CODEN:LUTFD2/(TFRT-5288)/1-080/(1982)

METHODS FOR AUTOMATIC TUNING OF PID REGULATORS

KAI SIEW, WONG

DEPARTMENT OF AUTOMATIC CONTROL
LUNDS INSTITUTE OF TECHNOLOGY

DECEMBER 1982

LUND INSTITUTE OF TECHNOLOGY
DEPARTMENT OF AUTOMATIC CONTROL
Box 725
S 220 07 Lund 7 Sweden

Document name
REPORT
Date of issue
Dec 1982
Document number
CODEN:LUTFD2/(TFRT-5288)/1-080/(1982)

Author(s)
Kai Siew,Wong

Supervisor
Tore Hägglund

Sponsoring organization

Title and subtitle

Methods for automatic tuning of PID regulators

Abstract

A self-tuning PID-regulator has been investigated.The design method is based on the very well-known and simple Ziegler-Nichols self oscillation method.By introducing the describing function technique and by connecting a relay in the closed loop system,an oscillation is obtained automatically.It is noticed that this oscillation, whose amplitude is depending on the amplitude of the relay,has the same characteristics as the one,which is obtained manually by the Ziegler-Nichols self oscillation method,i.e.only proportional regulator is used and the oscillation is then obtained by increasing the gain gradually.Two methods,namely the peaks and zero-crossings method and the recursive least squares method,are proposed for estimating the amplitude and the frequancy of the oscillation,in order to determine the critical gain and the ultimate period.The settings of the PID-regulator are then determined according to the tuning rules.The investigations have carried out first without any disturbance with four processes,and then only one of the four processes with measurement noise and load disturbance.Different levels of the measurement noise are used.A relay with hysteresis is used in order to make those methods for estimating the parameters less sensitive to the measurement noise.The problems of high frequancy point of convergence and the level adjustment of the control signal have also been investigated.The results have shown that there is an advantage of using the relay with hysteresis and the self-tuner works quite well as a whole.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
Swedish

Number of pages
80

Recipient's notes

Security classification

FÖRORD

Föreliggande rapport har utförts i anslutning till mitt examensarbete under året 1982.

Adaptiv reglering är ett centralt område för forskning vid institutionen för Reglerteknik,Lunds Tekniska Högskola.En speciell klass av adaptiva regulatorer,som har utvecklats här vid institutionen är självinställande regulatorer.

Det här arbetet går ut på att undersöka en självinställande regulator baserad på Ziegler-Nichols självsvängningsmetod med hjälp av datorsimuleringar.

Arbetet har genomförts med hjälp av en VAX 11/780 4025. Programspråket är SIMNON.

För värdefull hjälp och vägledning vill jag rikta ett varmt tack till forskningsassistenten,civilingenjör Tore Hägglund från institutionen.

Lund i december 1982

Kai Siew , Wong

FÖRORD

INNEHÅLLSFÖRTECKNING

1. INLEDNING

Idag,är de flesta reglerproblemen inom industrin lösta med enkla regleringsalgoritmer av PID-typ.Dessa regulatorer är ofta analoga PID-regulatorer,och tillverkas i stora mängder som standardprodukter.Ingenjörer och processoperatörer är välbekanta med dem,och har ofta en bra känsla för effekterna av deras olika parametrar.

På grund av den snabba utvecklingen av digitala datorer, ersätts de analoga PID-regulatorerna mer och mer av små digitala datorer,och därför har det varit möjligt att införa mer avancerade regleringsalgoritmer.

När de opererande förhållandena i en process ändras,borde egentligen regulatorerna ställas om.Eftersom ett stort reglersystem kanske innehåller hundratals regulatorer,är detta ofta ett både svårt och tidsödande arbete.Det är därför nyttigt att ha anordningar,som automatiskt kan ställa in de enkla regulatorerna.Sådana anordningar och regulatorer kallas för självinställare.Det är också nyttigt om sådana självinställare är baserade på de välkända ideerna,eftersom det är så många personer som är vana vid standardregulatorerna och deras inställningsregler.

Vid institutionen har många självinställare baserade på olika metoder tagits fram.Detta arbete går ut på att undersöka en av dessa självinställare,som är baserad på Ziegler-Nichols självsvängningsmetod.

Rapporten är organiserad på följande sätt : I kapitel 2, görs en beskrivning av självinställaren.Här beskrivs dess princip,inställningsreglerna och regulatorparametrarna, tekniken hur man automatiskt kan framkalla självsvängningen,samt metoderna för att kunna skatta de önskade processparametrarna.En undersökning av självinställaren utförs i kapitlet 3.Här modifieras regulatorparametrarna och självinställaren testas med olika processer först utan störningar och därefter bara en av processerna med mätbrus och belastningsstörning. Mätbrusets standardavvikelse varieras.På samma sätt har en annan skattningsmetod,nämligen rekursiva minsta kvadratmetoden,testats på en och samma process.En undersökning av de olika skattningsmetodernas tillförlitlighet görs och sist i detta kapitel,undersöks två praktiska problem,nämligen den högfrekventa konvergenspunkten och justeringen av styrsignalens referensnivå.I kapitel 4 noteras några slutsatser.

2. BESKRIVNING AV SJÄLVINSTÄLLAREN

2.1 Princip

Självinställaren kan tänkas bestå av tre olika block.Se Figur 1.Block ett består av en vanlig PID-regulator,i block två skattas vissa processparametrar medan block tre beskriver vilken relation som ska gälla mellan regulatorparametrarna och de skattade processparametrarna.

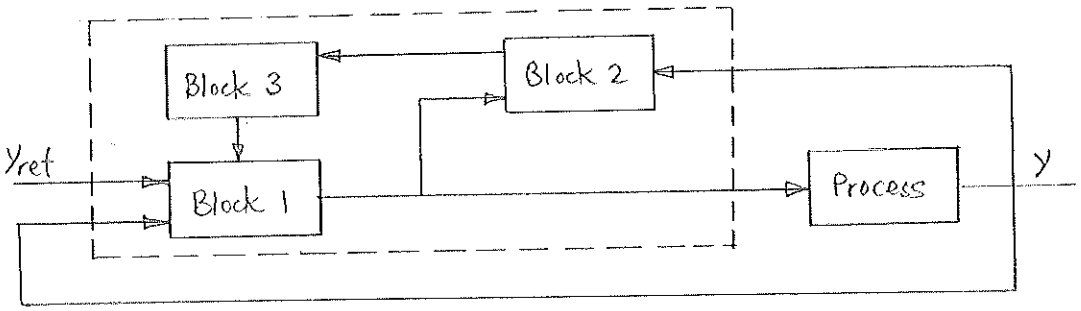


Fig.1 Blockschema för en självinställare.

Självinställaren,som kommer att behandlas i fortsättningen, är baserad på Ziegler-Nichols självsvängningsmetod.Detta innebär att block tre ska se till att det slutna systemet oscillerar,så att relationen mellan regulatorparametrarna och processparametrarna enligt inställningsreglerna, uppfylls.I block två skattas alltså de två processparametrarna och i block ett byts PID-regulatorn ut mot en olinjär länk,så att villkoret på självsvängningen uppfylls.När skattningarna är klara,byts PID-regulatorn omedelbart tillbaka och samtidigt sker inställningen.Block ett fungerar då som en ordinarie regulator med konstanta parametrar.

2.2 Regulatorn

Regulatorn som ingår i block ett ,är av PID-typ och dess inställning är baserad på Ziegler-Nichols metod.Figur 2 visar ett slutet system som består av en process och en enkel PID-regulator.Processen antas kunna beskrivas som ett linjärt tidsinvariant system med överföringsfunktionen,G.

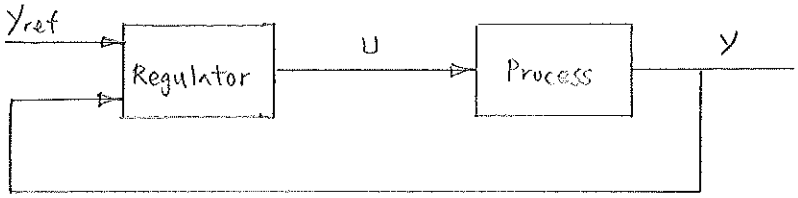


Fig.2 Blockschema för ett slutet system.

Metoden som heter självsvängningsmetod,är baserad på en mycket enkel karakterisering av processens dynamik och inställningen är baserad på en mätning av processens dynamiska egenskaper.Regulatorn kopplas först som en ren proportionell regulator,dvs $T_I = \infty$ och $T_D = 0$.Sedan ökas regulatorns förstärkning tills det slutna systemet uppnår gränsen för instabilitet.Det kritiska värdet på förstärkningen,K_c och svängningens period,T_c registreras.

Detta utförande baseras alltså på de egenskaper av endast en punkt på Nyquistdiagrammet av processens överföringsfunktion ,G,nämligen den första punkten där Nyquistdiagrammet skär den negativa reella axeln,dvs den punkten som ger 180 graders fasförskjutning.Se Figur 3.

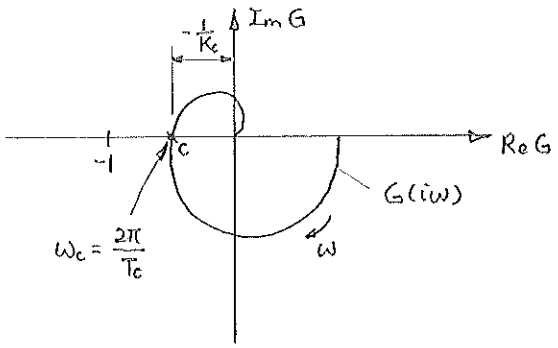


Fig.3 Nyquistdiagram för processens överföringsfunktion.

Lämpliga värden på regulatorparametrarna i olika fall ges i tabell 2.1.Dessa inställningsregler enligt Ziegler-Nichols metod är ursprungligen baserade på resultat av olika simuleringar av många enkla system,och de brukar ge de slutna systemen dålig dämpning.Men detta kan förbättras genom att göra modifikationer av koefficienterna i tabellen.

Regulator	K	T_I	T_D
P	0.5 K_c		
PI	0.45 K_c	0.83 T_c	
PID	0.6 K_c	0.5 T_c	0.12 T_c

Tabell 2.1 Rekommenderade regulatorinställningar baserade på Ziegler-Nichols metod baserad på självsvängning.

2.3 Skattning av parametrarna,K_c och T_c

Förutsättning

Processen har antagits vara av lågpasstyp och kunna beskrivas som ett linjärt tidsinvariant system med överföringsfunktionen,G.För att kunna uppskatta de två parametrarna,som nämndes i avsnitt 2.2,dvs K_c och T_c,krävs det att processen måste oscillera.Detta innebär att Nyquistdiagrammet av processens överföringsfunktion,G ska skära den negativa reella axeln i G-plan et minst en gång. Figur 4 visar några typiska Nyquistdiagram för olika processer.

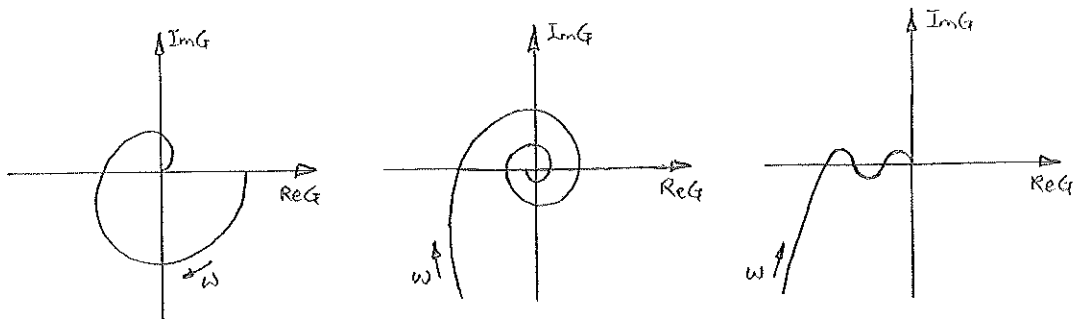


Fig.4 Tre typiska Nyquistdiagram för olika processer.

