man har valt vid designen, och därefter med den riktiga, "långsamma" samplingstiden av scriptet mulsamp. Detta görs för att erhålla de kovariansmatriser med vars hjälp scriptet averance beräknar medelvariansen och variansen i samplingsögonblicket. Men innan dess samplas processen en gång till för att de egentliga regulatorparametrarna skall kunna beräknas av dompidz (se under punkten 8.4 - Brusmodellering). Konvertering till S-, R- och T-polynom samt beräkning av det slutna systemet sköts av pid2srt respektive close, precis som tidigare.

nova.m

Designar den adaptiva regulatorn. Processen samplas av scriptet sprocess, som ger en modell på polynomform. Denna modell används av nastar, som lämnar regulatorparametrarna i S, R och T. Därefter sluts reglerkretsen av scriptet close och efter simulering erhålles två diagram, ett för ärvärdet och ett för styrsignalen. Slutligen beräknas stegsvars-specifikationerna stigtid, insvängningstid samt översläng.

Om man antar att man har icke-ideala förhållanden, vilket man ofta har, kan man välja att göra en identifikation av processen. Man har då möjlighet att addera brus, vilket naturligvis innebär att inte exakt rätt modell kommer att hittas. Att göra den här typen av simulering i Matlab är dock inte speciellt meningsfull, eftersom vi i fallet Novatune har ett olinjärt system, utan den bör naturligtvis utföras i Simnon.

stova.m

Beräknar medelvarians och varians i samplingsögonblick för Novatune. Till den ursprungliga processen adderar man brus. Processen samplas först, av scriptet processd, med en kortare samplingstid än den man har valt vid designen, och därefter med den riktiga, "långsamma" samplingstiden av scriptet mulsamp. Regulatorparametrarna beräknas med hjälp av nastar och reglerkretsen sluts som vanligt av close. Averance, slutligen, beräknar medelvariansen och variansen i samplingsögonblicket.

8
ALLMÄNT ANGÅENDE SIMULERINGARNA

8.1
Allmänt

Alla resultat och slutsatser i detta examensarbete har tagits fram genom simuleringar i Matlab och Simnon. Samtliga Matlab-simuleringar har gjorts med hjälp av programmen som finns beskrivna i kapitel 7 och listade i appendix C.

För att kunna göra en korrekt Simnon-simulering behöver man ange designparametrarna NA, NB och KD. Dessa bestämms med hjälp av de typkurvor som finns i installations- och igångkörningsguiden för ABB MasterPiece 200/1, i vilken det ju sitter en STAR-regulator. Dessa kurvor är stegsvar för den öppna processen. Vid varje sådan kurva står det rekommendationer på samplingstiden, NA, NB och KD. Man får således göra en simulering av det öppna systemet och se vilken av typkurvorna som stämmer bäst överens med det stegsvar man erhåller. Parametrarna väljs därefter enligt rekommendationen. I Matlab däremot, använder jag alltid de riktiga värdena på NA, NB och KD, dvs jag antar att man har angivt rätt parametervärden vid initieringen. För vidare diskussion om samplingstider hänvisas till punkten 8.3 - Val av samplingstid.

Rent generellt har alla simuleringar av PID:n gjorts i Matlab, medan Novatune huvudsakligen har testats i Simnon. Den främsta anledningen härtill är att man kan följa adaptionen av regulatorn i Simnon, dvs man kan göra olinjära simuleringar, till skillnad mot Matlab. Matlabsimuleringarna för Novatune visar hur ett stegsvar ser ut när adaptionen är avslutad. Härigenom får man ett mått på hur länge man behöver simulera i Simnon, för att man skall anse att regulatorparametrarna inte uppdateras längre. Ett mera subjektivt sätt att bestämma hur lång simuleringstid man skall välja är att simulera tills man tycker att utsignalen och styrsignalen inte förändras nämnvärt under två på varandra följande börvärdessteg.

8.2
Processerna

Jämförelser har gjorts mellan en PID-regulator och Novatune för ett antal olika typprocesser. Dessa är

$$1.\ G(s) = \frac{K}{1 + sT} e^{-sL}$$

$$2.\ G(s) = \frac{1}{s} * \frac{K}{1 + sT} e^{-sL}$$

$$3.\ G(s) = \frac{K}{(1 + sT_1)(1 + sT_2)} e^{-sL}$$

$$4.\ G(s) = \frac{K}{s^2 + 2\zeta\omega s + \omega^2} e^{-sL}$$

$$5.\ G(s) = \frac{K}{s^2} e^{-sL}$$

Rättvisa jämförelser mellan en PID-regulator och en adaptiv regulator är ganska svåra att göra därför att principen för reglering och parameterinställning skiljer sig så markant. För en PID-regulator ställer man upp en modell av processen, utifrån vilken man sedan beräknar k, T_i och T_d. Skulle sedan processens dynamik ändra sig under körningens gång, eller om man har gjort en felaktig modellering, kommer PID:n inte att uppföra sig så som man hade tänkt sig. Den adaptiva regulatorn är bättre på så sätt att den själv försöker identifiera processen och kan således själv ändra regulatorparametrarna vid en dynamikförändring.

8.3
Val av samplingstid

Ett av de allra största problemen vid regulatordesign är val samplingstid. Det finns inga klart specificerade regler, utan endast rekommendationer. Åström/Wittenmark (Computer-controlled systems) rekommenderar följande; har man en första ordningens process är det rimligt att se till så att man samplar 4 till 10 gånger under stigtiden för det öppna systemets stegsvar.

För ett andra ordningens system bör man se till att produkten ωh ligger mellan 0.1 och 0.5, dvs

$$\omega h \approx 0.1 - 0.5$$

ω är här systemfrekvensen i radianer och h är samplingstiden. Om stigtiden ges av

$$T_r = \frac{e^{(\phi/\tan\phi)}}{\omega}$$

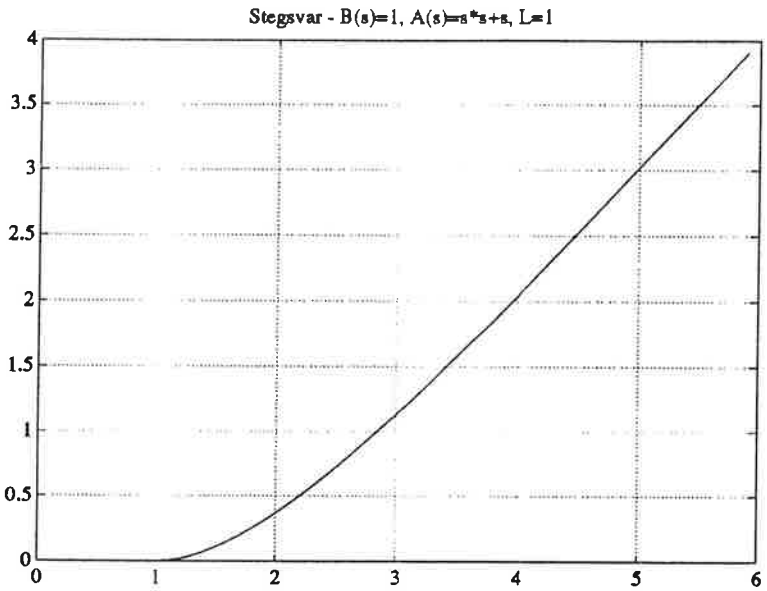
för ett andra ordningens system där

$$\phi = \arccos \zeta \qquad (\zeta = \text{dämpningen})$$

är det bara att ta reda på stigtiden och därefter räkna ut ett värde på samplingstiden h. Observera att det är det slutna systemet som avses.

När jag har valt samplingstider har jag gått till väga på olika sätt beroende på vilken regulator som skall testas. Om vi börjar med PID:n så har jag konsulterat en teknisk rapport av Per Erik Modén, där han har valt samplingstiden som 1/4 av dödtiden. Därefter har jag gjort en simulering med denna samplingstid samt någon lite snabbare, för att konstatera att inte utsignalen är distorderad på grund av för långsam sampling, och slutligen har jag även gjort en kontroll för att se om samplingstiden uppfyller de rekommendationer som Åström/Wittenmark har ställt upp.

Val av samplingstider för Novatune har skett på ett helt annat sätt. I installations- och igångkörnings-guiden för MasterPiece 200/1 finns ett antal typkurvor på sidan 4:59. Dessa visar några olika typer av stegsvar för det öppna systemet. Under var kurva står det rekommendationer på parametrarna NA, NB, KD och samplingstiden. Dessa rekommendationer har följts i alla de fall då detta har varit möjligt. Här saknas nämligen kurvor för processer som innehåller en ren integration. Stegsvaret för en sådan process kan t ex se ut så här:



Figur 11 Stegsvar för en integrerande process.

I sådana fall har jag gjort ett antal simuleringar med olika samplingstider för att kunna avgöra vilken som ger bäst styr- och utsignal. Ibland har det inte gått att finna någon bra samplingstid. Då har jag i stället gjort så att jag har tagit den som är minst dålig och lagt på ett straff på styrsignalen, dvs PN <> 0. Detta innebär att man får en mera dämpad styrsignal som inte oscillerar lika kraftigt.

8.4 Brusmodellering

När man vill undersöka bruskänsligheten måste man ansätta en speciell modell i vilken det ingår brus.

$$Ay = Bu + Ce \qquad (8.4.1)$$

Detta innebär att jag måste ange bruspolynomet C då jag skall designa min regulator. Jag har genomgående valt att sätta C=1.

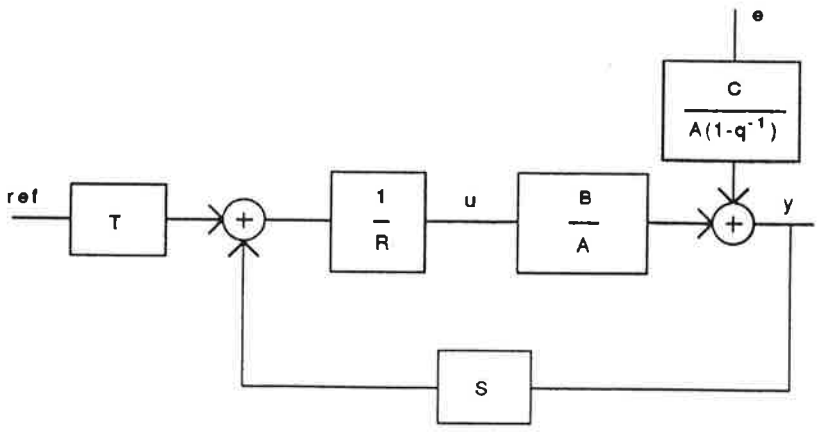
I Novatune sätter man så gott som alltid parametern INT=1, dvs man tar hänsyn till skillnaden mellan två på varandra följande sampel vid prediktionen. Detta ger en regulator med integralverkan, vilket innebär att den modell som används ser ut så här:

$$Ay = Bu + \frac{C}{(1-q^{-1})}e \qquad (8.4.2)$$

Denna ekvation kan skrivas på följande sätt:

$$y = \frac{B}{A}u + \frac{C}{A(1-q^{-1})}e$$

Ett blockschema är mer överskådligt.



Figur 12 Blockschema för Novatune.

Problem uppkommer nu om man vill jämföra utsignalsvarianserna för PID:n och Novatune. Ett krav är ju att bruset som kommer in har samma varians i de båda fallen. Men PID-regulatorn är designad under förutsättning att vi har en modell enligt 8.4.1, vilket innebär att jag inte kommer att få samma varians på

bruset i den punkt där återkopplingen börjar. Dessutom förutsätter programmen för beräkning av utsignalsvariansen en modell enligt 8.4.2.

Detta problem löses till att börja med genom att skriva om formel 8.4.2 så här:

$$A(1-q^{-1})y = B(1-q^{-1})u + Ce$$

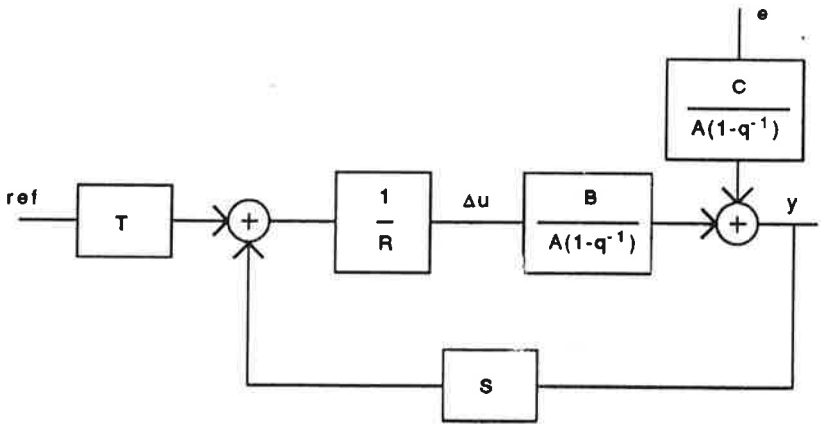
Faktorn framför B kan skrivas om

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

och då får man en modell som ser ut så här:

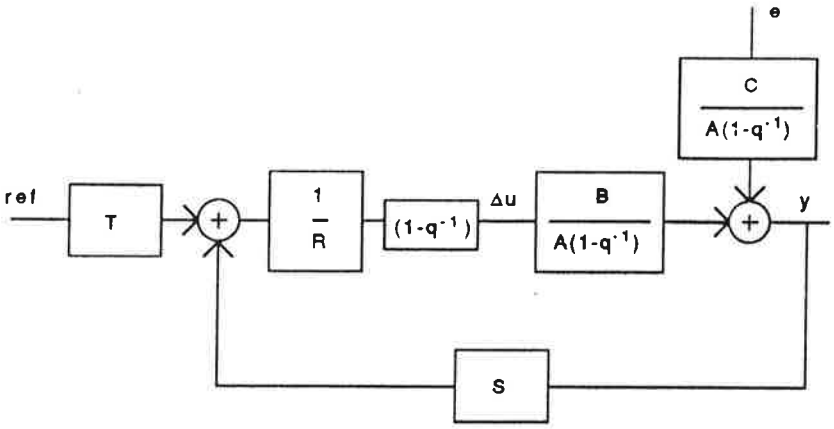
$$y = \frac{B}{A(1-q^{-1})}\Delta u + \frac{C}{A(1-q^{-1})}e$$

Detta ger följande blockschema:



Figur 13 Variant av blockschemat för Novatune.

Här ser vi att det som skickas till processen från regulatorn är Δu i stället för u. Eftersom det redan finns ett antal rutiner i Matlab som gör medelvärdesberäkningar för Novatune, vill vi gärna utnyttja dessa för PID:n också. Därför filtrerar vi utsignalen från PID-regulatorn med $(1-q^{-1})$, vilket ger oss följande blockschema:



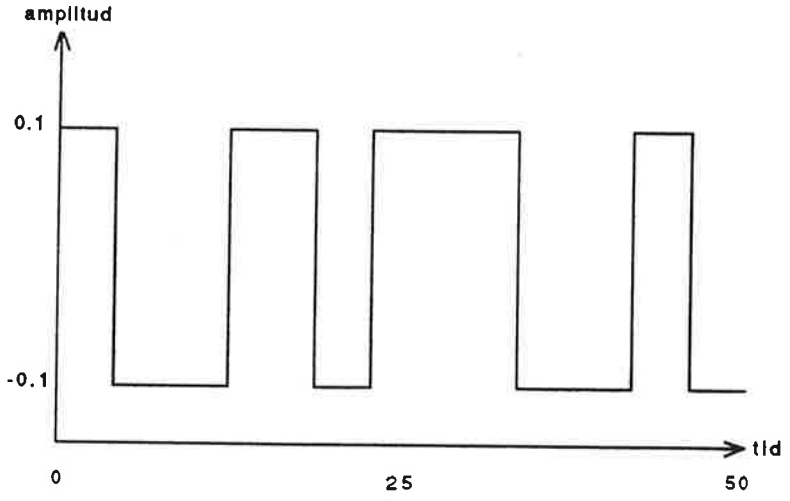
Figur 14 Blockschema för PID:n.