Princip

Proceduren,som beskrevs i avsnitt 2.2 för att få det slutna systemet att oscillera,är inte lätt att automatisera.En

annan metod är därför föreslagen för att automatiskt kunna bestämma den specifika punkten på Nyquistdiagrammet.Metoden med beskrivande funktion används för detta ändamål,så att villkoren för svängningen kan undersökas.Se t.ex ref.(4).

Beskrivande funktion

Betrakta det reglersystem,vars blockschema visas i Figur 5. Blocket N är ett olinjärt tidsinvariant system och blocket L är ett linjärt tidsinvariant system av lågpasstyp och har överföringsfunktion,G.Det förutsätts vidare att länken N inte likriktar insignalen eller ger upphov till subharmoniska svängningar.

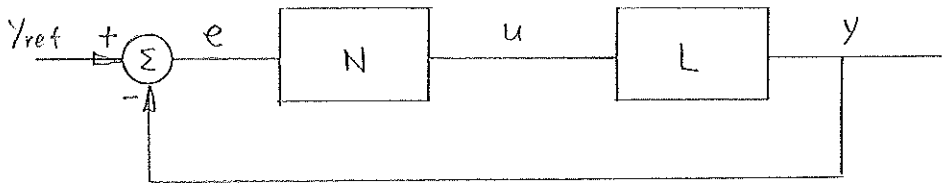


Fig.5 Blockschema för ett reglersystem.

För att kunna tillämpa linjär teori,är det önskvärt att den olinjära länken N approximeras med ett linjärt uttryck för en lämplig klass av insignaler till N.Eftersom villkoren för självsvängningen ska undersökas,väljs en insignal av formen e(t)=A*sin(ωt),se Figur 6.

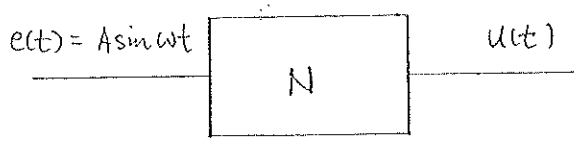


Fig.6 Det olinjära systemet,N,som ska approximeras.

Insignal-utsignal sambandet förutsätts känt.Med beskrivande funktion,N(A) för det olinjära systemet i Figur 6 menas det komplexa talet $(b_1+ia_1)/A$.

$$\text{dvs } N(A) = \frac{b_1 + ia_1}{A} = \frac{c_1 * e^{i\varphi}}{A}$$

där a_1 och b_1 är koefficienterna i fourierserieutvecklingen av u(t),då insignalen är A*sin(ωt).N(A) kan fysikaliskt tolkas som en amplitudberoende överföringsfunktion från sinusformad insignal till grundtonen av den periodiska utsignalen.Genom ett lämpligt val av olinjäriteten,kan man t.ex med hjälp av uppritningen av -1/N(A) i G-planet,bestämma en viss punkt i Nyquistdiagrammet av det linjära systemets överföringsfunktion.

Beskrivande funktionen för ett relä

Figur 7 visar att den olinjära länken,N i Figur 5 består av ett relä.

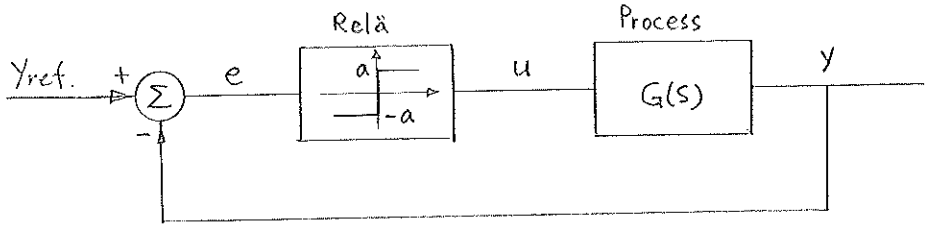


Fig.7 Ett slutet system med ett relä.

För att metoden för beskrivande funktionen ska fungera,krävs det att insignalen till reläet är sinusformad.Om så är fallet, blir insignalen till processen då en fyrkantvåg,och eftersom processen är av lågpasstyp,så är svängningsamplituder av de höga frekvenserna i y mindre än grundtonens amplitud,dvs y blir då nära sinusformad.Detta stämmer bra för metoden. Anta nu att insignalen till reläet är en sinusvåg med amplitud, A och frekvens,ω,dvs e(t)=A*sin(ωt).Då blir den beskrivande funktionen för reläet:

$$N(A) = \frac{4a}{\pi A}$$

Den negativa inversen av N(A) uppritas vanligen tillsammans med Nyquistdiagrammet av G i G-planet,se Figur 8.

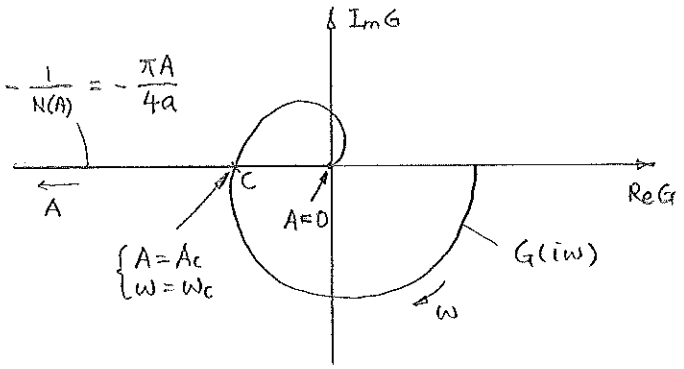


Fig.8 Den negativa inversen av N(A) för relä och Nyquistdiagrammet av G.

Det slutna systemet kommer att oscillera då båda kurvorna skär varandra,och skärningspunkten bestämmer amplituden och frekvensen av just denna oscillation.Jämförs Figur 3 och Figur 8,märker man att det ju är samma punkt som ska bestämmas enligt självsvängningsmetoden,därför kan de två parametrarna,K_c och T_c då bestämmas enligt nedan:

$$K_c = \frac{4a}{\pi A_c}$$

$$T_c = \frac{2\pi}{\omega_c}$$

Beskrivande funktion för ett relä med hysteres

Figur 9 visar karakteristiken för ett relä med hysteres. Anledningen till att använda ett relä med hysteres,är att göra metoden mindre känslig för mätbrus.

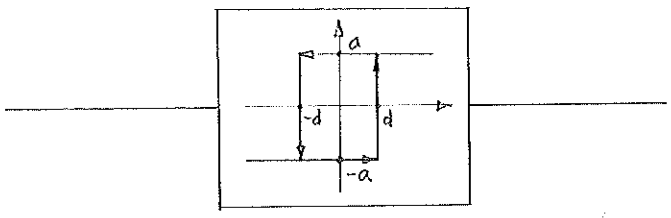


Fig.9 Karakteristik för relä med hysteres.

Den beskrivande funktionen för ett relä med hysteres blir:

$$N(A) = \frac{4a}{\pi A} e^{-i\varphi} \quad ; \text{ där } \varphi = \arcsin\left(\frac{d}{a}\right) \; ; \; a \geq d$$

Den negativa inversen av N(A) blir då:

$$-\frac{1}{N(A)} = -\frac{\pi A}{4a} e^{i\varphi} = -\frac{\pi A}{4a}(\cos\varphi + i\sin\varphi) =$$

$$= -\frac{\pi}{4a}\sqrt{A^2 - d^2} - i\frac{\pi d}{4a} \; ; \; a \geq d.$$

Tydligen,är den imaginära delen av -1/N(A) en konstant,och detta innebär att kurvan -1/N(A) då är en rät linje parallell med den reella axeln.Se Figur 10.

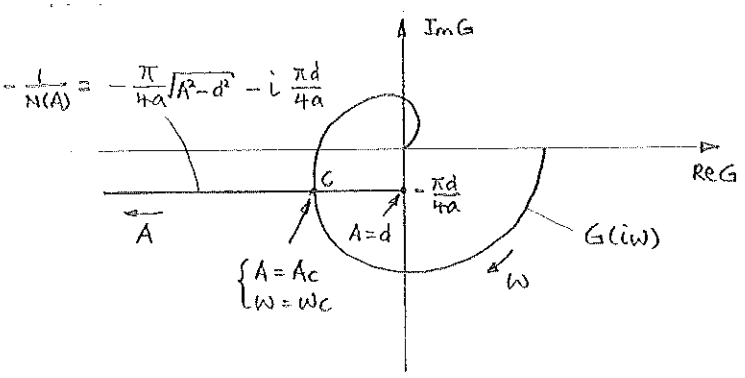


Fig.10 Den negativa inversen av N(A) för relä med hysteres och Nyquistdiagrammet av G.

Jämförs Figur 8 och Figur 10,ser man alltså att man får en annan stabil svängning.Men man kan få en bra uppskattning på skärningspunkten med reella axeln genom att välja d-värdet tillräckligt litet i förhållande till a.De två parametrarna ,K_c och T_c ,kan då approximeras enligt nedan:

$$K_c = \frac{4a}{\pi\sqrt{A_c^2 - d^2}}$$

$$T_c = \frac{2\pi}{\omega_c}$$

Man kan konstatera att ett vanligt relä är ett specialfall till ett relä med hysteres då d=0,därför kommer i fortsättningen relä med hysteres att behandlas.

Bestämning av K_c och T_c

För att kunna bestämma de två parametrarna,K_c och T_c, enligt den beskrivna metoden,måste svängningsamplituden och svängningsfrekvensen(eller svängningsperioden) bestämmas. Detta kan utföras på många olika sätt.Två metoder beskrivs nedan:

Toppar och nollgenomgångar

Denna metod är enkel för uppskattningarna är bara baserade på räkning och jämförelser.Svängningsperioden kan lätt bestämmas genom att notera tiderna mellan nollgenomgångar och svängningsamplituden kan bestämmas genom att notera topparnas värden (både maximum och minimum) hos utsignalen, y(t).

Rekursiva minsta kvadratmetoden

Uppskattning av perioden är baserad på observationen att en sinusformad funktion med period T uppfyller den linjära differentialekvationen.

$$y(t) - \theta y(t-h) + y(t-2h) = 0$$

där h är samplingsperioden och

$$\theta = 2\cos\left(\frac{2\pi h}{T}\right)$$

Genom att minimera nedanstående funktion med avseende på parametern,$\hat{\theta}$,

$$\sum \left(y(t) - \hat{\theta} y(t-h) + y(t-2h) \right)^2$$

kan perioden uppskattas från:

$$\hat{T} = \frac{2\pi h}{\arccos(\frac{\hat{\theta}}{2})}$$

När perioden har framtagits,kan man utgå från det allmänna uttrycket för sinusfunktionen enligt nedan:

$$y(t) = A\sin(\omega t + \varphi) \quad \text{eller}$$

$$y(t) - \theta_1 \sin\omega t - \theta_2 \cos\omega t = 0$$

$$\text{där } \omega = \frac{2\pi}{\hat{T}}$$

Amplituden bestämmes genom att minimera nedanstående funktion med avseende på parametrarna,$\hat{\theta}_1$ och $\hat{\theta}_2$.

$$\sum \left(y(t) - \hat{\theta}_1 \sin\omega t - \hat{\theta}_2 \cos\omega t \right)^2 \; ; \text{ där } \omega = \frac{2\pi}{\hat{T}}$$

Amplituden kan då uppskattas från:

$$A = \sqrt{\hat{\theta}_1^{\,2} + \hat{\theta}_2^{\,2}}$$

Justering av svängningsamplituden

Självsvängningsmetoden åstadkommer en naturlig självsvängning hos det slutna systemet,men däremot är självsvängningen amplitudberoende då metoden med beskrivande funktion används.Genom att justera reläets amplitud,kan man alltså få en önskad svängningsamplitud.Detta kan lätt göras automatiskt,men för enkelhets skull,används i stället en konstant amplitud på reläet i fortsättningen.

3. UNDERSÖKNING AV SJÄLVINSTÄLLAREN

3.1 Inledning

Ett antal simuleringar av olika processer genomfördes för att undersöka en självinställare baserad på de ideer som beskrevs i föregående kapitel.Figur 11 visar blockschemat för självinställaren.

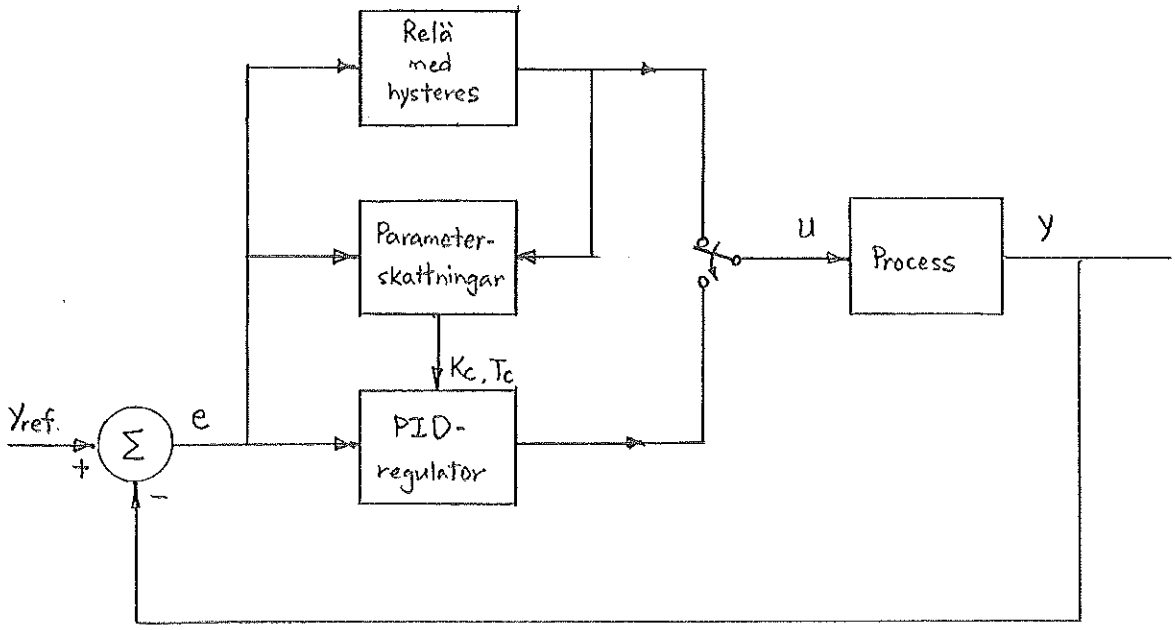


Fig.11 Blockschema för självinställaren.

Undersökningarna är organiserade på följande sätt i de olika avsnitten : Fyra olika processer simuleras utan störningar i avsnitt 3.2.Här används ett relä samt metoden med toppar och nollgenomgångar för parameterskattningar.Eftersom inställningsreglerna brukar ge de slutna systemen en dålig dämpning,görs också modifikationer av koefficienterna i tabellen för olika processer i detta avsnitt.Simuleringar med mätbrus och belastningsstörning genomförs i avsnitt 3.3. Här används ett relä med hysteres i stället och metoden för parameterskattningar är densamma som i avsnitt 3.2.I avsnitt 3.4 används en annan metod,nämligen den rekursiva minsta kvadratmetoden i stället och i avsnitt 3.5 görs en undersökning av metodernas tillförlitlighet genom att skatta parametrarna under många perioder.Några problem och/eller svårigheter berörs i avsnitt 3.6.

3.2 Simuleringar utan störningar

Följande fyra processer studeras först utan störningar:

Process 1 :

$$G(s) = \frac{1}{(1 + s)^4}$$

Process 2 :

$$G(s) = \frac{e^{-5s}}{(1 + s)}$$

Process 3 :

$$G(s) = \frac{1 - s}{(1 + 2s)^3}$$

Process 4 :

$$G(s) = \frac{10(s + 5)^2}{s(s + 0.5)^2(s + 50)^2}$$

Här används specialfallet av ett relä med hysteres,dvs ett relä,se bilaga prog.1,för att sätta det slutna systemet i oscillation och metoden med toppar och nollgenomgångar för att utföra parameterskattningar under oscillationen.

Skattningsalgoritmen går ut på att räkna antalet nollgenomgångar och notera tiden mellan varannan nollgenomgång,dvs perioden.Samtidigt noteras också både det maximala och det minimala värdet under varje period och därpå erhålls amplituden genom att halvera skillnaden mellan de två värdena.Se bilaga prog.3 och prog.4.Skattningarna börjar inte förrän några perioder har gått,detta för att låta det slutna systemet uppnå en stabil svängning,innan skattningarna sker.Hur många perioder som behöver filtreras bort är beroende på processens trögheter och här sätts initialvärdet,JO t.ex till -2 om två perioder ska filtreras bort.Efter filtreringen sker skattningarna under två perioder och här sätts parametern,IP till 2.Reläets amplitud sätts till ett.

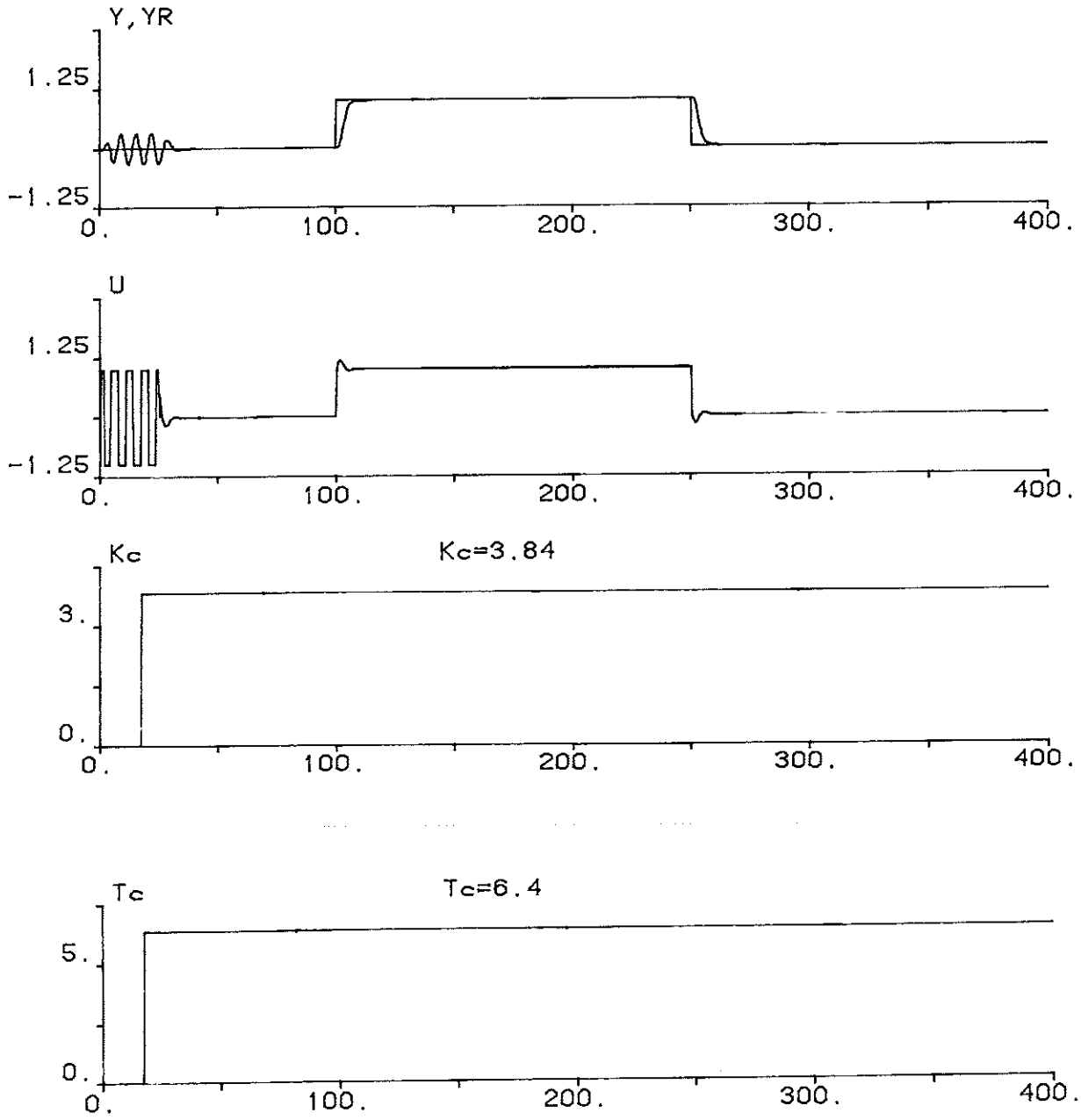
Det är en allmän erfarenhet att inställningsreglerna ger de slutna systemen en dålig dämpning.Det är därför klokt att först göra modifikationer av koefficienterna i tabell 2.1, så att det kan underlätta undersökningarna i fortsättningen. Med hjälp av Bode-diagrammet,kan de två kritiska värdena, dvs K_c och T_c,lätt erhållas.PID-regulatorn ställs in enligt tabell 2.1 och modifikationer av koefficienterna görs därefter manuellt.Tabell 3.1 visar de modifierade värdena på regulatorparametrarna för de olika processerna.Dessa värden används i fortsättningen.

Process	K	T_I	T_D
1	0.25 K_c	0.53 T_c	0.13 T_c
2	0.18 K_c	0.20 T_c	0.05 T_c
3	0.28 K_c	0.52 T_c	0.12 T_c
4	0.25 K_c	1.00 T_c	0.25 T_c

Tabell 3.1 Modifierade värdena på regulatorparametrarna.