Detta innebär att vi måste sampla processen två gånger. Första gången för att brusmodellen enligt 8.4.2 används, och andra gången för att erhålla de regulatorparametrar vi egentligen skulle ha erhållit, dvs då designen endast baseras på A- och B-polynomen.

8.5 Processbrus

Det kan ofta vara intressant att undersöka hur ett reglersystem uppför sig då inte den signal som regulatorn har skickat iväg når ställdonet, utan det i stället är en störd signal som mottages, dvs vi har slumpmässiga störningar. Man kan då lägga på ett så kallat processbrus på styrsignalen. I detta exjobb har jag använt mig av ett brus med lite speciella egenskaper.

Till skillnad från vitt brus, är det använda bruset ett relativt lågfrekvent brus, som dock kan innehålla högfrekvenskomponenter. Bruset kan bara anta två värden (eller tillstånd). Vilket värde det antar bestäms på ett mer eller mindre slumpmässigt sätt. Det genereras enligt följande: man drar ett värde ur en normalfördelning*, t ex mellan -1 och 1. Om detta värde är (absolut) större än ett visst tröskelvärde, låter man bruset byta tillstånd, dvs bruset byter tillstånd med sannolikhet som man själv kan bestämma. Härigenom erhålles en lågfrekvent signal som tillfälligt kan ha högfrekvent karaktär. Figur 15 visar hur det kan se ut.



Figur 15 Processbrus.

Koden för genereringen återfinns i appendix B.

* Vanligare är kanske att man drar värden ur en rektangelfördelning. Tanken var väl från början att jag skulle använda mig av denna fördelning.

9
SIMULERING OCH ANALYS

9.1
Allmänt

Undersökningarna går i huvudsak ut på att se hur regulatorerna reagerar på olika typer av dynamikförändringar eller störningar, utan att man ändrar på parametrarna k, T_i och T_d för PID-regulatorn eller anger nya designparametrar för Novatune. Men naturligtvis låter jag regulatorparametrarna för Novatune svänga in sig innan jag drar några slutsatser om robustheten.

De prestanda som har undersökts mest noggrant är

A. Dödtidskänslighet. Hur känslig är regulatorn för en ändring av dödtiden utan att regulatorn behöver designas om.

B. Ändring av tidskonstant. Kan PID:n klara en ökning eller minskning och kan Novatune adaptera med acceptabelt resultat.

C. Stegstörningar. Hur snabbt utregleras ett steg som kommer in precis före respektive efter processen.

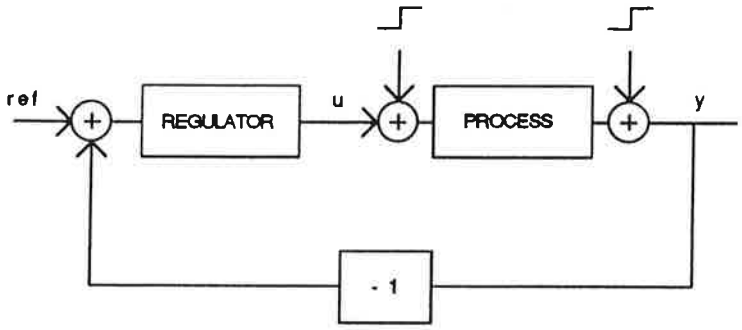
D. Bruskänslighet. Hur stor blir utsignalsvariansen om vi lägger på brus på systemet (se under punkten 8.4 - Brusmodellering). För Novatune kontrolleras även hur bra börvärdesregleringen blir genom simuleringar i Simnon.

Konkret går undersökningarna till på följande sätt:

A. Dödtidskänsligheten - man modellerar både PID:n och Novatune för en viss dödtid. Därefter samplar man processen med en annan dödtid och sluter reglerkretsen med den modellerade regulatorn. Dödtidsändringar både uppåt och nedåt testas.

B. Ändring av tidskonstant - för PID:n designar man regulatorn för en viss tidskonstant. Därefter samplar man processen med en annan tidskonstant och sluter reglerkretsen med den modellerade regulatorn. För Novatune gör man en abrubt, eller i vissa fall stegvis, ändring av tidskonstanten under körning och undersöker stegsvarens utseende under och efter adaptionen.

C. Stegstörning - ett extra steg på styrsignalen respektive utsignalen kan väljas vid simulering i Matlab av både PID:n och Novatune (se figur 16).



Figur 16 Extra steg på styr- respektive utsignal.

Intressant att titta på är hur lång tid det tar innan utsignalen åter är lika med börvärdet och hur stora styringrepp som krävs för att uppnå detta.

D. Bruskänsligheten - som jämförelse mellan PID:n och Novatune beräknas variansen i samplingsögonblicken och medelvariansen. Detta görs i Matlab och bruset är stokastiskt vitt. För Novatune görs även Simnon-simuleringar, där brus i form av kontinuerligt vitt brus läggs på mätsignalen och/eller ett mer lågfrekvent processbrus på styrsignalen.

I appendix D finns en tabell som visar intressanta parametrar, vilka erhållits vid designen av den PI- eller PID-regulator som används på respektive process.

9.2
Process 1

9.2.1
Resultat

Processen är av första ordningen inklusive dödtid. PID:n designas så att det slutna systemet uppför sig som en andra ordningens process. Således kommer det slutna systemets stegsvar att se ut som ett stegsvar för en process av andra ordningen.

Val av samplingstider:

Regulator	Dödtid	T_s
PID	1	0.25
PID	2	0.5
PID	10	2
Novatune	1	0.5
Novatune	2	1
Novatune	10	5

9.2.1.1
Simulering

Vid undersökning av dödtidskänsligheten utgår jag från tidskonstanten T=1 och låter dödtiden variera. Dödtidsminskningar orsakar i allmänhet sällan några problem, så därför betraktas i huvudsak bara ökningar. PID:n klarar en dödtidsökning med en faktor 3 innan systemet blir instabilt, medan Novatune redan förlorar kontrollen vid en liten ökning (se figur B1).

Då jag låter tidskonstanten variera, sker modellering för T=1. Generellt gäller för PID:n att ju kortare processens dödtid är desto mer oscillativt blir systemet då tidskonstanten ökar. Använder vi Novatune i stället, får vi stora amplitudtoppar om tidskonstanten minskar abrupt (se figur B2). Ju större dödtid vi har desto högre blir topparna. Minskar T gradvis i stället för abrubt, blir inte amplituden på topparna lika stor.

Stegstörningar orsakar inga problem för varken PID:n eller Novatune, men dessa simuleringar är gjorda i Matlab och således linjära. Man skulle kunna tänka sig att om man gjorde motsvarande simuleringar i Simnon för Novatune, skulle man få ett annat resultat. Det visade sig att så inte var fallet, men eftersom

regulatorn har adapterats om något, blir de efterföljande börvärdesföljningarna inte optimala så länge regulatorn inte har lyckats hitta en bra modell.

Undersökning av medelvarians och varians i samplingsögonblicken för dödtiderna L=1, L=2 och L=10 gav följande resultat för T=1:

Dödtid	Regulator	Medelvar.	Varians i sampl. punkter
1	PID	7.90	7.94
2	PID	10.71	10.79
10	PID	14.85	15.10
1	Novatune	3.92	4.25
2	Novatune	7.02	7.60
10	Novatune	12.93	14.60

Tabellen visar tydligt att Novatunens utsignalvarians är lägre än PID:ns, då man lägger på brus med samma varians. Vi ser också att för de samplingstider vi har valt har vi inga "hidden oscillations".

9.2.1.2
Analys

Nyquistkurvorna för det öppna sammanslagna systemet, dvs regulator och process, ger information om den allmäna stabiliteten, dvs känsligheten mot dynamikförändringar i processen. I dessa kan man bland annat utläsa fasmarginal, amplitudmarginal samt kortaste avstånd från kurvan till -1. Egentligen säger inte nyquistkurvorna för Novatune inte så mycket här, eftersom man tittar på en speciell inställning av regulatorn. En dynamikändring av processen ger ju en annan inställning och således en annan nyquistkurva. Rent generellt kan man dock säga att värdena på fasmarginalen, amplitudmarginalen och avståndet till -1 inte är kritiska, utan vi har relativt god stabilitets-marginal. Dock finner man vid en mer ingående analys att regulatorn är instabil för dödtiden L=1, dvs regulatorn har poler utanför enhetscirkeln. Stabilitetsmässigt sett är detta kanske inte helt förtroendeingivande. Man skulle kunna sampla med en annan samplingstid eller ange ett annat värde på PL än 0.3, men exakt vad dessa åtgärder har för inverkan på regulatorns poler har ej närmare undersökts.

Dödtid	Regulator	Fas-marginal	Amplitud-marginal	Kortaste avst. till -1
1	PID	61°	2.7	0.6
2	PID	64°	2.9	0.6
10	PID	61°	2.7	0.6
1	Novatune	53°	1.6	0.4
2	Novatune	52°	1.9	0.5
10	Novatune	62°	2.2	0.5

9.2.2
Slutsats

Har man dödtidsvariationer i en första ordningens process, bör man vara säker på att dödtiden inte kommer att vara längre än den dödtid som Novatune är designad för. Om det föreligger en sådan risk skall PID:n väljas. Förekommer andra typer av dynamikändringar, jag tänker då närmast på förändringar av tidskonstanten, kan Novatune med fördel användas, eftersom börvärdesföljningen blir mycket bättre än för PID:n. Om systemen är brusiga får vi dessutom en lägre utsignalsvarians för Novatune, eftersom denna just försöker minimera variansen.

Generellt gäller att om man har en process vars dynamik är känd och inte varierar alltför ofta, väljes PID:n. I annat fall väljes Novatune, men kom ihåg ovanstående kommentarer. En annan fördel med Novatune är att den i regel ger ett snabbare svar på ett börvärdessteg och mindre eller ingen översläng. Detta kan dock bero på inställningen av PID:n, dvs designen av PID-regulatorn behöver inte vara den optimala. Små justeringar av regulatorparametrarna kanske ger ett snabbare stegsvar och/eller mindre översläng.

9.3 Process 2

9.3.1 Resultat

Vi har en integrerande process av andra ordningen. För Novatune finns inga typkurvor till hjälp vid val av samplingstid. Därför har dessa valts så att ett något så när hyfsat uppträdande erhålles.

Val av samplingstider:

Regulator	Dödtid	T_s
PID	1	0.25
PID	2	0.5
PID	10	2
Novatune	1	1
Novatune	2	2
Novatune	10	5

Val av rätt samplingstid har mycket stor betydelse för det slutna systemets stabilitet för Novatune. I fallet då dödtiden L=10 ger t ex samplingstiderna 5 resp. 10 ett stabilt system, medan T_s=7 ger ett instabilt.

Eftersom styrsignalen får ett svängigt utseende då man inte har styrsignalsstraff för dödtiderna L=1 och L=2, har jag valt att sätta PN=14 då L=1 och PN=18 då L=2.

9.3.1.1 Simulering

Vid undersökning av dödtidskänsligheten utgår jag från tidskonstanten T=1 och låter dödtiden variera. Redan vid en dödtidsökning på 2 ggr befinner vi oss på gränsen till instabilitet för PID:n (se figur B3). Novatune klarar av en liten dödtidsökning då modellering skett för L=2, men inte då L=1 eller L=10. Att den råkar klara en ökning för L=2 måste bero på att värdet på PN är sådant att stabiliteten bibehålls just i detta fallet.

Då jag låter tidskonstanten variera, sker modellering för T=1. PID:n klarar inte en ökning av T med mer än en en faktor 4 innan det blir instabilt för dödtiden L=1, medan för L=10 innebär en ökning på 8 ggr inga problem. För Novatune gäller att tidskonstanten inte bör öka mer

än en faktor 3 för samtliga dödtider. Om PN justeras i samband med en ändring av tidskonstanten, är större ökningar möjliga.

En stegstörning som kommer in precis efter processen orsakar en kraftig styrsignal hos framför allt Novatune (se figur B4)

Undersökning av medelvarians och varians i samplingsögonblicken för dödtiderna L=1, L=2 och L=10 gav följande resultat för tidskonstanten T=1:

Dödtid	Regulator	Medelvar.	Varians i sampl. punkter
1	PID	139.6	139.6
2	PID	431.8	431.8
10	PID	6924	6925
1	Novatune	15.88	14.33
2	Novatune	96.58	94.48
10	Novatune	1423	1469

Anledningen till att vi får så stora varianser är att vi har en integrerande process som integrerar upp bruset också. Tabellen visar tydligt att Novatunens utsignal-varians är lägre än PID:ns, då man lägger på brus med samma varians. Vi ser också att för de samplingstider vi har valt har vi lägre varians i samplingsögonblicken än medelvarians för Novatune. Om denna skillnaden är tillräckligt stor har vi ju ripple i utsignalen, men i vårt fall är skillnaden obetydlig.

9.3.1.2
Analys

Dödtid	Regulator	Fas-marginal	Amplitud-marginal	Kortaste avst. till -1
1	PID	35°	2.5	0.5
2	PID	34°	2.2	0.5
10	PID	33°	2.0	0.4
1	Novatune	20°	1.2	0.2
2	Novatune	23°	1.3	0.2
10	Novatune	24°	1.3	0.2

Fasmarginalen har här minskat betydligt för både PID:n och Novatune, men amplitudmarginalen är fortfarande god för PID:n. För Novatune har däremot även amplitudmarginalen minskat så att vi ligger ganska nära stabilitetsgränsen. Detta kan tolkas som att Novatune för den här typen av process är mindre robust än PID:n.

9.3.2
Slutsats

Har man en process av den här typen, dvs en integrerande process, bör man nog försöka undvika att använda Novatune, eftersom den är ganska känslig för dynamikförändringar. Visserligen är det just i sådana fall man normalt har nytta av en adaptiv regulator, men här kan en ofördelaktig förändring lätt orsaka instabilitet. Dessutom är själva regulatorn instabil, dvs regulatorns överföringsfunktion har minst en pol utanför enhetscirkeln.

En annan nackdel med att använda Novatune för den här typen av process är att den kan vara betydligt svårare att ställa in. Speciellt för en servicetekniker som kanske bara har erfarenhet av PID-reglering, där endast K, T_i och T_d behöver justeras. Samplingstiden och PN är ju två oerhört kritiska parametrar om man vill erhålla ett vettigt uppförande av den adaptiva regulatorn för den här typen av process.

9.4 Process 3

Processen är en andra ordningens process med två tidskonstanter T_1 och T_2. Först studeras uppförandet av PID:n och Novatune för $T_1=T_2=1$ och sedan för $T_1=1$, $T_2=4$.

9.4.1 $T_1=1$, $T_2=1$

Följande samplingstider har valts:

Regulator	Dödtid	T_s
PID	1	0.25
PID	2	0.5
PID	10	2
Novatune	1	3.5
Novatune	2	2
Novatune	10	5

9.4.1.1 Simulering

En dödtidsökning leder direkt till instabilitet för Novatune, medan PID:n klarar en dödtidsökning med en faktor 3 (se figur B5).

Då vi undersöker hur en ändring av en tidskonstant påverkar systemet, låter vi T_2 variera medan T_1 genomgående hålles konstant. Om man använder PID:n går en ökning av T_2 med en faktor 8 bra då dödtiden L=1, vilket också innebär att en betydligt större ökning är möjlig då L=10, eftersom en ändring med en faktor 8 här ger ett mindre svängigt svar. För Novatune får vi kraftiga utslag på styrsignalen, men erhåller en snygg utsignal. Utslagen är mindre kraftiga för motsvarande ökning på T_2 då processens dödtid är längre (se figur B6).