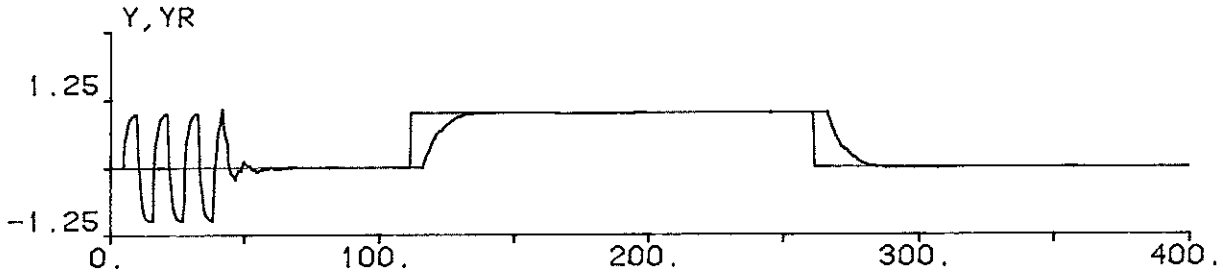
Simulering 3.2.1

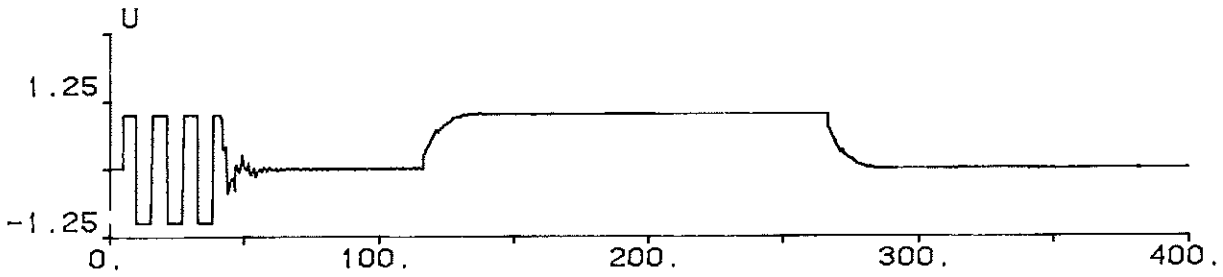
JO : -2 , IP : 2

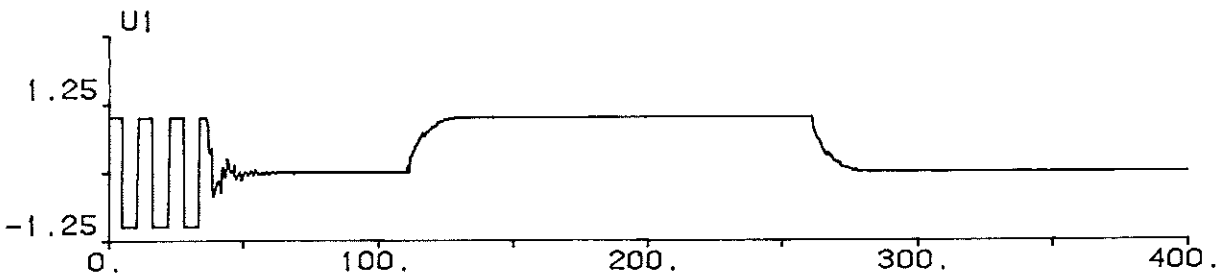


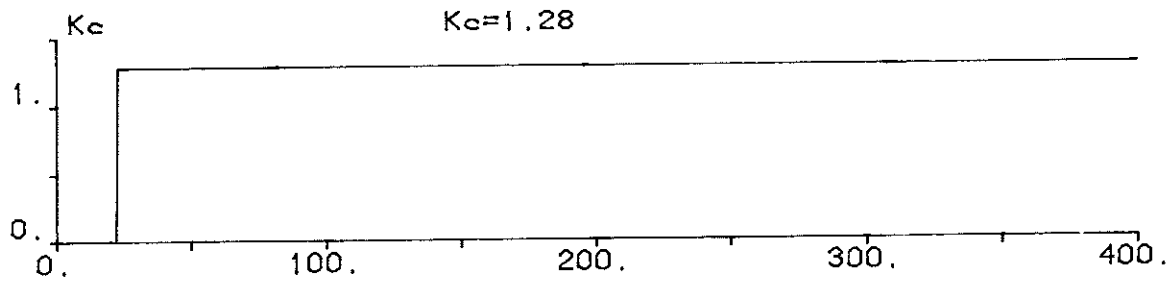
Simulering 3.2.2

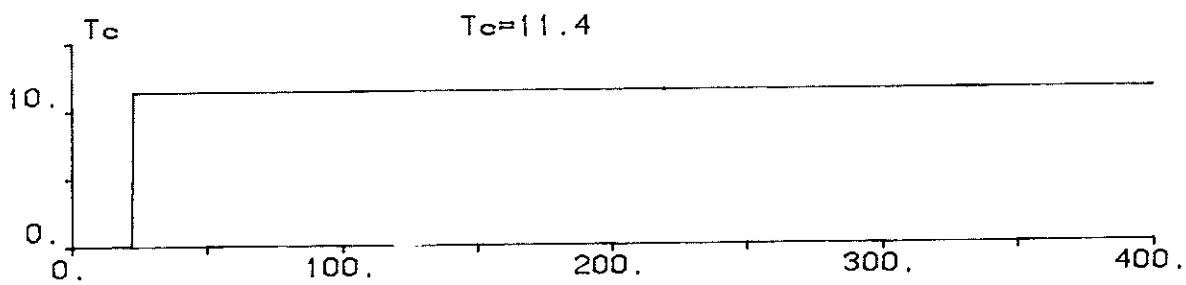
JO : 0 , IP : 2





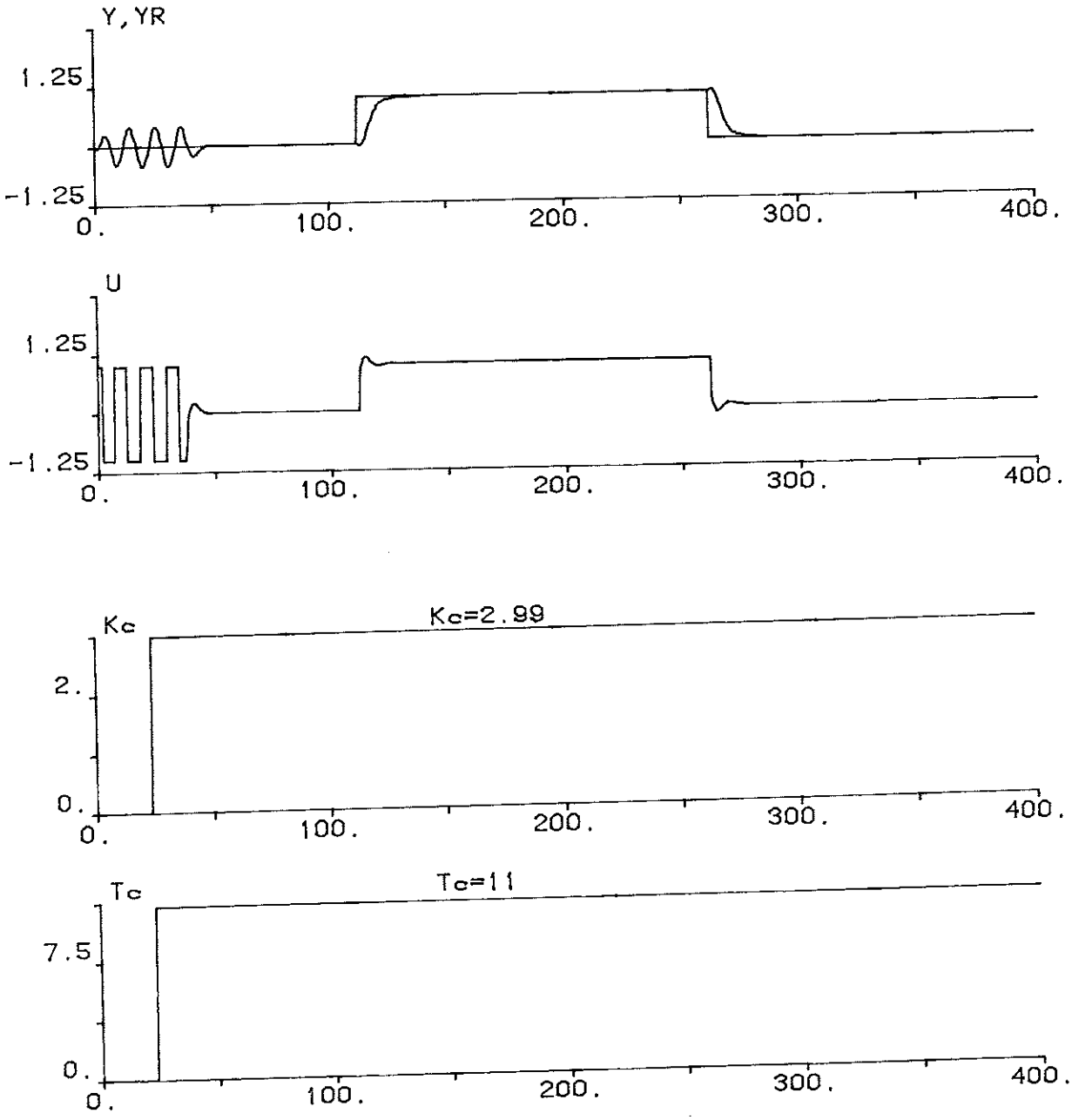






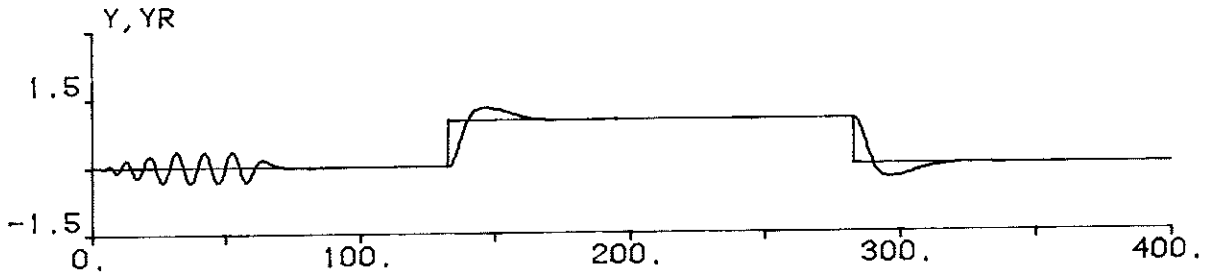
Simulering 3.2.3

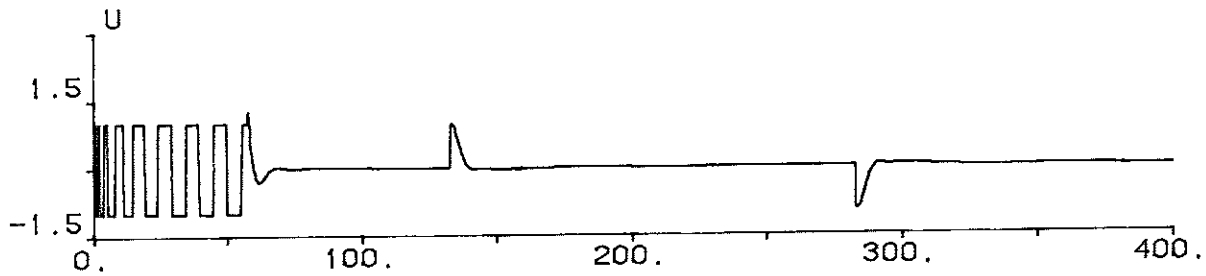
JO : -1 , IP : 2

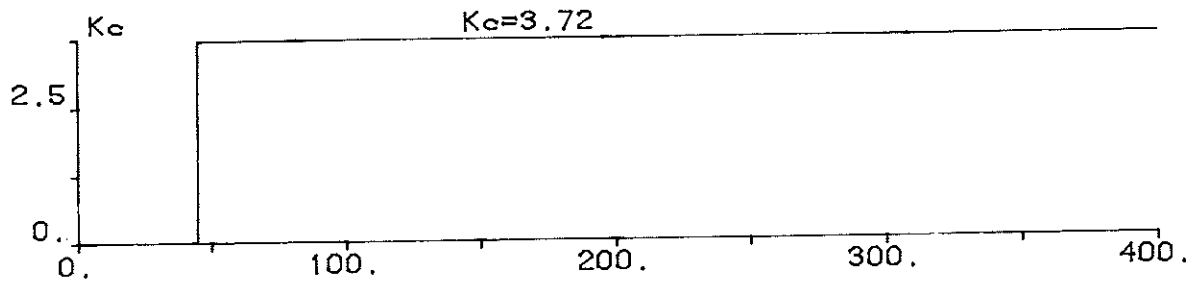


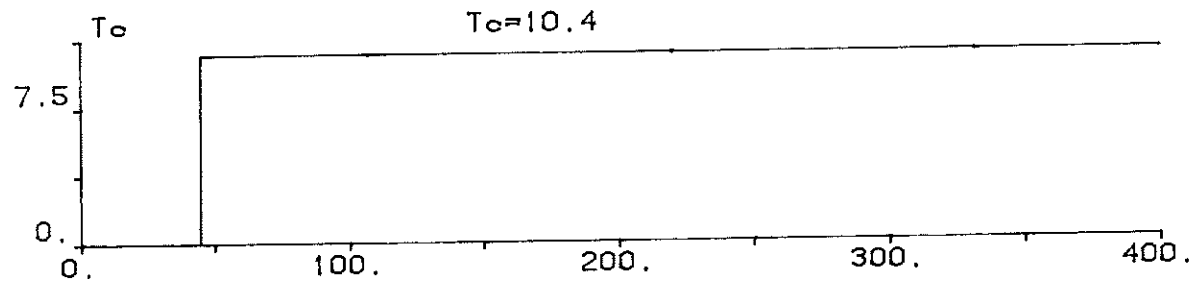
Simulering 3.2.4

JO : -6 , IP : 2









Kommentar

I de diskreta systemen har samplingsperioden valts till 0.1. Ju mindre samplingsperiod man väljer,desto bättre skattningar erhåller man,men i gengäld kräver det en betydligt längre simuleringstid.

D-delen i PID-regulatorn predikterar reglerfelet,som är differensen mellan referensvärdet,Yr och utsignalen,Y,dvs e=Yr-Y.Men ofta kan referensvärdet,Yr ej predikteras, särskilt då referensvärdesändringarna är stegformade och som resultat får man pulserande styrsignaler vid dessa ändringar och därför ersätts derivatatermen,de/dt ofta med -dy/dt.

Filtreringskonstanten,N i D-delen ligger vanligen mellan 3 och 30,och här har N valts till 3 för att få en hög filtreringsverkan.Detta gäller särskilt process 2 med tidsfördröjningen då man får en smidigare styrsignal.Dessa störningar uppstår troligen på grund av det diskreta systemet,DELAY.

Man märker att både perioden och amplituden växer i olika fart beroende på processernas trögheter,tills en stabil svängning har uppnåtts.

Processer med stora trögheter tar lång tid att uppnå en stabil svängning.Detta innebär att flera perioder i början bör filtreras bort innan skattningarna sker.För process 1 har 2 perioder filtrerats bort,dvs JO=-2,och för process 2 JO=0,för process 3 JO=-1,och för process 4 JO=-6. Observera att JO maximalt får vara lika med noll,dvs ingen filtrering behövs.

Tabell 3.2 visar en jämförelse mellan de "riktiga" och skattade processparametrarna och regulatorparametrarna för alla fyra processerna.

(a).

Process	K_c		T_c	
	Riktig	Skattad	Riktig	Skattad
1	4.0	3.84	6.3	6.4
2	1.14	1.28	11.6	11.4
3	3.2	2.99	10.6	11.0
4	4.0	3.72	10.0	10.4

(b).

Process	Regulator					
	Riktiga			Skattade		
	K	T_I	T_D	K	T_I	T_D
1	1.0	3.34	0.82	0.96	3.39	0.83
2	0.21	2.32	0.58	0.23	2.28	0.57
3	0.9	5.51	1.27	0.84	5.72	1.32
4	1.0	10.0	2.5	0.93	10.4	2.6

Tabell 3.2 "Riktiga" och skattade värdena på processparametrarna och regulatorparametrarna.

Det måste nämnas att Nyquistdiagrammet av process 4:s överföringsfunktion skär den negativa reella axeln i G-planet fler än en gång,dvs risken finns att den stabila svängningen hamnar på en annan konvergenspunkt vid högre frekvens,vilket inte är önskvärt.Detta problem kommer att behandlas i avsnitt 3.6.

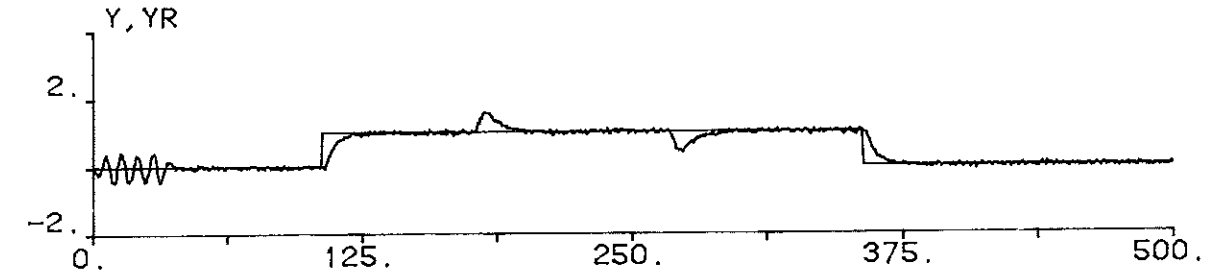
3.3 Simuleringar med mätbrus och belastningsstörning

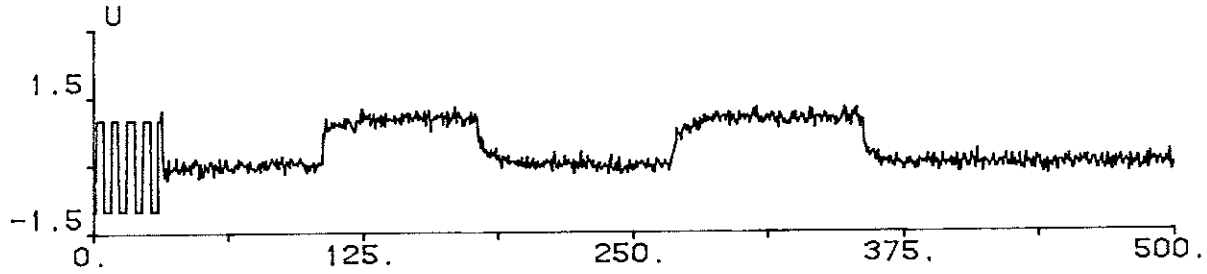
Föregående avsnitt visade att metoden med toppar och nollgenomgångar fungerade mycket bra för alla processerna när det inte fanns några störningar.Metoden ska i detta avsnitt testas med mätbrus och process 1 ska simuleras med tre olika brusnivåer.För att göra metoden mindre känslig för mätbruset,används här ett relä med hysteres.Se bilaga prog.2.

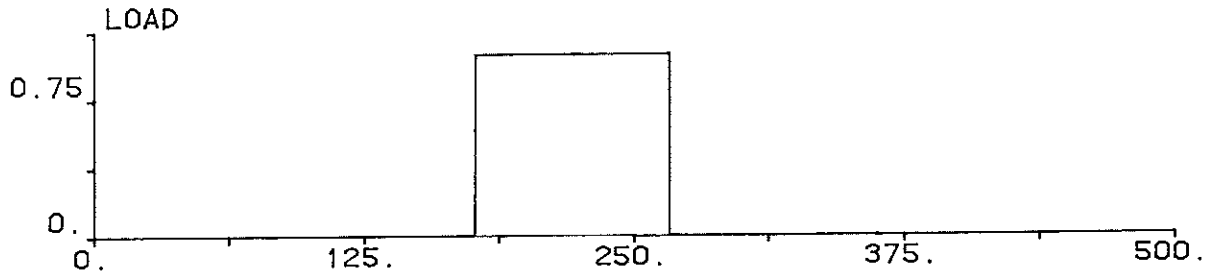
Belastningsstörningen antas kunna undvikas under skattningarna och verkar bara på systemet då inställningen är klar.Mätbrusets samplingsperiod är densamma som för alla andra diskreta system,dvs DT=0.1 och olika nivåer erhålls genom att variera standardavvikelsen på mätbruset,dvs parametern STDEV1.Eftersom reläets amplitud är fastställd på förhand,så måste hänsyn tagas till svängningsamplituden när STDEV1 skulle väljas och varieras.På grund av mätbruset utföres skattningarna under ytterligare en period,dvs IP sätts till 3.Samtidigt väntas ytterligare en period innan skattningen startar,dvs JO sätts till -3.