Undersökning av medelvarians och varians i samplingsögonblicken för dödtiderna L=1, L=2 och L=10 gav följande resultat:

Dödtid	Regulator	Medelvar.	Varians i sampl. punkter
1	PID	9.30	9.30
2	PID	11.27	11.28
10	PID	15.11	15.16
1	Novatune	18.30	16.53
2	Novatune	8.82	9.22
10	Novatune	12.84	13.84

Brus med samma varians har lagts på då jag har jämfört PID:n med Novatune för en bestämd dödtid. Man ser att utsignalsvariansen är lägre för Novatune än för PID:n bara då dödtiden L=10. Anledningen till detta är de långa samplingstiderna för de kortare dödtiderna. Om vi minskar samplingstiden något men ändå håller oss till rekommendationerna får vi en mindre varians, dock inte lägre än PID:ns. Men å andra sidan får vi också kraftigare, och framför allt svängigare, styringrepp.

9.4.1.2
Analys

Dödtid	Regulator	Fas-marginal	Amplitud-marginal	Kortaste avst. till -1
1	PID	63°	3.1	0.6
2	PID	64°	2.9	0.6
10	PID	61°	2.7	0.6
1	Novatune	70°	2.6	0.6
2	Novatune	68°	2.3	0.6
10	Novatune	62°	2.2	0.5

Båda regulatorerna verkar i princip vara lika robusta.

9.4.2
T_1=1, T_2=4

Följande samplingstider har valts:

Regulator	Dödtid	T_s
PID	1	0.25
PID	2	0.5
PID	10	2
Novatune	1	7
Novatune	2	8
Novatune	10	5

9.4.2.1
Simulering

Novatune klarar ovanligt nog av en dödtidsökning från L=2 till L=4 (se figur B7). Om dödtiden däremot ökar med mer än så blir systemet instabilit, men först efter en relativt lång tid. För PID:n bör dödtiden inte öka med mer än en faktor 2. Speciellt gäller detta då man modellerat för de kortare dödtiderna L=1 och L=2.

När jag simulerar en ändring av tidskonstanten låter jag endast en tidskonstant åt gången få variera. Undersökningar har gjorts för en ändring med en faktor 4 uppåt och nedåt, men endast för dödtiden L=1. Det mest anmärkningsvärda är att om man använder PID:n då T_2 minskar, överlagras stegsvaret av relativt högfrekventa svängningar pga en styrsignal som svänger med samma frekvens (se figur B8).

Undersökning av medelvarians och varians i samplingsögonblicken för dödtiderna L=1, L=2 och L=10 gav följande resultat:

Dödtid	Regulator	Medelvar.	Varians i sampl. punkter
1	PID	3.45	3.45
2	PID	6.37	6.37
10	PID	13.93	13.94
1	Novatune	25.78	24.61
2	Novatune	17.47	16.65
10	Novatune	9.82	10.23

Brus med samma varians har lagts på då jag har jämfört PID:n med Novatune för en bestämd dödtid. Vi ser att variansen är högre för Novatune än för PID:n då L=1 och L=2. Anledningen till detta är återigen den långa samplingstiden. Om vi minskar samplingstiden något men ändå håller oss inom rekommendationsgränserna får vi en

mindre varians, dock inte lägre än PID:ns. Men å andra sidan får vi också kraftigare, och framför allt svängigare, styringrepp.

9.4.2.2
Analys

Dödtid	Regulator	Fas-marginal	Amplitud-marginal	Kortaste avst. till -1
1	PID	50°	2.8	0.6
2	PID	57°	2.8	0.6
10	PID	64°	2.8	0.6
1	Novatune	68°	2.2	0.6
2	Novatune	69°	2.3	0.6
10	Novatune	64°	2.1	0.5

Båda regulatorerna verkar i princip vara lika robusta.

9.4.3
Slutsats

Novatune kan mycket väl användas då man har en andra ordningens process av den här typen, dvs endast reella poler. Speciellt då inte $T_1=T_2$. Den klarar ju då till och med av en liten ökning av dödtiden. Då $T_1<>T_2$ har man nästan inga initialsvängningar, men i gengäld justeras parametrarna under en längre tid, dvs process-identifikationen tar lång tid. Men finns det risk för stora förändringar av dödtiden, bör man nog trots allt använda sig av en PID-regulator. Då tidskonstanterna ändras får man snyggare stegsvar för Novatune än för PID:n. Vid en ökning får man stora överslängar för PID:n, vilket undvikes om man väljer Novatune, men till priset av en svängigare styrsignal. För de kortare dödtiderna är utsignalsvariansen mindre för PID:n än för Novatune. Detta har förmodligen att göra med att vi har haft möjlighet att välja en betydligt kortare samplingstid för PID:n. Samplingstider på 7 och 8 sekunder, vilket erhölls för de korta dödtiderna då $T_1=1$ och $T_2=4$, är inte speciellt bra.

9.5 Process 4

Detta är också en andra ordningens process, men med den skillnaden att vi har komplexa poler i det tidskontinuerliga planet. Tre typfall testas:

1 - ζ=0.3, ω=1
2 - ζ=0.7, ω=1
3 - ζ=0.3, ω=4

Jämförelser mellan PID:n och Novatune görs för varje typfall. Fallet då ζ=0.7 och ω=4 undersöks ej, eftersom detta inte ger någon ny information. Ett fall med stort ζ, litet ω och stort L, motsvarar ett fall med stort ζ, stort ω och litet L, fast med olika tidsskalor.

9.5.1 ζ=0.3, ω=1

Val av samplingstider:

Regulator	Dödtid	T_s
PID	1	0.25
PID	2	0.5
PID	10	2
Novatune	1	3.5
Novatune	2	2
Novatune	10	5

Här har jag frångått principen att välja samplingstid enligt rekommendation för Novatune då dödtiden L=1. Anledningen är helt enkelt de föreslagna dödtiterna ger ett instabilt stegsvar. För de längre dödtiderna duger dock rekommendationerna väl.

För PID:n erhålles negativa värden på T_i och T_d (se appendix D), vilket kanske inte är bra med tanke på att man inte kan ställa in negativa värden. För att undvika att designen ger negativa parametrar bör man specificera ett lägre värde än 0.7 på dämpningsfaktorn ζ, vilket innebär att det slutna systemet får ett svängigare uppförande. Detta bör beaktas då resultaten av simuleringarna studeras.

9.5.1.1
Simulering

För PID:n gäller att en minskning av dödtiden ger ett "tvekande" stegsvar (se figur B9). Tvärtom blir det vid en ökning av dödtiden. Styrsignalen blir kraftigare och vi får ett oscillativt beteende på gränsen till instabilitet vid en ökning av L med en faktor 4. För Novatune leder en liten ökning till instabilitet.

Då jag har simulerat ändringar av processvariabler har följande undersökningar gjorts:

ω minskas till ω=0.5
ω ökas till ω=4
ζ minskas till ζ=0.1
ζ ökas till ζ=0.7

Den mest intressanta iaktagelsen är att den adaptiva regulatorns styrsignal blir mycket svängig då ω minskar (se figur B10). Om ω ökar erhålls en kraftig negativ spik i utsignalen. Ingen spik förekommer om ω ökar gradvis i stället för abrupt. För PID:n gäller att stegsvaret överlagras av en oscillation under insvängningsförloppet då ω ökar. Beträffande ζ är PID:n att föredra om ζ minskar medan Novatune bör väljas om ζ ökar.

Undersökning av medelvarians och varians i samplingsögonblicken för dödtiderna L=1, L=2 och L=10 gav följande resultat:

Dödtid	Regulator	Medelvar.	Varians i sampl. punkter
1	PID	47.37	47.37
2	PID	24.53	24.53
10	PID	16.89	16.91
1	Novatune	44.47	42.02
2	Novatune	22.24	22.49
10	Novatune	14.97	16.47

I alla tre fallen har vi lägre varians för Novatune än för PID:n. Inga "hidden oscillations" förekommer.

En liten underlighet är värd att notera. För dödtiden L=2 är inte utsignalerna identiska i Matlab och Simnon, fastän styrsignalen har samma utseende. Vad detta kan bero på vet jag inte. Mina testresultat påverkas dock inte nämnvärt av detta faktum, därför att både L=1 och L=10 ger ett lugnare system för de samplingstider jag har valt. Således är testerna för dödtiden L=2 en slags "worst case"-tester.

9.5.1.2
Analys

Dödtid	Regulator	Fas-marginal	Amplitud-marginal	Kortaste avst. till -1
1	PID	63°	2.5	0.6
2	PID	63°	2.6	0.6
10	PID	60°	2.8	0.6
1	Novatune	65°	2.6	0.6
2	Novatune	64°	2.6	0.6
10	Novatune	62°	2.3	0.6

Båda regulatorerna verkar i princip vara lika robusta.

9.5.2
ζ=0.7, ω=1

Val av samplingstider:

Regulator	Dödtid	T_s
PID	1	0.25
PID	2	0.5
PID	10	2
Novatune	1	3.5
Novatune	2	2
Novatune	10	5

Även här har jag valt en annan samplingstid än rekommenderat då dödtiden L=1, av samma anledning som ovan. För PID:n blir T_i negativ vilket innebär att de restriktioner och konsekvenser som angavs i föregående fall även gäller här.

9.5.2.1
Simulering

Om dödtiden minskar tar ett börvärdessteg ganska lång tid att reglera ut för PID:n, trots en relativt brant stigning i början och trots att systemet inte oscillerar (se figur B11). Vid en dödtidsökning med en faktor 4 når vi stabilitetsgränsen med PID:n, medan Novatune inte ens klarar av en liten ökning.

Då jag har simulerat ändringar av processvariabler har endast följande undersökningar gjorts:

ω minskas till ω=0.5
ω ökas till ω=4

Om ζ skulle ändra sig uppåt eller nedåt, erhålles samma resultat som då motsvarande tester gjordes för ζ=0.3, ω=1. Då ω ökar genererar PID:n en "hoppig" styrsignal gör att utsignalen svänger in mot sitt korrekta värde med en överlagrad oscillation (se figur B12). Observera att denna simulering är gjord för dödtiden L=1, dvs då vi har längre dödtider framträder liknande oscillationerna först vid en ännu större ökning av ω. Precis som i föregående fall genererar Novatune en mycket svängig styrsignal då ω minskar.

Undersökning av medelvarians och varians i samplingsögonblicken för dödtiderna L=1, L=2 och L=10 gav följande resultat:

Dödtid	Regulator	Medelvar.	Varians i sampl. punkter
1	PID	16.35	16.35
2	PID	15.22	15.22
10	PID	15.54	15.58
1	Novatune	25.49	23.35
2	Novatune	12.54	12.92
10	Novatune	13.61	14.79

Här har Novatune högre utsignalsvarians än PID:n då dödtiden L=1, vilket betyder att Novatune är bruskänsligare än PID:n för denna dödtid och för de samplingstider som valts.

9.5.2.2
Analys

Dödtid	Regulator	Fas-marginal	Amplitud-marginal	Kortaste avst. till -1
1	PID	69°	4.4	0.69
2	PID	65°	3.1	0.64
10	PID	60°	2.7	0.58
1	Novatune	70°	4.0	0.75
2	Novatune	65°	2.5	0.61
10	Novatune	62°	2.2	0.55

Både PID:n och Novatune uppvisar samma tendens beträffande robusthet kontra dödtid. I princip ingen skillnad mellan regulatorerna ur stabilitetssynpunkt.

9.5.3
ζ=0.3, ω=4

Val av samplingstider:

Regulator	Dödtid	T_s
PID	1	0.25
PID	2	0.5
PID	10	2
Novatune	1	0.5
Novatune	2	1
Novatune	10	5

9.5.3.1
Simulering

För PID:n bör inte dödtiden öka med mer än en faktor 3, medan Novatune inte klarar av någon ökning alls. Den klarar faktiskt inte heller en dödtidsminskning med en faktor 2 (se figur B13). I denna simulering sker dock en abrupt ändring av dödtiden så därför har även tester gjorts med en gradvis förändring. Detta ger dock samma resultat. Anledningen till att det blir instabilt är att Novatune misslyckas med att identifiera processen. Den hittar en modell som den tycker verkar bra, men som förmodligen är i icke-minfas. Detta gäller för de kortare dödtiderna. För L=10 resulterar motsvarande minskning inte i instabilitet, men bara om förändringen sker gradvis.

Då jag har simulerat ändringar av processvariabler har följande undersökningar gjorts:

ω minskas till ω=1
ω ökas till ω=8

Novatune genererar återigen en mycket svängig styrsignal då ω minskar, medan man för PID:n erhåller en jämn styrsignal som ger en ganska kraftig översläng i utsignalen (se figur B14). Om ζ skulle ändra sig uppåt eller nedåt, erhålles i princip samma resultat som då motsvarande tester gjordes för ζ=0.3, ω=1 och ζ=0.7, ω=1.

Undersökning av medelvarians och varians i samplingsögonblicken för dödtiderna L=1, L=2 och L=10 gav följande resultat:

Dödtid	Regulator	Medelvar.	Varians i sampl. punkter
1	PID	0.063	0.063
2	PID	0.057	0.057
10	PID	0.059	0.062
1	Novatune	0.047	0.047
2	Novatune	0.049	0.052
10	Novatune	0.055	0.063

I alla tre fallen har vi lägre varians för Novatune än för PID:n.

9.5.3.2
Analys

Dödtid	Regulator	Fas-marginal	Amplitud-marginal	Kortaste avst. till -1
1	PID	60°	2.7	0.6
2	PID	60°	2.9	0.6
10	PID	60°	2.7	0.6
1	Novatune	62°	2.4	0.6
2	Novatune	62°	2.4	0.6
10	Novatune	62°	2.1	0.6

Båda regulatorerna verkar i princip vara lika robusta.

9.5.4
Slutsats

Då vi har ett andra ordningens system med komplexa poler är Novatune speciellt känslig för abrupta dynamikförändringar. Långsammare förändringar av ω ger något bättre resultat. Novatune är som vanligt mycket dödtidskänslig. Då ζ=0.3 och ω=4, klarar den inte en dödtidsminskning med en faktor 2, inte ens om man minskar den gradvis. Till den adaptiva regulatorns fördel talar ett snyggare svar på ett börvärdessteg, speciellt vid ovan nämnda värden på ζ och ω. PID:ns svar blir nämligen lite knyckig då. Generellt gäller att Novatune genererar mycket kraftiga och oscillativa styrsignaler då ω minskar, medan PID:n genererar en mjukare, men dock oscillativ, signal. Då ω ökar uppför sig adaptiva regulatorn väl, medan PID:n är mera "tveksam". Båda regulatorerna klarar av stegstörningar lika bra.

Det är svårt att dra några generella slutsatser vad gäller val av regulatortyp när processen enbart har komplexa poler, så det får avgöras från fall till fall, beroende på processens dynamik och eventuella risker för dynamikändringar, vilken som bör väljas.

9.6
Process 5

9.6.1
Resultat

Processen består endast av en dubbelintegrator och dödtid.