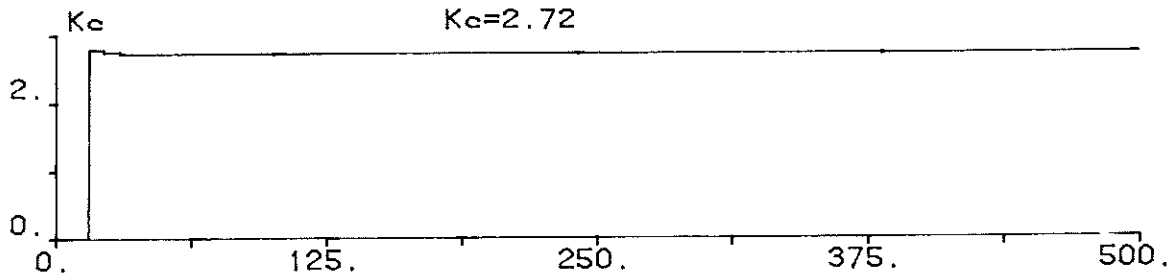
Simulering 3.3.1

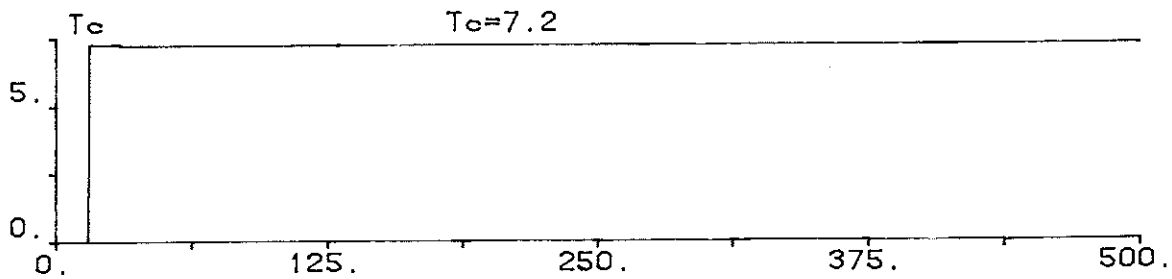
JO : -3 , IP : 3





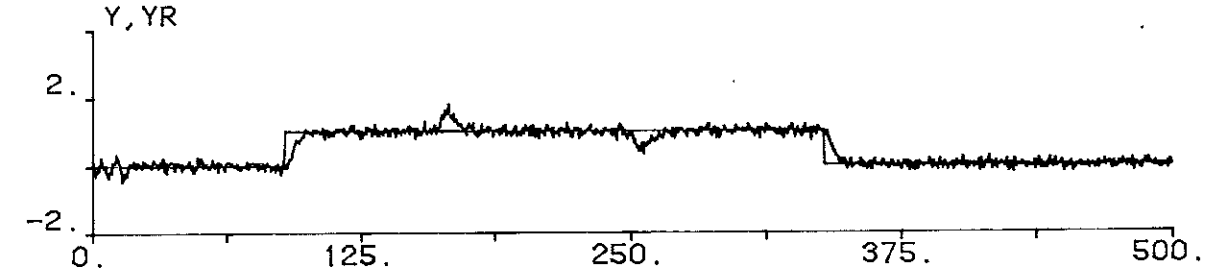


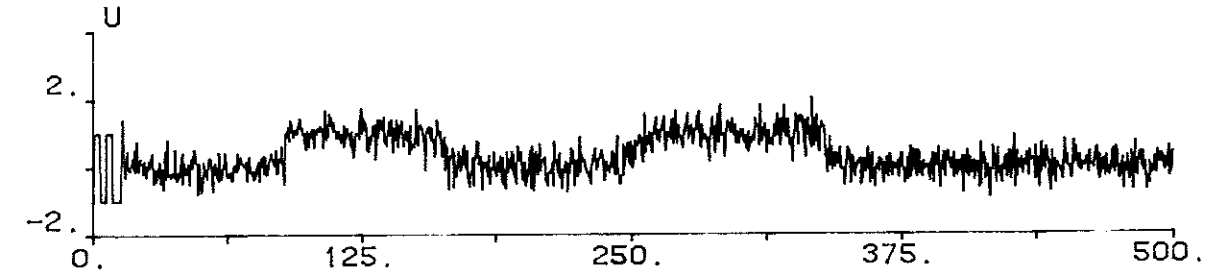


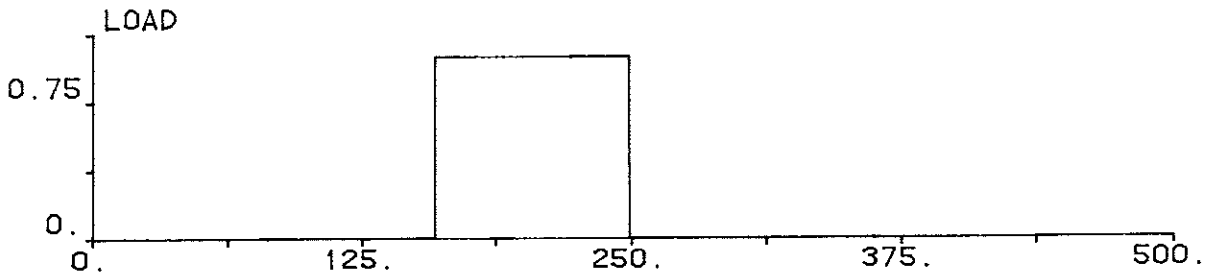


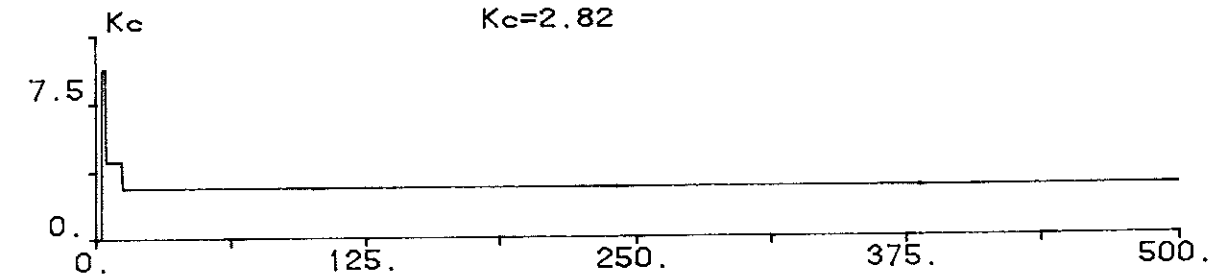
Simulering 3.3.2

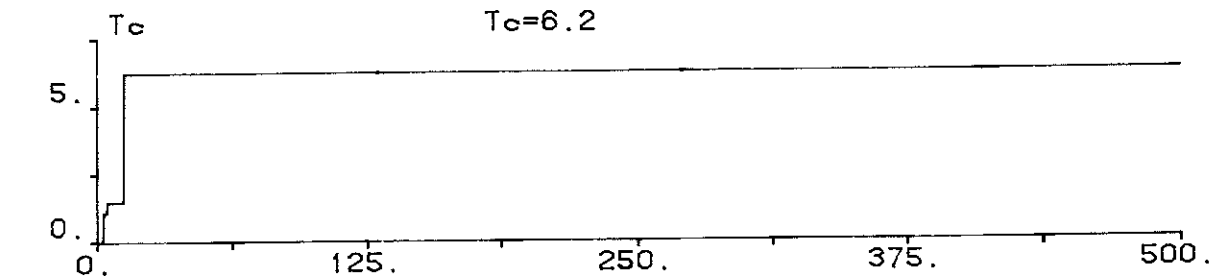
JO : -5 , IP : 3





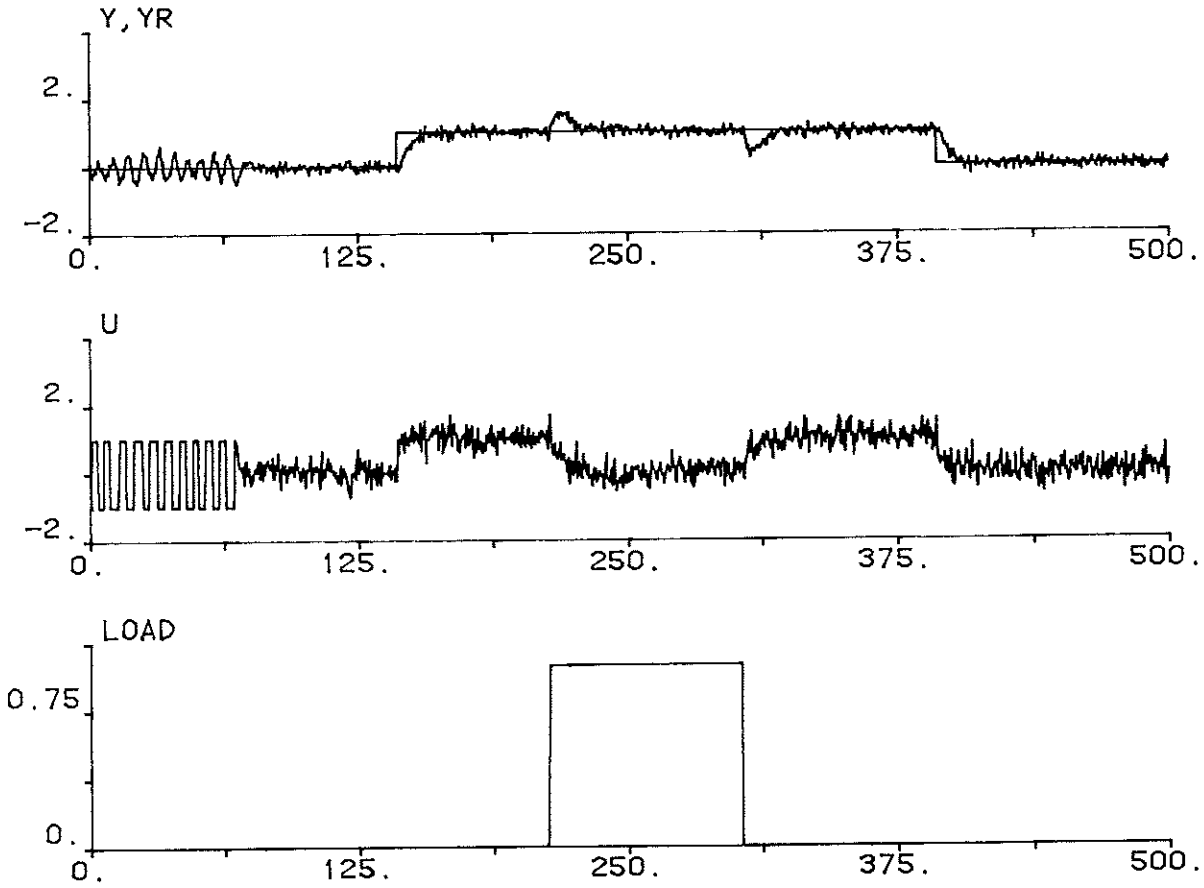


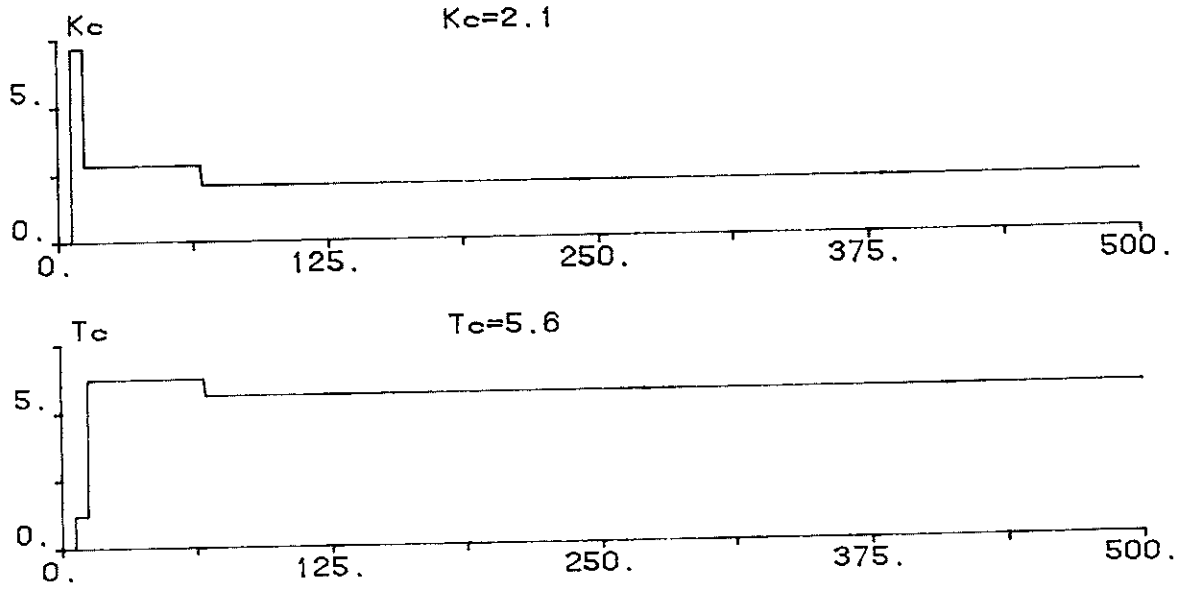




Simulering 3.3.3

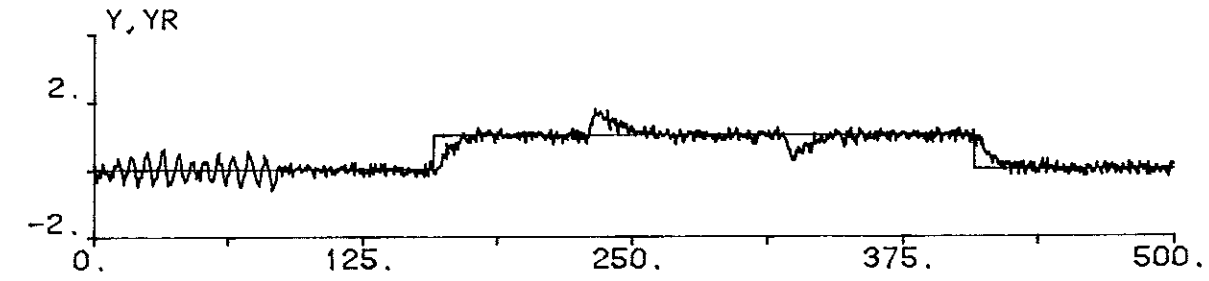
JO : -7 , IP : 3

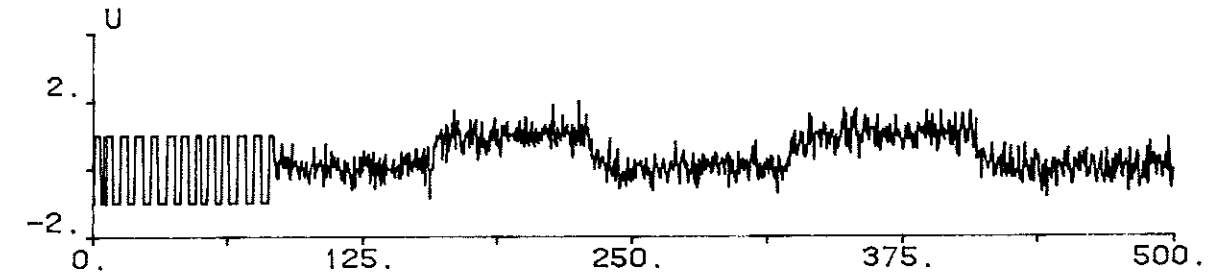


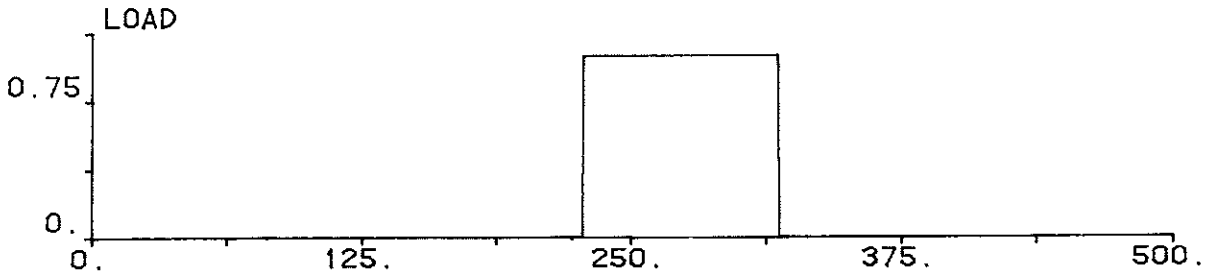


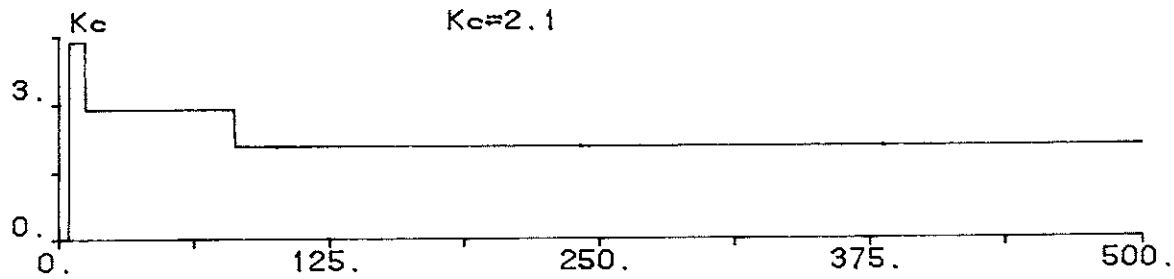
Simulering 3.3.4

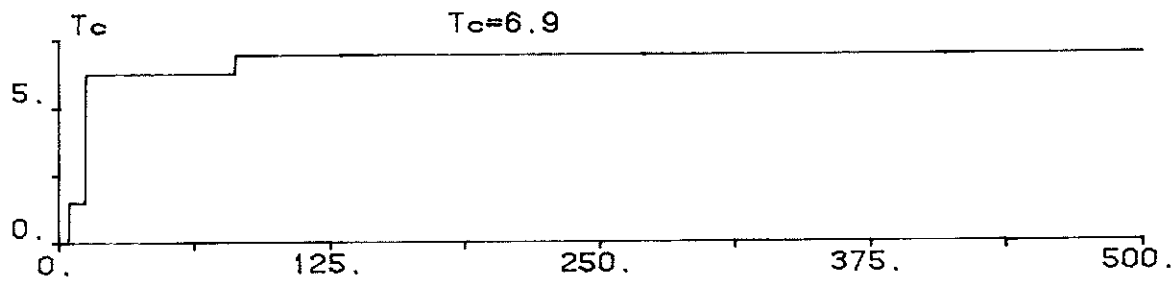
JO : -7 , IP : 3











Kommentar

Det har bekräftats här att filtreringskonstanten,N i D-delen i PID-regulatorn bör vara så liten som möjligt för att få en smidigare,dvs mindre brushaltig styrsignal.

Bredden på reläet med hysteres påverkar skattningarna.I de fall då skattningarna sker utan störningar,får man mindre värde på K_c och större värde på T_c jämfört med fallet då ett rent relä används.Detta innebär att man kommer att få mindre förstärkning och större integral- och derivatatider hos PID-regulatorn.Ju mindre d-värdet är,desto bättre är skattningarna.Med mätbrusets inverkan får man välja ett större värde på d (här d=0.1),så att känsligheten vid nollgenomgångarna minskas.Men oavsett hur stort d-värdet än väljs,kommer ändå reläet att ibland slå om,när det inte borde göra det.På grund av mätbruset, överskattas amplituden och man får då ett mindre värde på K_c ju större brusnivån är.Detta är bra ur robusthetssynpunkt. Perioden underskattas när det är ett högfrekvensmätbrus och man får då mindre värde på T_c ju högre frekvensen är.Detta innebär att man i stort sett får mindre proportional- och derivataverkan samt större integralverkan hos PID-regulatorn.På grund av de ovannämnda orsakerna,har därför skattningsalgoritmen modifierats något för att filtrera bort helt oacceptabla perioder och amplituder.Detta har gjorts på följande sätt: Bara perioder, som inte är mindre än 90% av den förra,är acceptabla, eftersom stora perioder nästan aldrig kan uppstå och det är alltid säkrare att börja med en mindre integralverkan.Se bilaga prog.5.Även om man inte vet på förhand att perioden är växande,kan denna modifikation ändå tillämpas för T_c-värden får ligga inom ett visst band.Amplituderna skattas bara under de acceptabla perioderna.Toppvärdena noteras under alla perioder och inga omställningar på deras initialvärden sker efter varje period,dvs ett tidigare toppvärde får förbli störst i en senare period,för att göra metoden robust med avseende på mätbruset.Se bilaga prog.6.

Filtreringen i början har visat sig att fungera dåligt med mätbrusets tillvaro,och orsaken är just processens stora trögheter,dvs innan utsignalen har uppnått sin stabila svängning,är signal-brus förhållandet så litet att räkningen av antalet perioder kraftigt påverkas av mätbruset i början.Detta innebär att flera perioder behövs, dvs mindre värde på JO.Ju större brusnivån och högre frekvensen är,desto mindre JO-värdet väljs.Detta kan också lösas genom att vänta en viss tid istället för räkningen i början och skattningarna sker direkt efter och JO sätts då till noll.

Tabell 3.3 visar en jämförelse mellan de "riktiga" och skattade processparametrarna och regulatorparametrarna för process 1 med tre olika brusnivåer.Det visar sig att stegsvaren i stort sett svänger in sig långsammare när brusnivån ökar. Detta beror dels på den minskade förstärkningen,dels på den ökade integraltiden.

(a).

STDEV1	K_c		T_c	
	Riktig	Skattad	Riktig	Skattad
0.03	4.0	2.72	6.3	7.2
0.09	4.0	2.1	6.3	5.6
0.1	4.0	2.1	6.3	6.9

(b).

STDEV1	Regulator					
	Riktig			Skattad		
	K	T_I	T_D	K	T_I	T_D
0.03	1.0	3.34	0.82	0.68	3.82	0.94
0.09	1.0	3.34	0.82	0.53	2.97	0.73
0.1	1.0	3.34	0.82	0.53	3.66	0.9

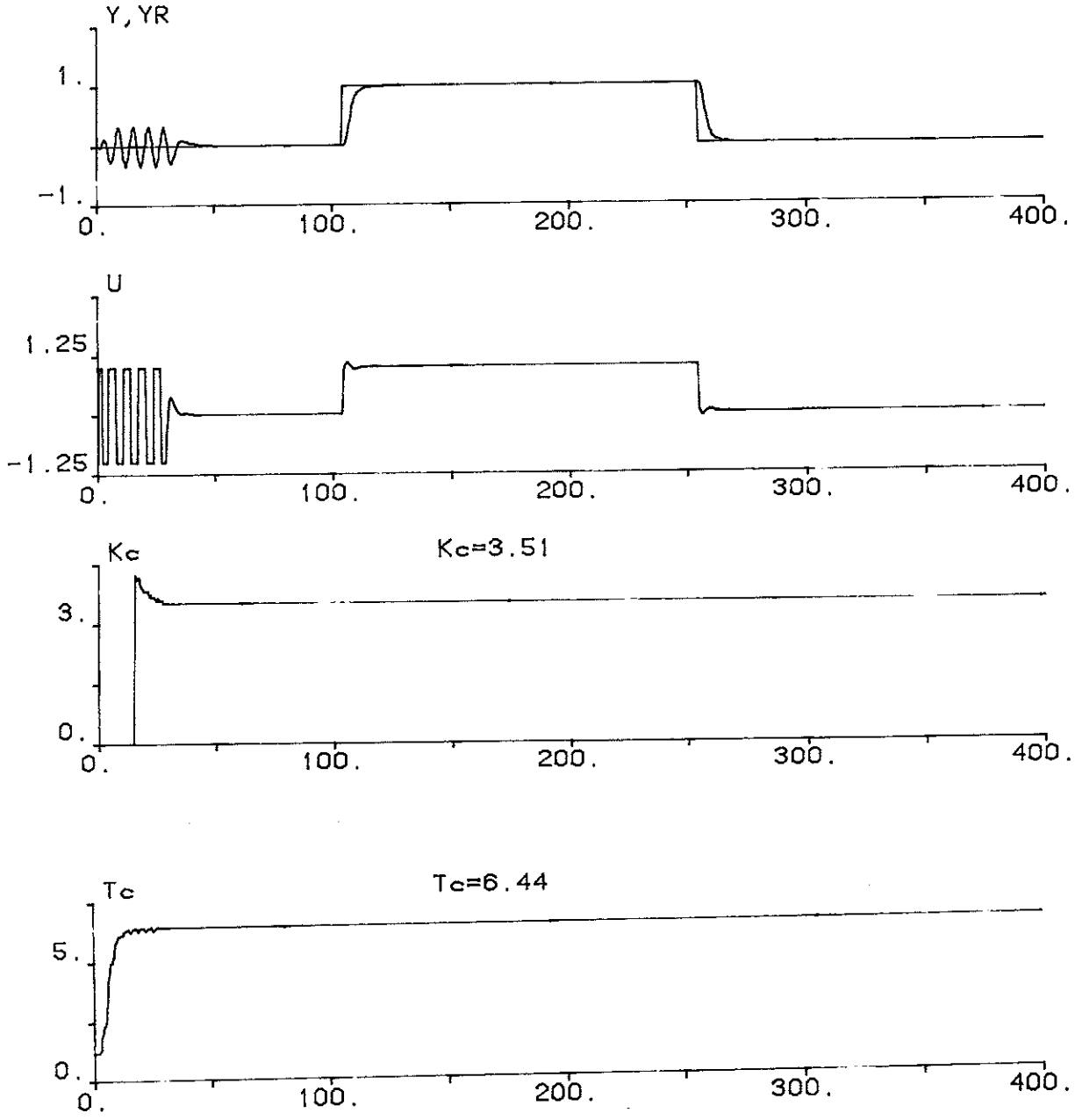
Tabell 3.3 "Riktiga" och skattade värdena på processparametrarna och regulatorparametrarna

Metoden har i stort sett visat sig att fungera bra även med mätbrusets inverkan efter modifikationen.

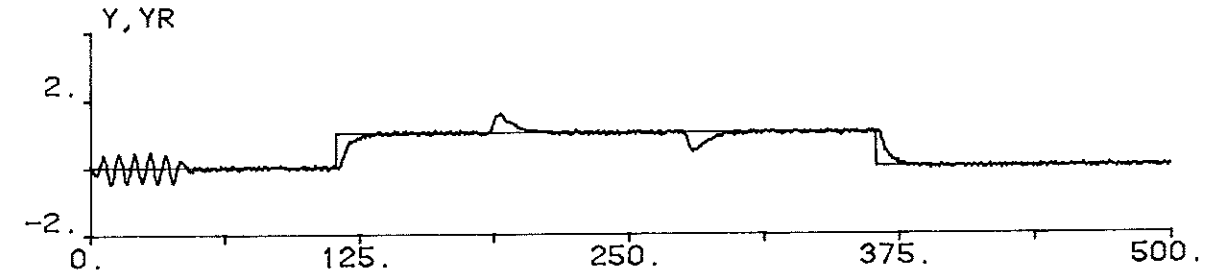
3.4 En annan metod

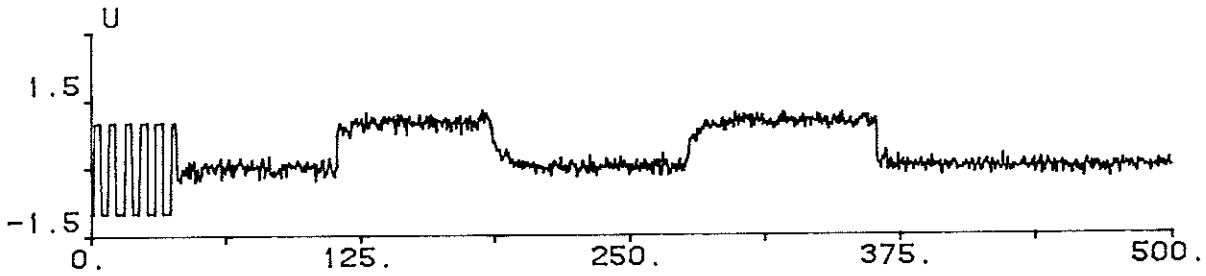
I detta avsnitt byts metoden med toppar och nollgenomgångar ut mot en annan metod,nämligen den rekursiva minsta kvadratmetoden för att skatta de två parametrarna.Även här studeras bara process 1.Hur skattningsalgoritmen ser ut,beskrivs i avsnitt 2.3.Se bilaga prog.7 och prog.8. Parametrarna,K_c och T_c noteras inte förrän en viss tid har gått,för att filtrera bort de helt oacceptabla värdena i början på grund av processens tröghet.Skattningarna sker under 2 perioder hos simulering utan störningar och 3 perioder hos simuleringar med mätbrus,efter första filtreringen.