Val av samplingstider:

Regulator	Dödtid	T_s
PID	1	0.25
PID	2	0.5
PID	10	2
Novatune	1	1
Novatune	2	1
Novatune	10	5

9.6.1.1
Simulering

Då man inte har något straff på styrsignalen är denna process omöjlig för Novatune att reglera. För dödtiderna L=1, L=2 och L=10 lyckades jag inte hitta några samplingstider som gav ett system med bra uppförande. Styrsignalen oscillerade med frekvensen 1/samplingstiden , vilket inte hade varit nyttigt för något ställdon. Om man däremot sätter PN till 15 för dödtiderna L=1 och L=2 , och PN till 24 för L=10, erhåller man relativt hyfsade styrsignaler, men en liten översläng i utsignalen, vid börvärdessteg. Steg som kommer in precis efter processen orsakar en relativt svängig styrsignal (se figur B15). En dödtidsökning klarar Novatune (som vanligt) inte av och dessutom är regulatorn mycket bruskänslig. Redan ett brus med variansen 0.05 ger ett mycket dåligt uppförande hos systemet (se figur B17).

PID:n, däremot, lyckas med mycket små styrsignaler följa börvärdessteg, men det tar relativt lång tid innan stationära felet är noll. Endast små dödtidsökningar kan kontrolleras av regulatorn, dock inte en fördubbling. Egentligen har vi en PD-regulator, ty den dominanta polplaceringsalgoritmen ger ett förhållandevis högt värde på T_i. Då stegstörningen kommer in precis före processen får utsignalen en stor amplitudtopp, vilket naturligtvis innebär att systemet

är mycket känsligt för störningar. Steg som kommer in precis efter processen regleras dock bort relativt enkelt, i motsats till Novatune (se figur B18).

9.6.1.2
Analys

Dödtid	Regulator	Fas-marginal	Amplitud-marginal	Kortaste avst. till -1
1	PID	20°	2.0	0.32
2	PID	20°	2.0	0.32
10	PID	20°	1.9	0.31
1	Novatune	9°	1.1	0.08
2	Novatune	4°	1.1	0.04
10	Novatune	4°	1.1	0.04

Mycket små amplitud- och fasmarginaler hos framför allt Novatune, men även hos PID:n. PID:n verkar vara något robustare, men båda regulatortyperna är dock mycket störkänsliga.

9.6.2
Slutsats

Framför allt analysen och införandet av extra steg precis före och efter processen, visar att Novatune nog helst bör undvikas då vi har en process av den här typen, dvs en ren dubbelintegrator inklusive dödtid. Varken PID:n eller Novatune är speciellt robusta, men amplitud- och fasmarginalerna är större för PID:n än i motsvarande fall för Novatune. Dessutom är själva regulatorn instabil i Novatune, dvs regulatorns överföringsfunktion har minst en pol utanför enhetscirkeln.

10
SAMMANFATTNING AV SLUTSATSERNA

Jag har nu jämfört PID:n med Novatune för i allt 8 olika processer, varav 5 processer är av olika typ. Vad man rent generellt kan säga är att PID:n kan väljas i så gott som alla fall, medan man bör testa den adaptiva regulatorn grundligt innan den används. Det är för det första mycket viktigt att man skaffar sig en så bra processmodell som möjligt. Detta för att kunna välja den kanske viktigaste parametern i Novatune, nämligen samplingstiden. Därefter gäller det att man väljer bra värden på övriga parametrar. Detta kan vara ganska besvärligt för en person som inte är helt införstådd med vad parametrarna har för innebörd.

Visar det sig att processen består av en ren integration eller dubbelintegration, bör man nog försöka undvika att använda Novatune. Lika så om man märker att man har stora dödtidsvariationer, framför allt uppåt.

Om processen är en första eller andra ordningens process med enbart reella poler uppför sig Novatune bra vid dynamikförändringar i form av ökad eller minskad tidskonstant. Det kan således vara ett bra val att välja Novatune för att reglera den här typen av process. Men man får dock komma ihåg att långa samplingstider kan ge större värden på utsignalsvariansen hos den adaptiva regulatorn än hos PID:n, trots att Novatune i princip är en minimal-varians regulator.

Består processen enbart av komplexa poler gäller det att se upp med specialfall, om man väljer att implementera Novatune. Se tex 9.5.3.1, där redan små ökningar och minskningar av dödtiden ledde till instabilitet. Vad det gäller övriga typer av dynamikändringar är det svårt att dra några generella slutsatser, men i regel gäller att om ändringarna är av sådan art att vi för PID:n får stora överslängar vid börvärdesändringar, får vi en kraftig, men dämpad, oscillativ styrsignal för Novatune. Vilket som är att föredra beror på storleken av dynamikändringen. Vilken regulator som skall väljas får här avgöras från fall till fall.

11 APPENDIX

11.1 A

avrerance.m

Beräknar medelvarians och varians i samplingsögonblick för det slutna systemet. Om INT=1 ersätts insignalen av insignalsinkrement.

close.m

Beräknar tillståndsmatriserna för det återkopplade systemet, dvs sluter reglerloopen. Inparametrar är den samplade processen samt regulatorparametrarna S,R och T.

dnyquist.m

Beräknar Nyquistkurvan för ett tidsdiskret system.

dompidz.m

Designar en PID-regulator med hjälp av principen för dominerande polplacering. Lämnar parametrarna k, T_i och T_d som resultat.

mulsamp.m

Samplar processen med en multipel m gånger långsammare än vad scriptet processd gör. Ger som resultat den samplade processens överföringsfunktion samt kovariansmatriser som används vid medelvariansberäkningen.

nastar.m

Beräknar S,R och T-polynomen för Novatune.

pid2srt.m

Omvandlar PID-parametrarna k, T_i och T_d till S,R och T-polynom.

procdef.m

Samplar en tidskontinuerlig processmodell inklusive dödtid. Som resultat erhålles både överföringsfunktionen och tillståndsmatriserna för det samplade systemet.

processd.m

Samplar en tidskontinuerlig processmodell inklusive brus. Processen kan innehålla dödtider. Om INT=1 ersätts insignalen av insignalsinkrement. Bruset blir då ett integrerat vitt brus. Förberedelser görs för långsammare sampling.

sprocess.m

Samplar en tidskontinuerlig processmodell inklusive dödtid. Endast överföringsfunktionen erhålles.

starpar.m

Ger de av Novatune identifierade processparametrarna.

11.2
B

```
Discrete system noisep

"Designed by Marco Möhle, AUT/KPU, 900720

output e1

state z
new nz

time t
tsamp ts

Initial
z=0.1

Sort
nz = if abs(norm(t))<limit then z else -z
e1 = nz

ts = t+h
h:8
limit:0.9

end
```

11.3
C

11.3.1
dompp.m

```
% File: dompp.m
%=======================
% Dominant pole placement
%=======================
Bs=[0 1]
As=[1 1]
num=Bs;
den=As;
[A,B,C,D]=tf2ss(Bs,As)
if exist('deadtime')~=1
   deadtime=input('Choose model deadtime: ')
end
if exist('tsamp')~=1
   tsamp=input('Choose tsamp: ')
end
z=0.7
if exist('z')~=1
   z=input('Choose z: ')
end
sampmeth=1
procdef
ks
dompidz
pid2srt
s,r,t

repeat=1;
newspec=input('Sample the true process? (Y/N): ','s');
while repeat==1
   if newspec=='Y' | newspec=='y'
      olddeadtime=deadtime;
      num=input('Numerator: ');
      den=input('Denominator: ');
      deadtime=input('Deadtime: ');
      oldtsamp=tsamp
      tsamp=input('Choose tsamp: ');
      procdef;
      tsamp=oldtsamp
      deadtime=olddeadtime;
   end
   [ac,bc,cc,dc]=close(a1,b1,c1,s,r,t);
   [Bdc,Adc]=ss2tf(ac,bc,cc,dc,1)

   %==================
   % Define plotlimits
   %==================
   if exist('xhigh')~=1
      xhigh=input('Choose xhigh: ');
```

```
end
if exist('ylow')~=1
   ylow=input('Choose ylow: ');
end
if exist('yhigh')~=1
   yhigh=input('Choose yhigh: ');
end
if exist('ulow')~=1
   ulow=input('Choose ulow: ');
end
if exist('uhigh')~=1
   uhigh=input('Choose uhigh: ';;
end

%==================
% Define inputmatrix
%==================
nrofpoints=fix(xhigh/tsamp);
if exist('inputvect')~=1
   inputvect=menu('Choose inputsignal',...
                  'Step',...
                  'Impulse',...
                  'Zero');
end
if inputvect==1
   inputsignal=[ones(nrofpoints+1,1)];
elseif inputvect==2
   nrofzeros=fix(nrofpoints/10)+1;
   rest=nrofpoints+1-(nrofzeros+1);
   inputsignal=[zeros(nrofzeros,1);ones(1,1);zeros(rest,1)];
elseif inputvect==3
   inputsignal=[zeros(nrofpoints+1,1)];
end
nrofzeros=fix(nrofpoints/2);
nrofones=nrofpoints-nrofzeros;
if exist('perturb')~=1
   perturb=menu('Choose perturbation location',...
                'no perturbations',...
                'at process input',...
                'at process output',...
                'at process input & output');
end
if perturb==1
   outputnoise=[zeros(nrofpoints+1,1)];
   processnoise=[zeros(nrofpoints+1,1)];
elseif perturb==2
   outputnoise=[zeros(nrofpoints+1,1)];
   processnoise=[zeros(nrofzeros,1);ones(nrofones+1,1)];
elseif perturb==3
   outputnoise=[zeros(nrofzeros,1);ones(nrofones+1,1)];
   processnoise=[zeros(nrofpoints+1,1)];
elseif perturb==4
   nrofzeros=2*(fix(nrofpoints/3));
   nrofones=nrofpoints-nrofzeros;
   outputnoise=[zeros(nrofzeros,1);ones(nrofones+1,1)];
   nrofzeros=fix(nrofpoints/3);
```

```
   nrofones=nrofpoints-nrofzeros;
   processnoise=[zeros(nrofzeros,1);ones(nrofones+1,1)];
end
inputmatrix=[inputsignal,outputnoise,processnoise];

%=========
% Simulate
%=========
tp=tsamp*[0:nrofpoints];
y=dlsim(ac,bc,cc,dc,inputmatrix);

%===============
% Display result
%===============
subplot(211)
axis([0 xhigh ylow yhigh])
plot(tp,y(:,1))
ylabel('Output')
grid
subplot(212)
axis([0 xhigh ulow uhigh])
[t2,u]=histog(tp,y(:,2));
plot(t2,u)
ylabel('Input')
grid
pause

%===============
% Specifications
%===============
if inputvect==1 & perturb==1
   fprintf('\n')
   overshoot=max(y(:,1))-1;
   lowerindexvec=find(y(:,1)<=0.95);
   upperindexvec=find(y(:,1)>=1.05);
   if Isempty(upperindexvec)
      length_LIV=length(lowerindexvec);
      settlingtime=tsamp*lowerindexvec(length_LIV,1);
   else
      length_LIV=length(lowerindexvec);
      length_UIV=length(upperindexvec);
      index=max([lowerindexvec(length_LIV,1)
upperindexvec(length_UIV,1)]);
      settlingtime=tsamp*index;
   end
   lowerindexvec=find(y(:,1)<=0.1);
   upperindexvec=find(y(:,1)>=0.9);
   if Isempty(upperindexvec)
      risetime=Inf;
   else
      length_LIV=length(lowerindexvec);
      lowerindex=lowerindexvec(length_LIV,1);
      upperindex=upperindexvec(1,1);
      risetime=tsamp*(upperindex-lowerindex);
   end
```

```
      fprintf('Overshoot: %g',overshoot)
      fprintf('\n')
      fprintf('Settlingtime: %g',settlingtime)
      fprintf('\n')
      fprintf('Risetime: %g',risetime)
      fprintf('\n')
   end

   %=========
   % Clean up
   %=========
   answer=input('Erase plotwindow? (Y/N) ','s');
   if answer=='Y' | answer=='y'
      answer=input('Erase axis scales? (Y/N) ','s');
      if answer=='Y' | answer=='y'
         clear xhigh ylow yhigh ulow uhigh
      end
      clg
   end

   newspec=input('Sample the true process? (Y/N): ','s');
   if newspec=='Y' | newspec=='y'
      repeat=1;
   else
      repeat=0;
   end
end
```

11.3.2
stoc.m

```
% File: stoc.m
%=============================================================
% Dominant pole placement from stocastic process model
%=============================================================
Bs=[0 0 1]
As=[1 1.4 1]
inputnum=Bs;
den=As;
noisenum=[0 0 1];
if exist('deadtime')~=1
   deadtime=input('Choose deadtime: ')
end
if exist('tbase')~=1
   tbase=input('Choose tbase: ')
end
z=0.7
if exist('z')~=1
   z=input('Choose z: ')
end
INT=1
kbase=deadtime/tbase
```

```
% Input: inputnum,noisenum,den
processd
% Output: a1,b1,c1,r1
c1_c=c1;

fprintf('Noise variance: %g',r1)
fprintf('\n')

q1;
pv;
m=input('Multiple for sampling: ')
while m>0
   % Input: m,a1,b1,r1,q1,pv,INT,c1
   mulsamp
   % Output: am,bm,as,bs,cs,ks
   as,bs,cs,ks
   bslength=length(bs);
   bstemp=bs(1,find(abs(bs(1,:))>0.00001):length(bs));
   bs=bstemp
   ks=ks+(bslength-length(bstemp))

   am_c=am; bm_c=bm;
   r1m;                          % If zero then cs=[],kg=0,rs=0
   kg;                           % Kalman gain
   rs;                           % Covariance
   sig2s;                        % White noise variance
   qm;                           % qm and pvm are used to obtain the
   pvm;                          % variances averaged over the longer
                                 % sampling periods

   num=Bs;
   tsamp=tbase*m
   procdef;
   % Output: as,bs,cs,ks & a1,b1,c1
   % Input: as,bs,ks,tsamp
   dompidz
   % Output: k,ti,td

   pid2srt
   if INT==1
      rr=deconv(r,[1 -1]);
   else
      rr=r;
   end
   ns=length(s); nr=length(rr); nt=length(t);
   nreg=max([ns nr nt]);
   s=[s zeros(1,nreg-ns)];
   rr=[rr zeros(1,nreg-nr)];
   t=[t zeros(1,nreg-nt)];
   s
   r=rr
   t

   am=am_c; bm=bm_c; c1=c1_c;
   [ac,bc,cc,dc]=close(am,bm,c1,s,r,t);
   averance
```

```
   fprintf('Variance in sampling instances: %g', varyu)
   fprintf('\n')
   fprintf('Average variance: %g',avgvary)
   fprintf('\n')

   m=input('Change multiple for sampling? (0=No / >0=New multipl
end
```

11.3.3
nova.m

```
% File: nova.m
%=========================
% Adaptive control design
%=========================
Bs=[0 0 1]
As=[1 1.4 1]
num=Bs;
den=As;
if exist('deadtime')~=1
   deadtime=input('Choose true deadtime: ')
end
if exist('tsamp')~=1
   tsamp=input('Choose tsamp: ')
end
if exist('PN')~=1
   PN=input('Choose PN: ')
end
%PN=0
[as,bs,cs,ks]=sprocess(num,den,tsamp,deadtime);
cs=1;
as,bs,cs,ks
bslength=length(bs);
bstemp=bs(1,find(abs(bs(1,:))>0.00001):length(bs));
bs=bstemp
ks=ks+(bslength-length(bstemp))
a=as;
b=bs;
kh=ks;

answer=input('Model a perturbed process? (Y/N): ','s');
if answer=='Y' | answer=='y'
   bl=[zeros(1,ks) bs]
   al=[as zeros(1,length(bl)-length(as))]
   u=sign(rand(300,1));
   ul=filter(bl,al,u);
   inclnoise=input('Add noise to the control signal (Y/N): ','s'
   if inclnoise=='Y' | inclnoise=='y'
      e=rand(300,1);
   else
      e=zeros(300,1);
   end
   ef=filter(1,al,e);
   y=filter(bl,al,ul);
   yl=y+ef/4;
```

```
    idNB=length(bs);
    idNA=length(as)-1;
    th=arx([y1 u1],[idNA idNB ks_help]);

    [A,B,C]=polyform(th)
    bs=B(1,find(B(1,:)~=0):length(B))
    as=A
    ks=ks_help
end

PL=0.3;
INT=1;
ro0=4*0.01*0.01/400;

[A,B,C]=starpar(as,bs,cs,0,ks,PL,INT)
NA=length(A), NB=length(B), NC=length(C)

[s,r,t]=nastar(as,bs,cs,ks,PL,INT,ro0,PN)

na=length(a)-1; nb=length(b)-1;
if na+kh>na
   dnum=[zeros(1,kh) b];
   dden=[a zeros(1,nb+kh-na)];
else
   dnum=[zeros(1,kh) b zeros(1,na-nb-kh)];
   dden=a;
end
nsrt=max([length(s) length(r) length(t)]);
s=leftadd(s,zeros(1,nsrt))
r=leftadd(r,zeros(1,nsrt))
t=leftadd(t,zeros(1,nsrt))
[ac,bc,cc,dc]=close(dnum,dden,s,r,t);
[Bdc,Adc]=ss2tf(ac,bc,cc,dc,1)

%==================
% Define plotlimits
%==================
if exist('xhigh')~=1
   xhigh=input('Choose xhigh: ');
end
if exist('ylow')~=1
   ylow=input('Choose ylow: ');
end
if exist('yhigh')~=1
   yhigh=input('Choose yhigh: ');
end
if exist('ulow')~=1
   ulow=input('Choose ulow: ');
end
if exist('uhigh')~=1
   uhigh=input('Choose uhigh: ');
end

%==================
% Define inputmatrix
%==================
```

```
nrofpoints=fix(xhigh/tsamp);
if exist('inputvect')~=1
   inputvect=menu('Choose inputsignal',...
                  'Step',...
                  'Impulse',...
                  'Zero');
end
if inputvect==1
   inputsignal=[ones(nrofpoints+1,1)];
elseif inputvect==2
   nrofzeros=fix(nrofpoints/10)+1;
   rest=nrofpoints+1-(nrofzeros+1);
   inputsignal=[zeros(nrofzeros,1);ones(1,1);zeros(rest,1)];
elseif inputvect==3
   inputsignal=[zeros(nrofpoints+1,1)];
end
nrofzeros=fix(nrofpoints/2);
nrofones=nrofpoints-nrofzeros;
if exist('perturb')~=1
   perturb=menu('Choose perturbation location',...
                'no perturbations',...
                'at process input',...
                'at process output',...
                'at process input & output');
end
if perturb==1
   outputnoise=[zeros(nrofpoints+1,1)];
   processnoise=[zeros(nrofpoints+1,1)];
elseif perturb==2
   outputnoise=[zeros(nrofpoints+1,1)];
   processnoise=[zeros(nrofzeros,1);ones(nrofones+1,1)];
elseif perturb==3
   outputnoise=[zeros(nrofzeros,1);ones(nrofones+1,1)];
   processnoise=[zeros(nrofpoints+1,1)];
elseif perturb==4
   nrofzeros=2*(fix(nrofpoints/3));
   nrofones=nrofpoints-nrofzeros;
   outputnoise=[zeros(nrofzeros,1);ones(nrofones+1,1)];
   nrofzeros=fix(nrofpoints/3);
   nrofones=nrofpoints-nrofzeros;
   processnoise=[zeros(nrofzeros,1);ones(nrofones+1,1)];
end
inputmatrix=[inputsignal,outputnoise,processnoise];

%=========
% Simulate
%=========
tp=tsamp*[0:nrofpoints];
y=dlsim(ac,bc,cc,dc,inputmatrix);

%===============
% Display result
%===============
subplot(211)
axis([0 xhigh ylow yhigh])
plot(tp,y(:,1),'o')
```

```
ylabel('Output')
grid
subplot(212)
axis([0 xhigh ulow uhigh])
[t2,u]=histog(tp,y(:,2));
plot(t2,u)
ylabel('Input')
grid
pause

%===============
% Specifications
%===============
if inputvect==1 & perturb==1
   fprintf('\n')
   overshoot=max(y(:,1))-1;
   lowerindexvec=find(y(:,1)<=0.95);
   upperindexvec=find(y(:,1)>=1.05);
   if Isempty(upperindexvec)
      length_LIV=length(lowerindexvec);
      settlingtime=tsamp*lowerindexvec(length_LIV,1);
   else
      length_LIV=length(lowerindexvec);
      length_UIV=length(upperindexvec);
      index=max([lowerindexvec(length_LIV,1) upperindexvec(lengt
      settlingtime=tsamp*index;
   end
   lowerindexvec=find(y(:,1)<=0.1);
   upperindexvec=find(y(:,1)>=0.9);
   if Isempty(upperindexvec)
      risetime=Inf;
   else
      length_LIV=length(lowerindexvec);
      lowerindex=lowerindexvec(length_LIV,1);
      upperindex=upperindexvec(1,1);
      risetime=tsamp*(upperindex-lowerindex);
   end

   fprintf('Overshoot: %g',overshoot)
   fprintf('\n')
   fprintf('Settlingtime: %g',settlingtime)
   fprintf('\n')
   fprintf('Risetime: %g',risetime)
   fprintf('\n')
end

%=========
% Clean up
%=========
answer=input('Erase plotwindow? (Y/N) ','s');
if answer=='Y' | answer=='y'
   answer=input('Erase axis scales? (Y/N) ','s');
   if answer=='Y' | answer=='y'
      clear xhigh ylow yhigh ulow uhigh
```

```
   end
   clg
end
```

11.3.4
stova.m

```
% File: stova.m
%=======================================
% Adaptive control/stocastic processes
%=======================================
Bs=[0 0 1]
As=[1 2 1]
inputnum=Bs;
den=As;
noisenum=[0 0 1];
if exist('deadtime')~=1
   deadtime=input('Choose deadtime: ')
end
if exist('tbase')~=1
   tbase=input('Choose tbase: ')
end
if exist('PN')~=1
   PN=input('Choose PN: ');
end
%PN=0
INT=1
kbase=deadtime/tbase

% Input: inputnum,noisenum,den
processd
% Output: a1,b1,c1,r1

fprintf('Noise variance: %g',r1)
fprintf('\n')

PL=0.3;
ro0=4*0.01*0.01/400;

m = input('Multiple for sampling: ')
while m>0
   % Input: m,a1,b1,r1,q1,pv,INT
   mulsamp
   as,bs,cs,ks
   bslength=length(bs);
   bstemp=bs(1,find(abs(bs(1,:))>0.00001):length(bs));
   bs=bstemp
   ks=ks+(bslength-length(bstemp))

   [s,r,t]=nastar(as,bs,cs,ks,PL,INT,ro0,PN)
   if INT==1
      rr=deconv(r,[1 -1]);
   else
      rr=r;
   end
```

```
ns=length(s); nr=length(rr); nt=length(t);
nreg=max([ns nr nt]);
s=[s zeros(1,nreg-ns)];
rr=[rr zeros(1,nreg-nr)];
t=[t zeros(1,nreg-nt)];
r=rr;

[ac,bc,cc,dc]=close(am,bm,c1,s,r,t);
averance

fprintf('Variance in sampling instances: %g',varyu)
fprintf('\n')
fprintf('Average variance: %g',avgvary)
fprintf('\n')

m = input('Change multiple for sampling? (0=No / >0=New multi
end
```

11.4
D

Denna tabell visar de dominerande polernas snabbhet ω samt vilka regulatorparametrar som erhållits vid designen av PID- eller PI-regulatorn av scriptet dompidz.

Process nr.	L (s)	T (s)	ω (rad)	k	T_i	T_d	antal dom. poler
T=1:							
1	1	0.25	1.322	0.569	1.076	0.147	3
1	2	0.5	0.737	0.294	1.205	0.172	3
1	10	2	0.140	0.126	2.257	0	2
T=1:							
2	1	0.25	0.661	0.516	6.196	0.784	3
2	2	0.5	0.402	0.271	10.371	0.944	3
2	10	2	0.108	0.067	39.351	2.881	3
T_1=1,T_2=1:							
3	1	0.25	0.987	0.729	1.847	0.453	3
3	2	0.5	0.675	0.414	1.951	0.507	3
3	10	2	0.147	0.156	2.920	0	2
T_1=1,T_2=4:							
3	1	0.25	0.764	2.108	3.709	0.683	3
3	2	0.5	0.504	1.066	4.414	0.778	3
3	10	2	0.155	0.293	5.980	1.161	3
ζ=0.3,ω=1:							
4	1	0.25	0.866	-0.186	-1.141	-0.59	3
4	2	0.5	0.865	-0.038	-0.238	0.31	3
4	10	2	0.121	0.051	0.958	0	2
ζ=0.7,ω=1:							
4	1	0.25	0.944	0.329	1.128	0.702	3
4	2	0.5	0.712	0.215	1.148	0.518	3
4	10	2	0.135	0.117	2.146	0	2
ζ=0.3,ω=4:							
4	1	0.25	1.154	0.034	0.071	0	2
4	2	0.5	0.555	0.018	0.075	0	2
4	10	2	0.136	0.113	2.015	0	2
5	1	0.25	0.393	0.098	10.093	4.278	3
5	2	0.5	0.196	0.026	20.187	8.555	3
5	10	2	0.041	0.001	97.266	41.096	3

12
REFERENSER

Torkel Glad/Lennart Ljung
Reglerteknik - grundläggande teori.
Studentlitteratur, Lund 1981.

Tore Hägglund/Karl J. Åström
Automatic tuning of PID controllers based on dominant pole design.
IFAC Workshop of Adaptive Control of Chemical Processes, Frankfurt am Main 1985.

Karl J. Åström/Tore Hägglund
Automatic tuning of PID controllers.
Instrument Society of America, 1988.

R.E. Kalman
Design of a self-optimizing control system.
Trans. ASME 80: 468-478, 1958.

Björn Wittenmark/J. Wieslander
An approach to adaptive control using real time identification.
Automatica 7: 211-217, 1971.

D.W. Clark/P.J. Gawthrop
A self-tuning controller.
IEE Proc. 122: 929-934, 1975.

Karl J. Åström/Björn Wittenmark
Computer-controlled systems.
Prentice-Hall International,Inc. 1990.

Karl J. Åström/Björn Wittenmark
Adaptive Control.
Addison-Wesley Publishing Company, 1989.

Bengt Lennartson
On the design of stochastic control systems with multirate sampling.
Avhandling No 161, Chalmers Tekniska Högskola, Göteborg 1986.

Per Erik Modén
Dominant pole placement as automatic design method for PI and PID controllers.
Teknisk rapport: TR AUT 88-32, ABB Automation, 1988.

Per Erik Modén
Den adaptiva regulatorn i Novatune.
KP 7364/3, ABB Automation, 1990.

ABB Masterpiece 200/1
Installation, igångkörning och service.
Art.nr. 7650 043-104, ABB Automation, 1989.

Hilding Elmquist/Karl J Åström/Tomas Schönthal
SIMNON - User's guide for MS-DOS computers.
Studentlitteratur, Lund 1986.

Författare-Author Från-From Datum-Date **1990-10-14**

Godkännare-Approved by

Uppdragsgivare-Requested by

Titel-Title

Ordernr-Ref.No.

Debiteras ordernr

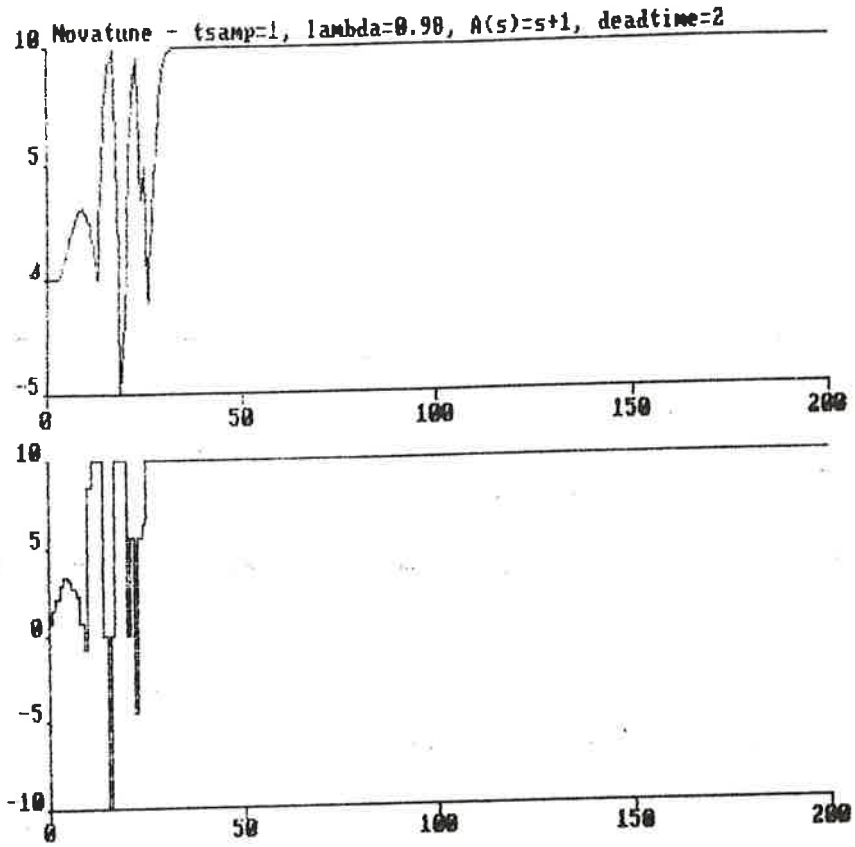
Pkl/Akl

Bilagesidor-Suppl.pages

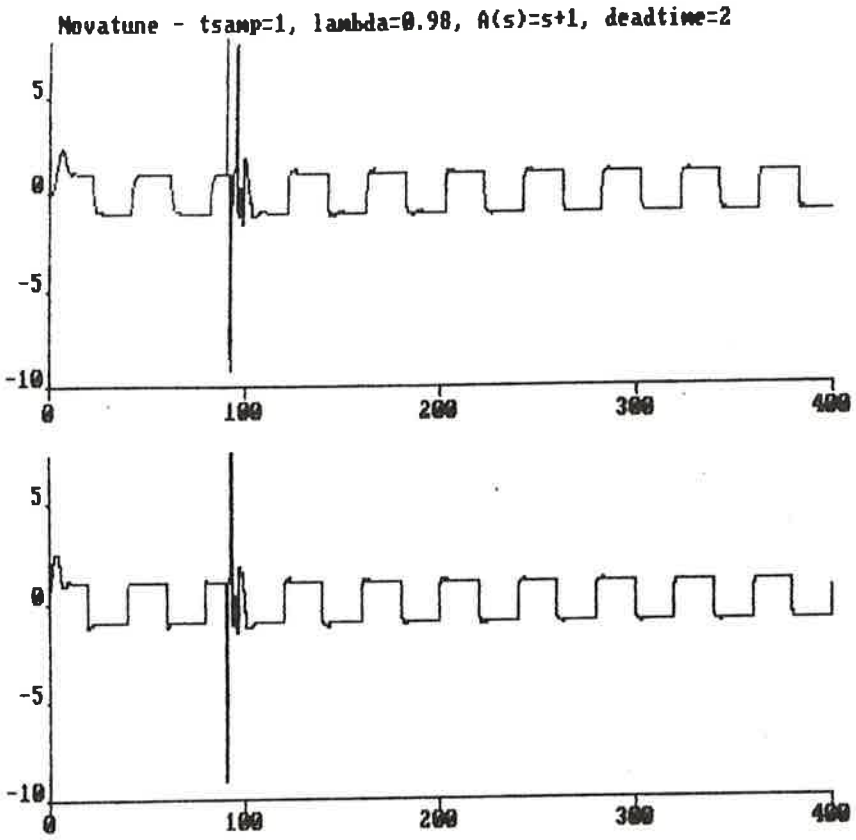
Sammanfattning-Summary

BILAGA

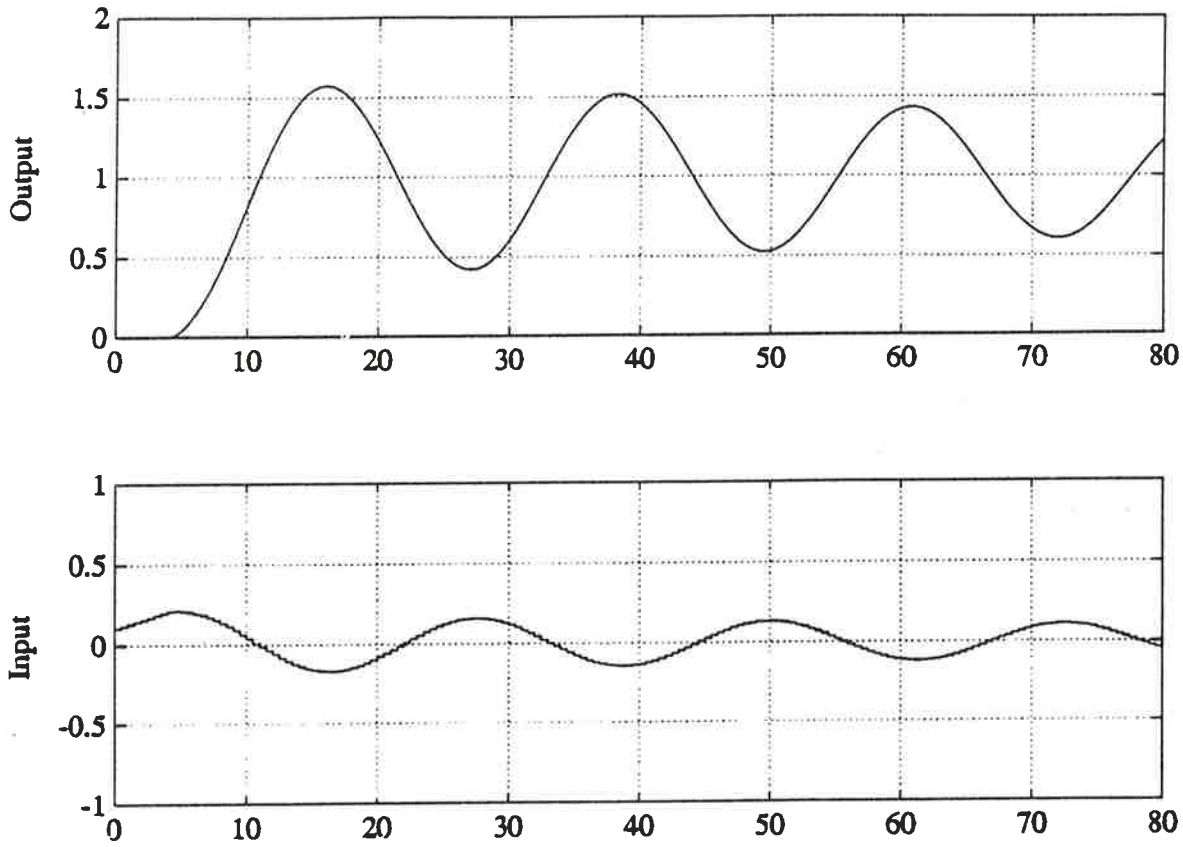
Simuleringsresultat för PID:n och Novatune. Figurerna är numrerade i den ordning de förekommer i rapporten. Detta är en delmängd av alla de diagram som har producerats. Slutsatserna i rapporten har baserats på ytterligare ca. 150 kurvor, men dessa redovisas inte pga omfånget.



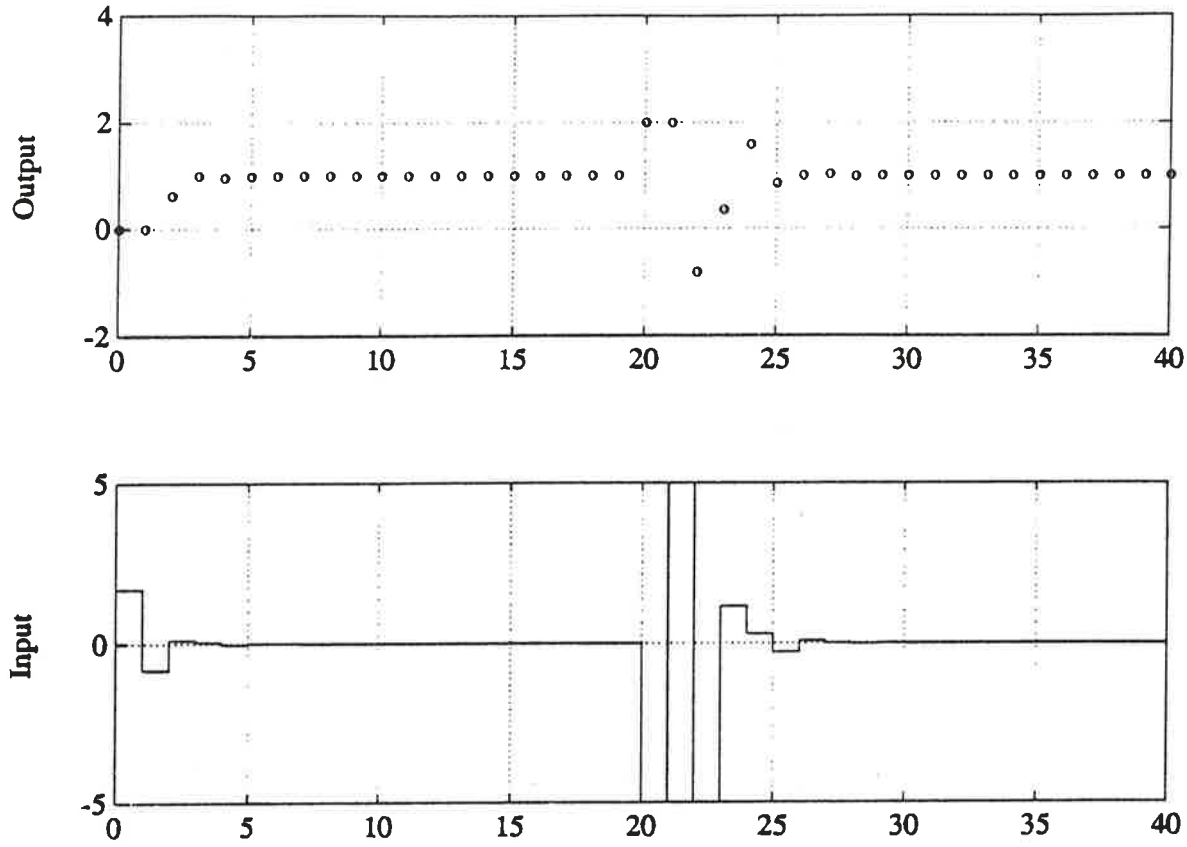
Figur B1 Övre diagrammet: utsignal,
nedre " : styrsignal,
verklig dödtid=3



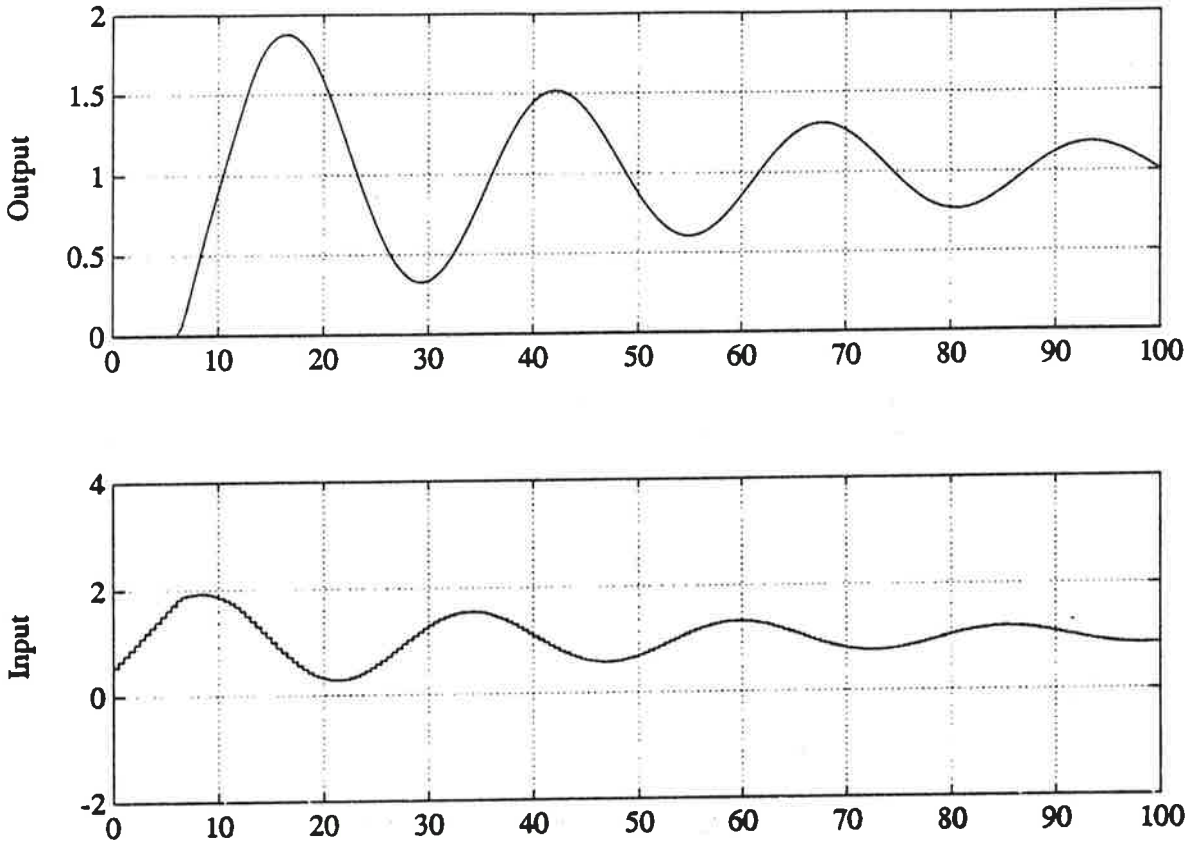
Figur B2 Övre diagrammet: utsignal,
nedre " : styrsignal,
modellerat T=1, verkligt T=0.125



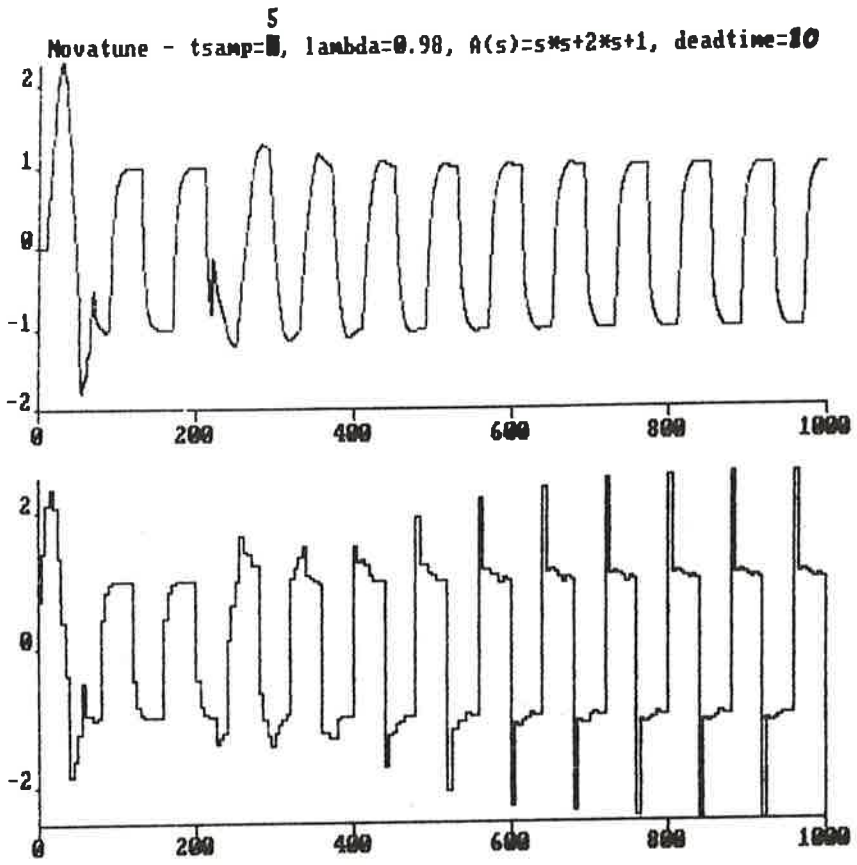
Figur B3 Stegsvar för PID, A(s)=s+1, T =0.5
modellerad dödtid=2, verklig dödtid=4
s



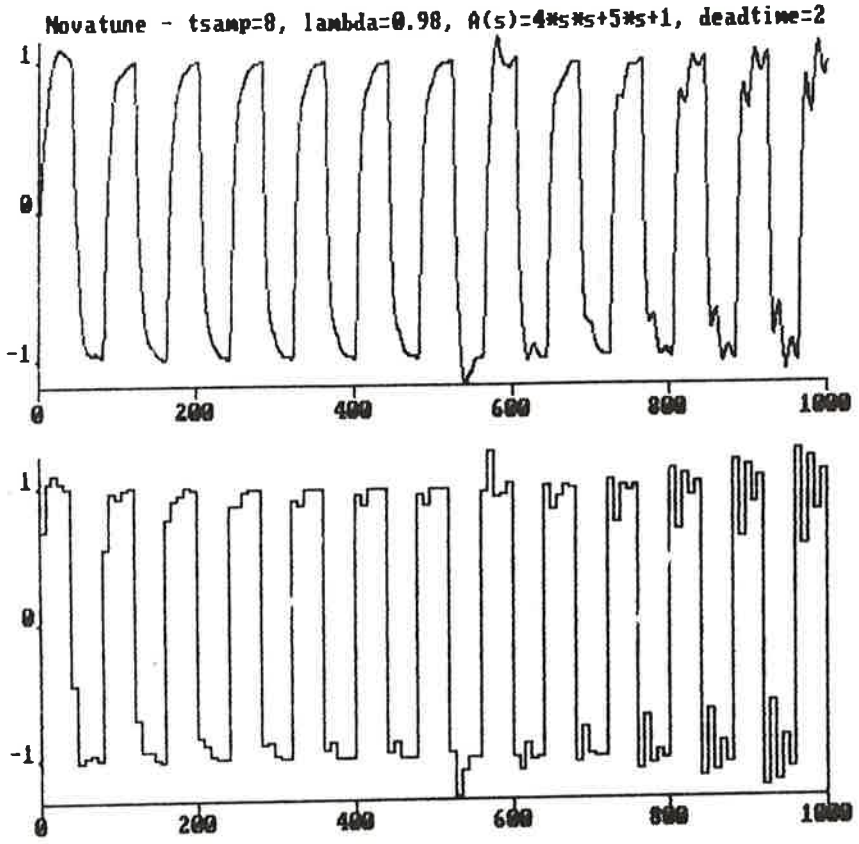
Figur B4 Stegsvar för Novatune plus extrasteg precis efter processen, A(s)=s*s+s, L=1, T_s=1



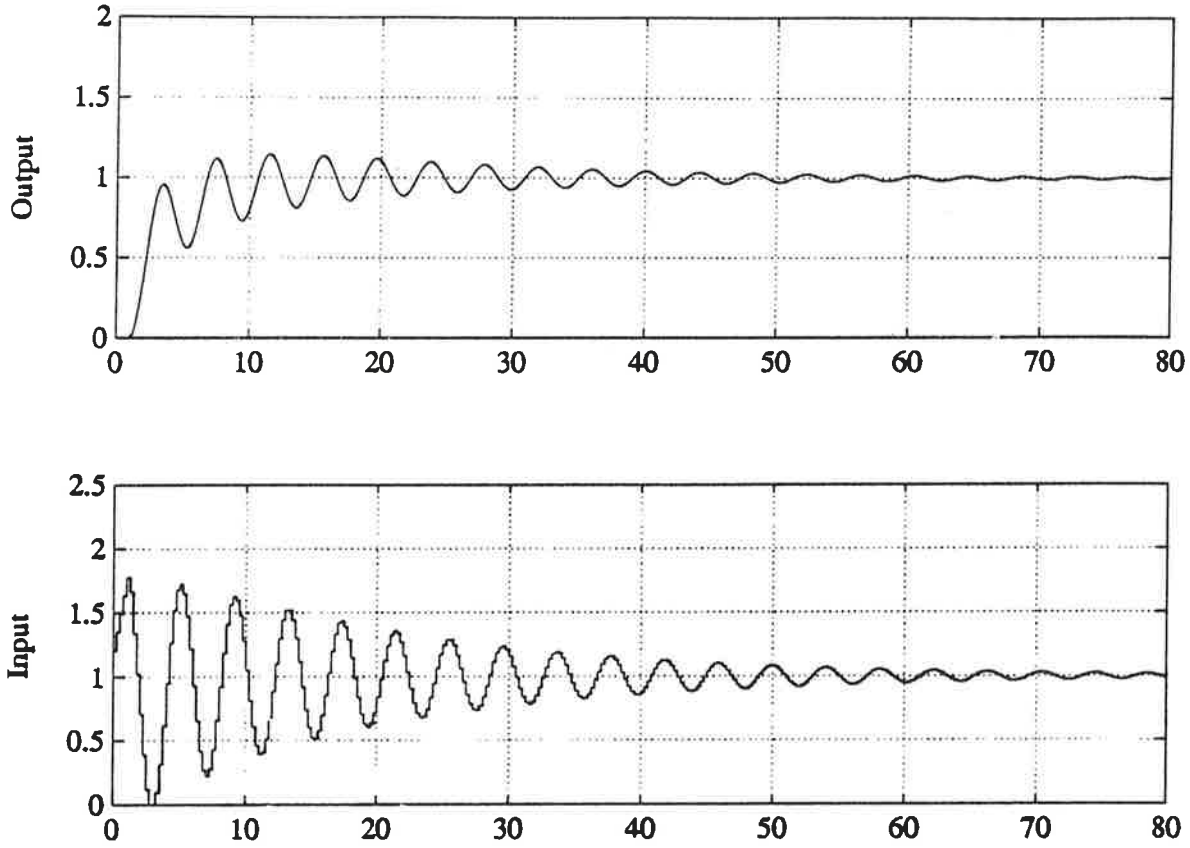
Figur B5 Stegsvar för PID, $A(s)=s^2+2s+1$, $T_s=0.5$, modellerad dödtid=2, verklig dödtid=6



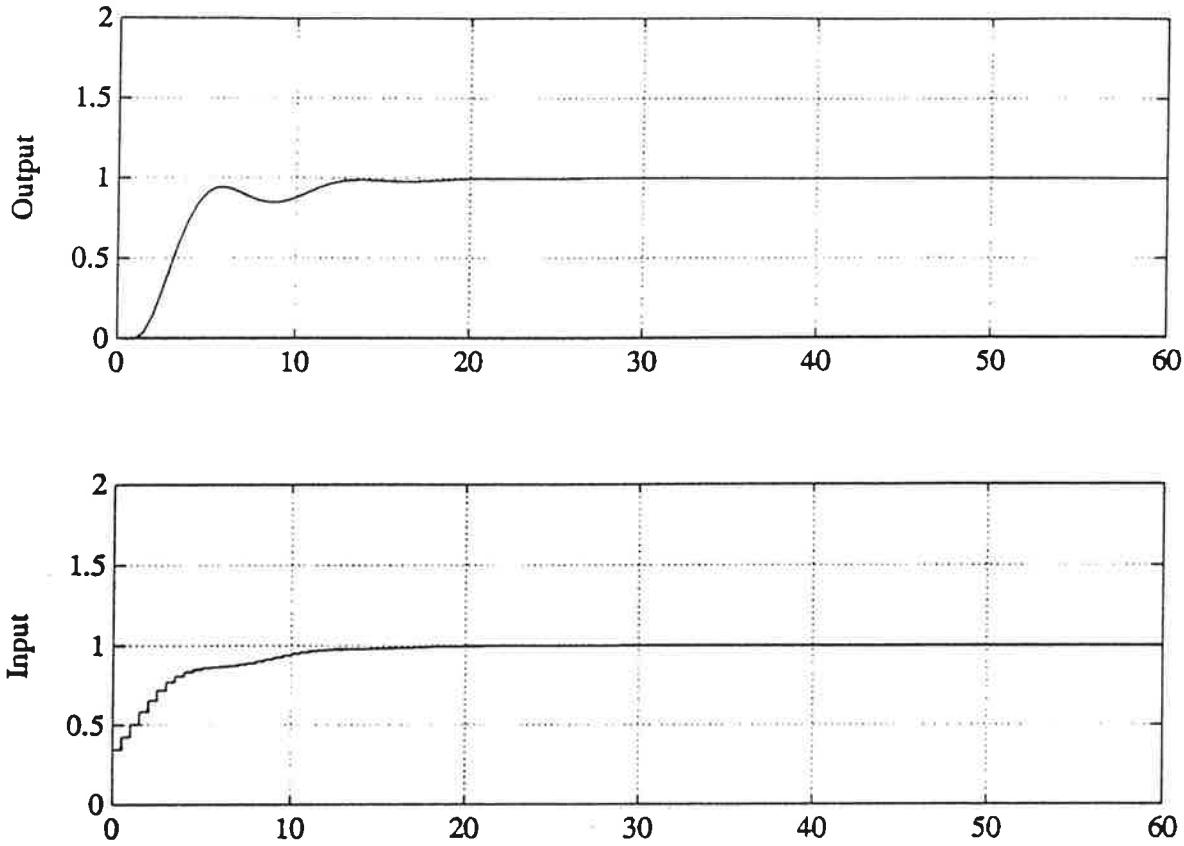
Figur B6 Övre diagrammet: utsignal,
nedre " : styrsignal,
modellerat $T_2=1$, verkligt $T_2=8$



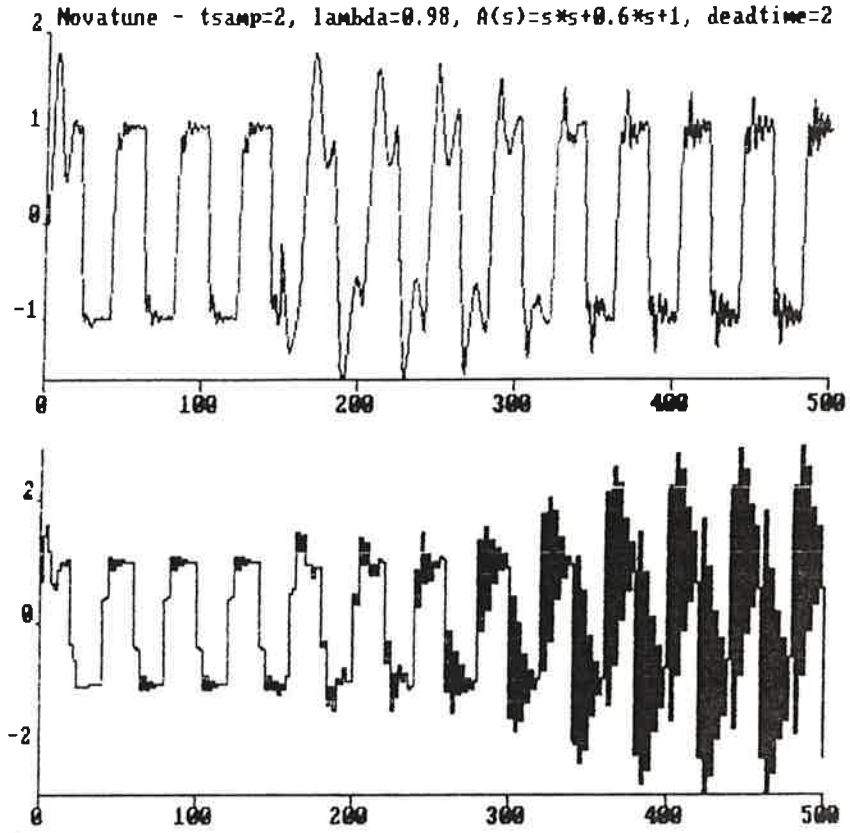
Figur B7 Övre diagrammet: utsignal,
nedre " : styrsignal,
verklig dödtid=4



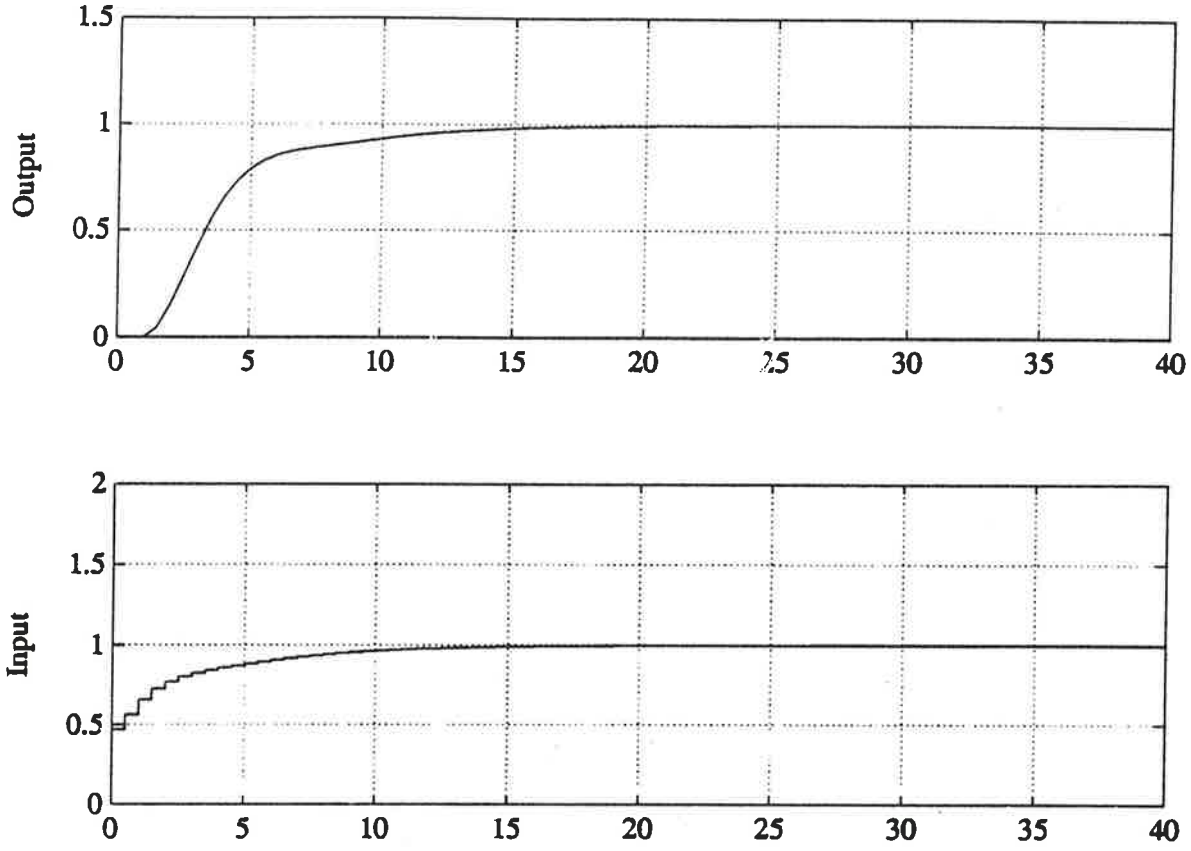
Figur B8 Stegsvar för PID, $A(s)=4s^2+5s+1$, L=1, $T_s=0.25$, modellerat $T_2=4$, verkligt $T_2=1$



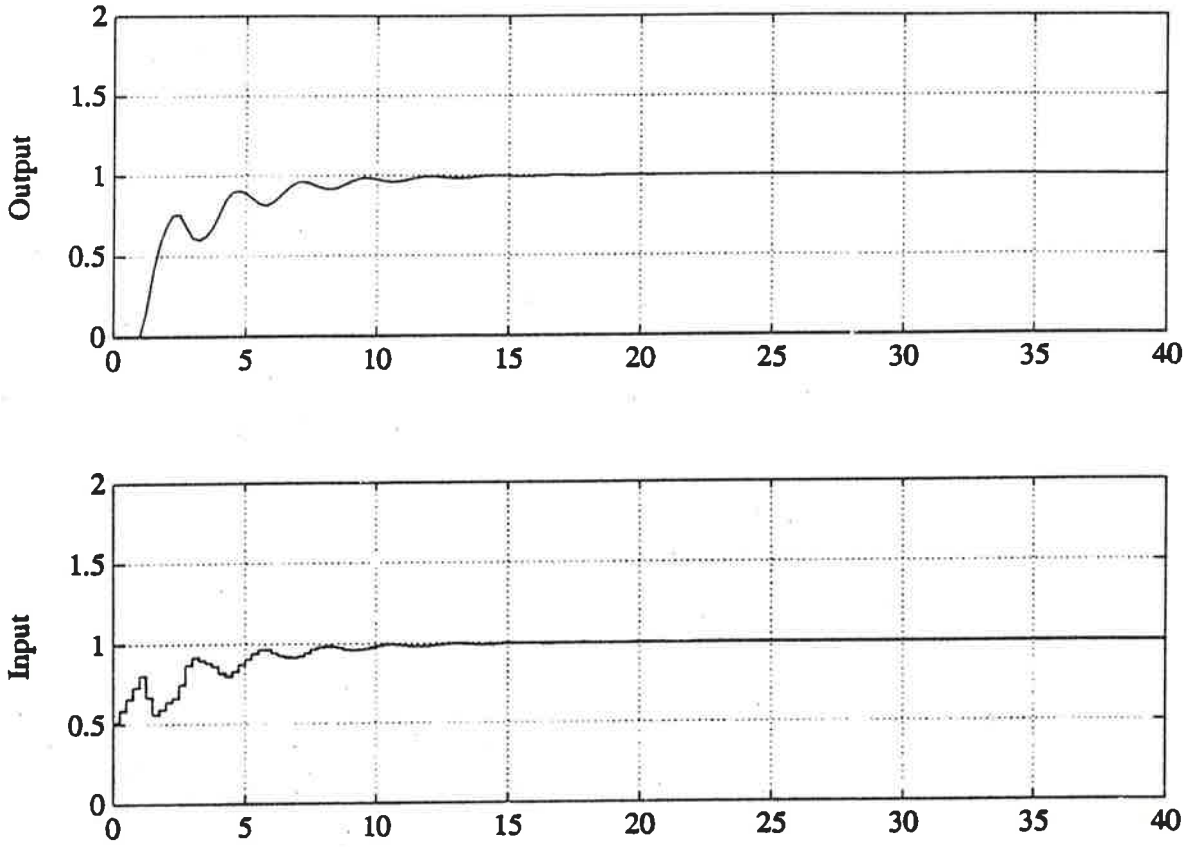
Figur B9 Stegsvar för PID, $A(s)=s^2+0.6s+1$, $T_s=0.5$, modellerad dödtid=2, verklig dödtid=1



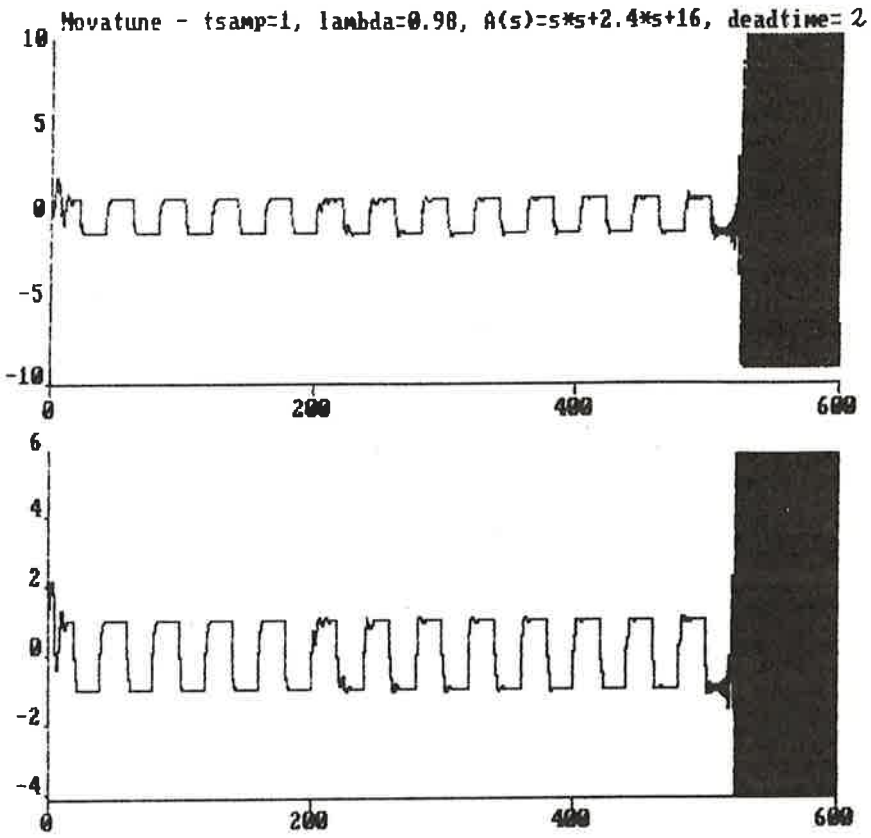
Figur B10 Övre diagrammet: utsignal,
nedre " : styrsignal,
modellerat ω=1, verkligt ω=0.5



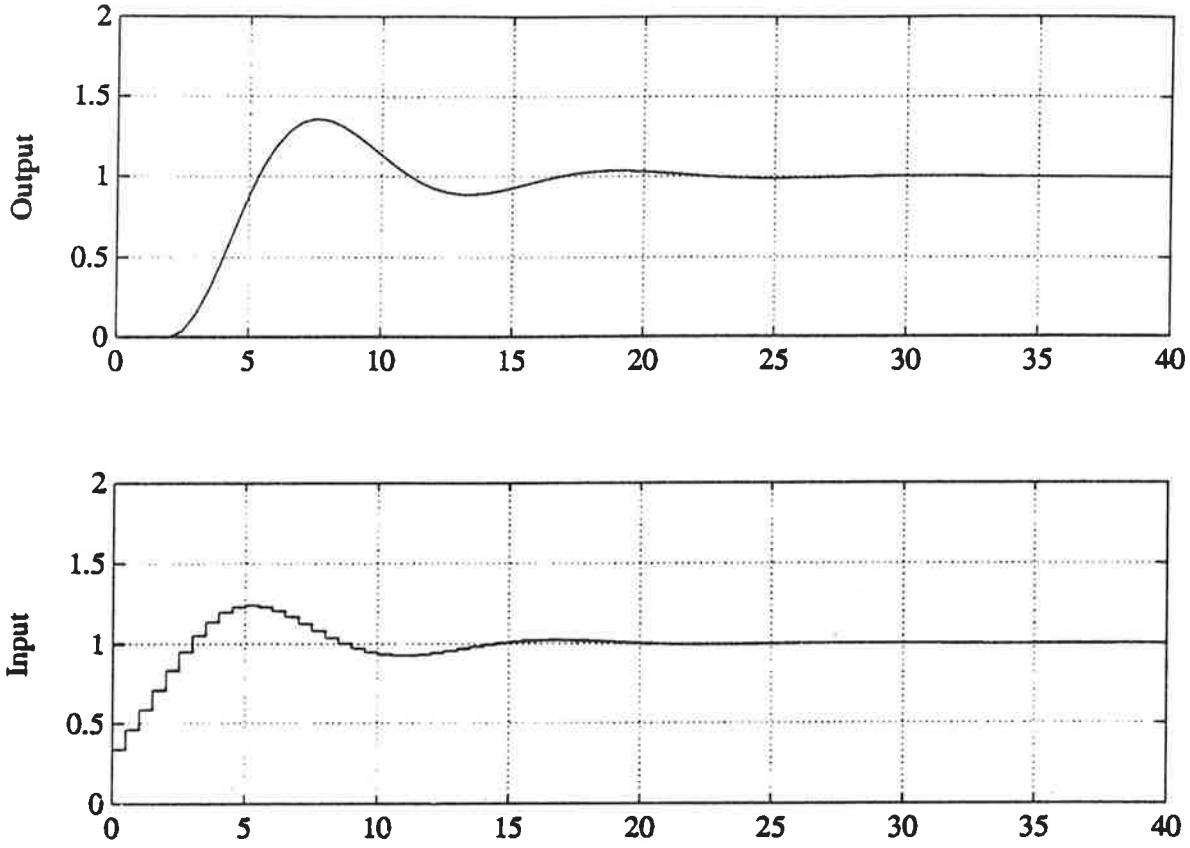
Figur B11 Stegsvar för PID, $A(s)=s^2+1.4s+1$, $T_s=0.5$, modellerad dödtid=2, verklig dödtid=1



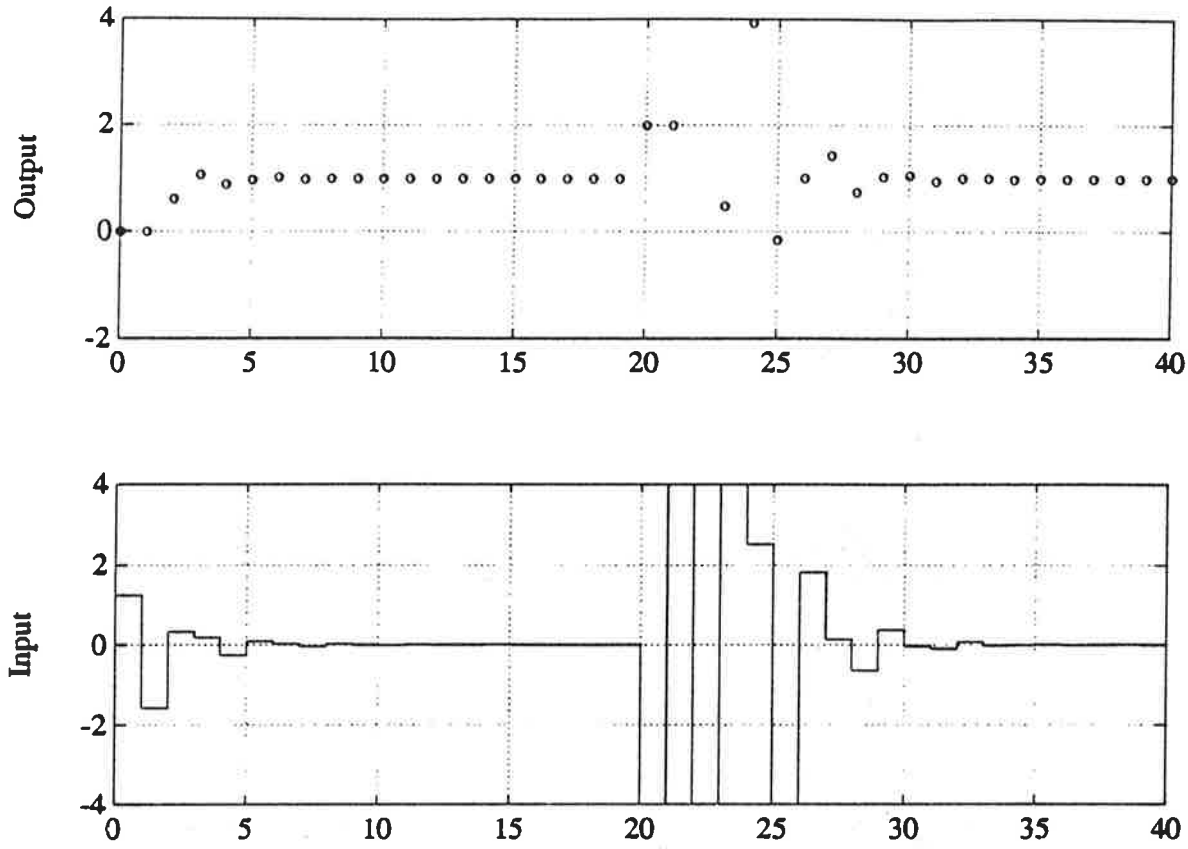
Figur B12 Stegsvar för PID, $A(s)=s^2+1.4s+1$, $L=1$, $T_s=0.25$, modellerat $\omega=1$, verkligt $\omega=4$



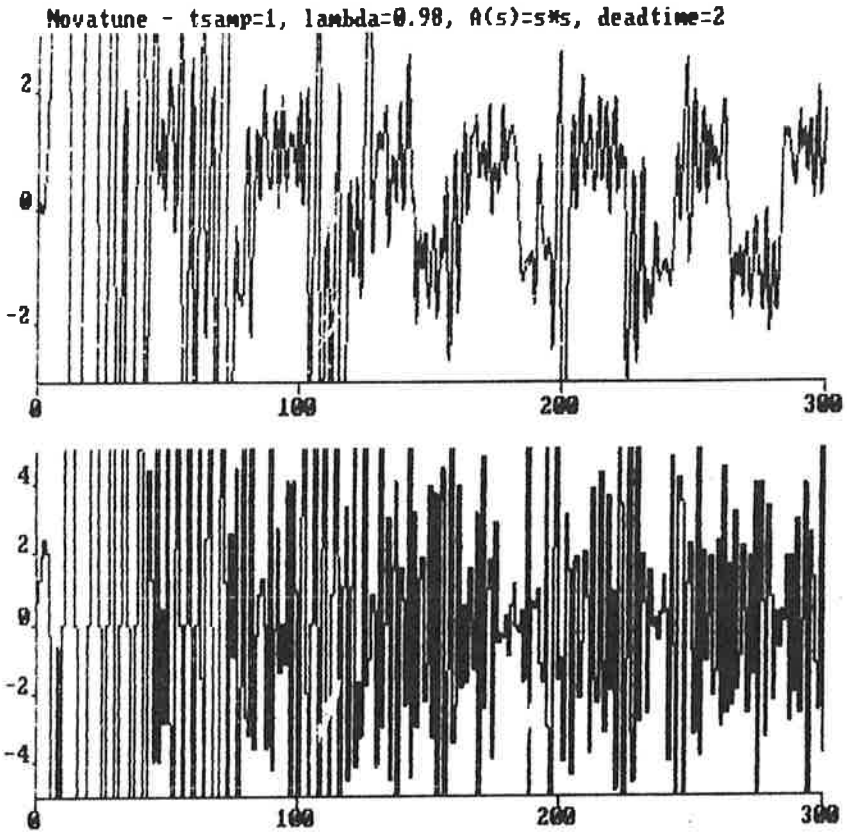
Figur B13 Övre diagrammet: utsignal,
nedre " : styrsignal,
verklig dödtid=1



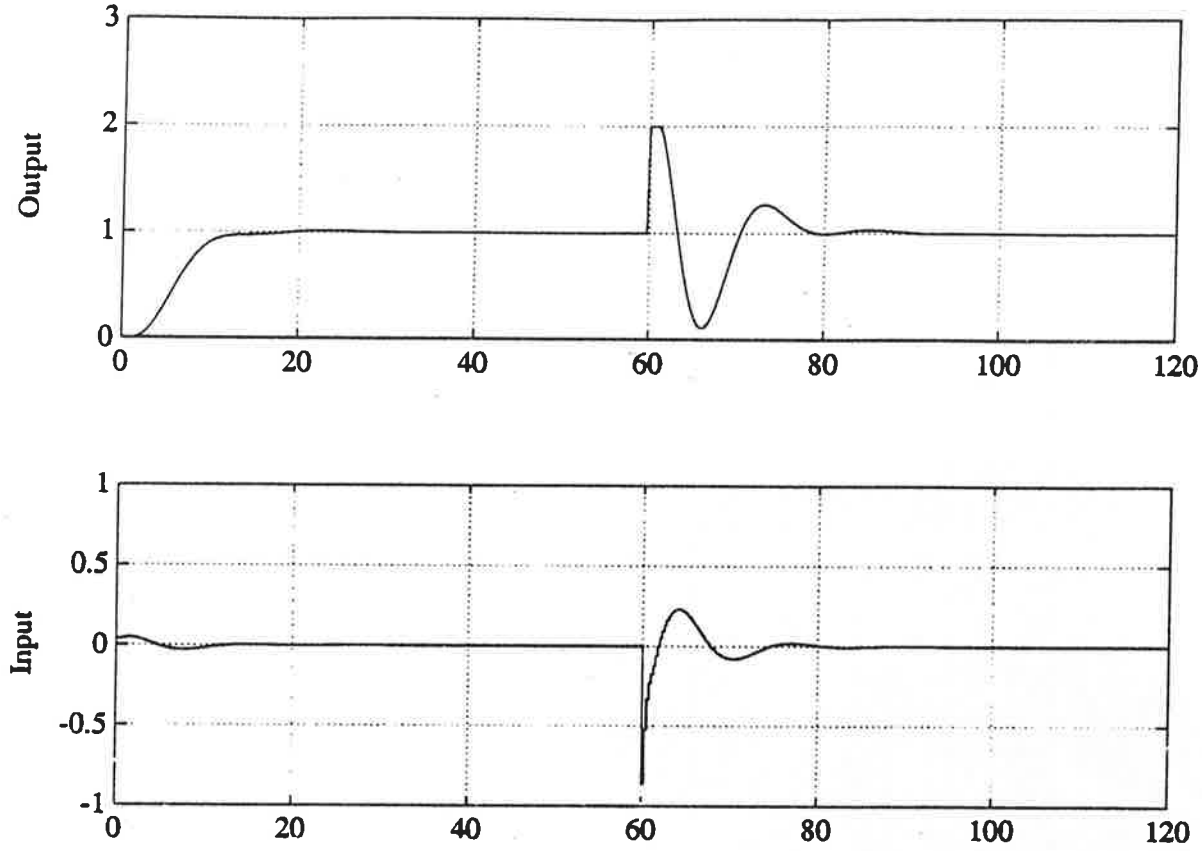
Figur B14 Stegsvar för PID, $A(s)=s^2+2.4s+16$, L=2, $T_s=0.5$, modellerat $\omega=4$, verkligt $\omega=1$



Figur B15 Stegsvar för Novatune plus extrasteg precis efter processen, $A(s)=s^2$, $L=1$, $T_s=1$



Figur B16 Övre diagrammet: utsignal,
nedre " : styrsignal,
mätbrus (vitt)



Figur B17 Stegsvar för PID plus extrasteg precis efter processen, $A(s)=s^2$, $L=1$, $T_s=0.25$