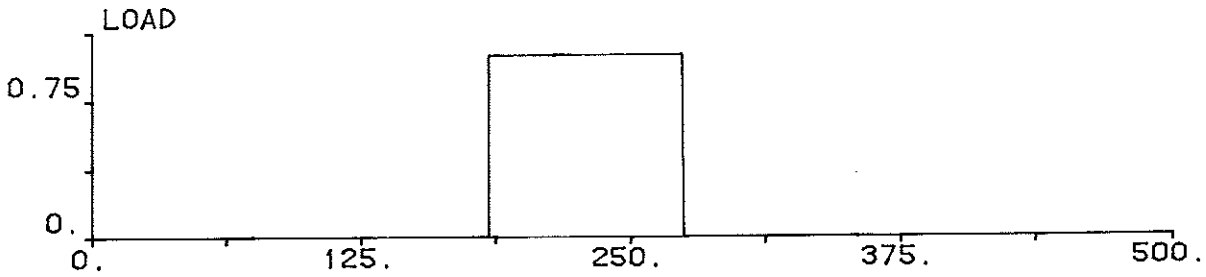
Simulering 3.4.1

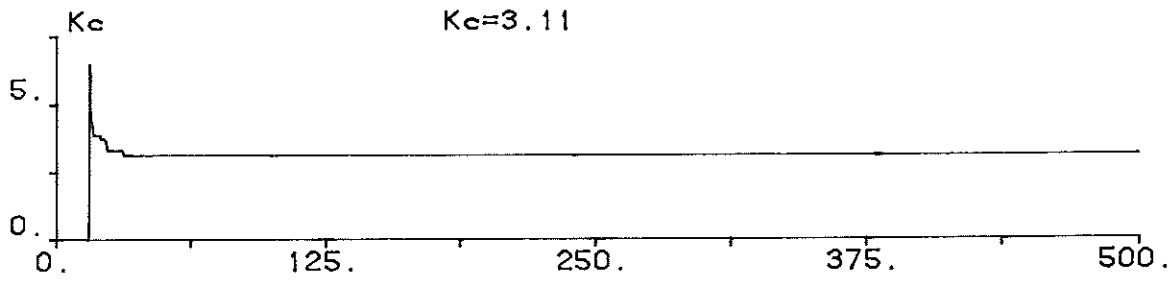


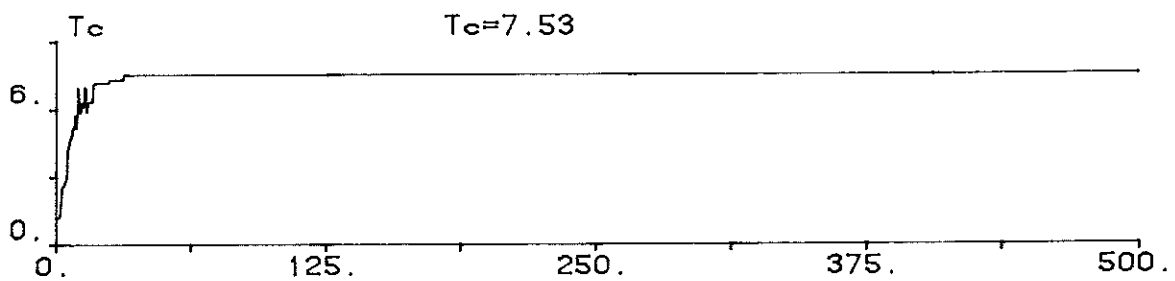
Simulering 3.4.2



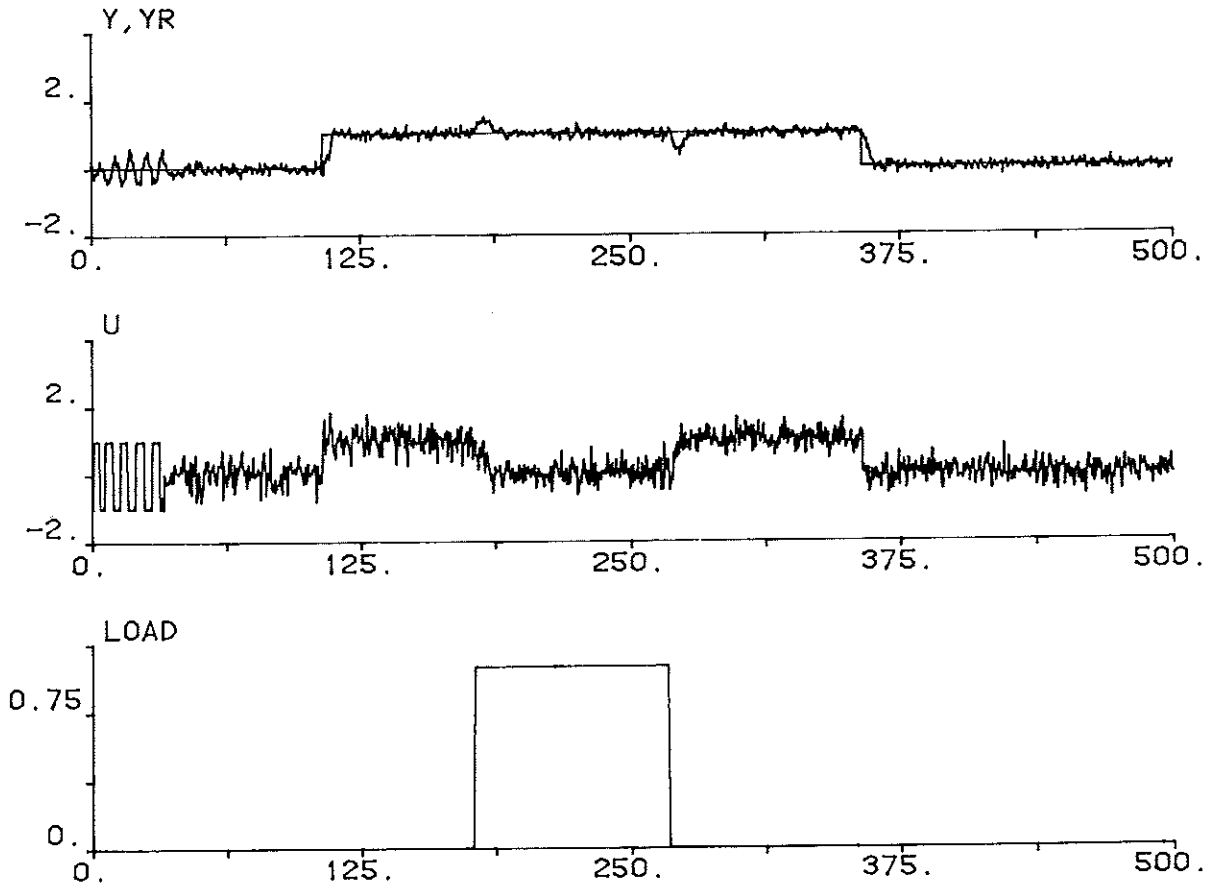


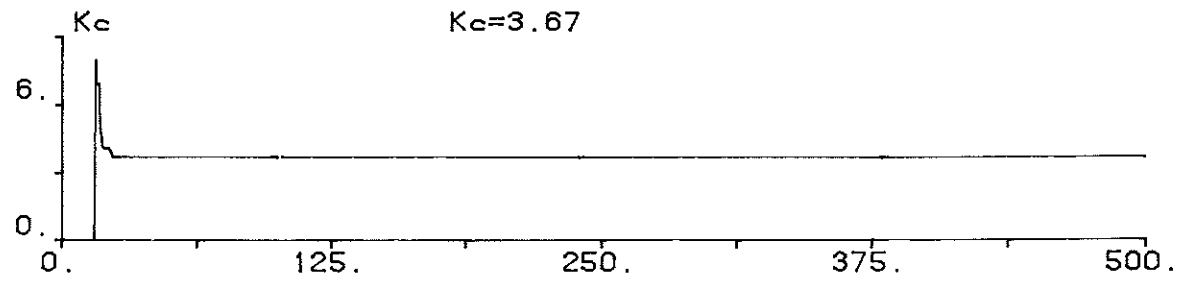


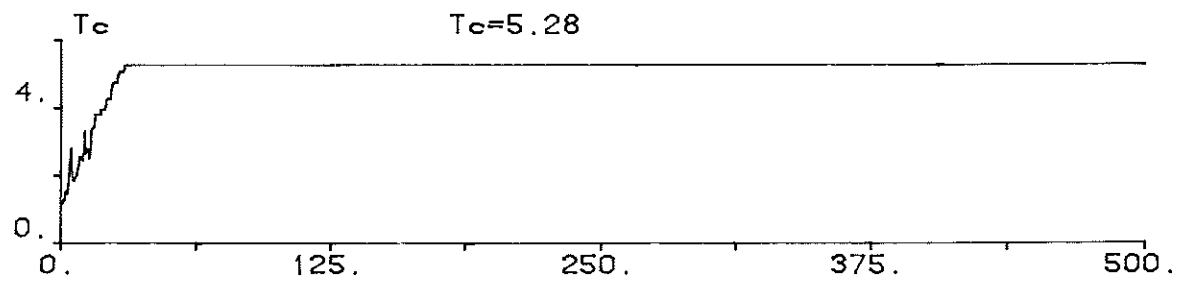




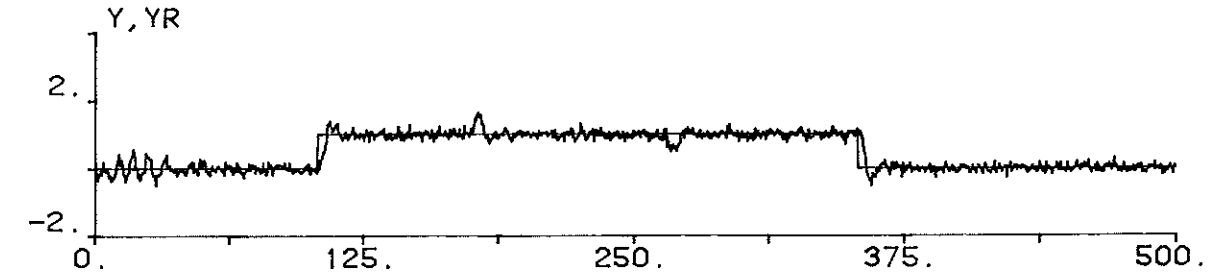
Simulering 3.4.3

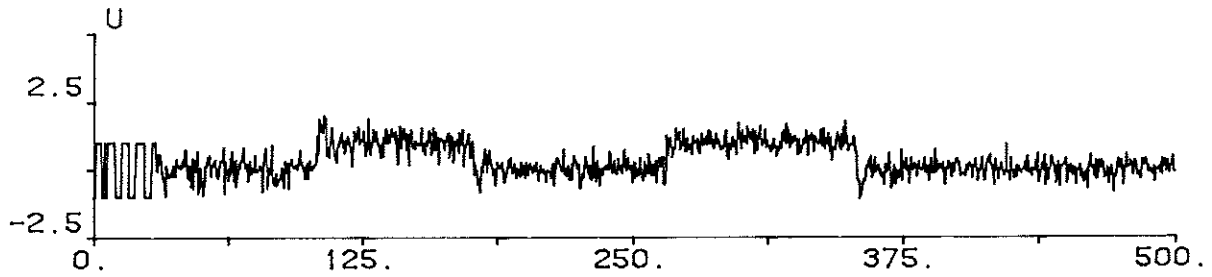


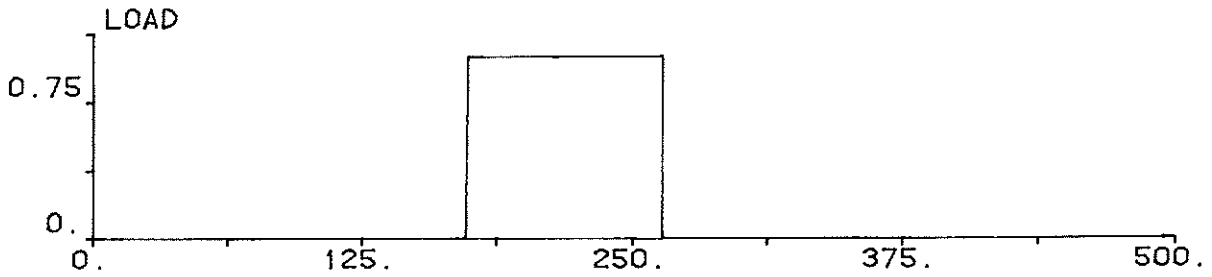


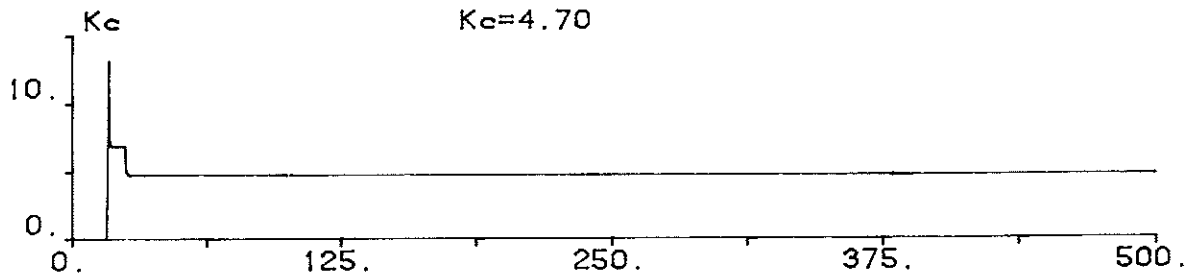


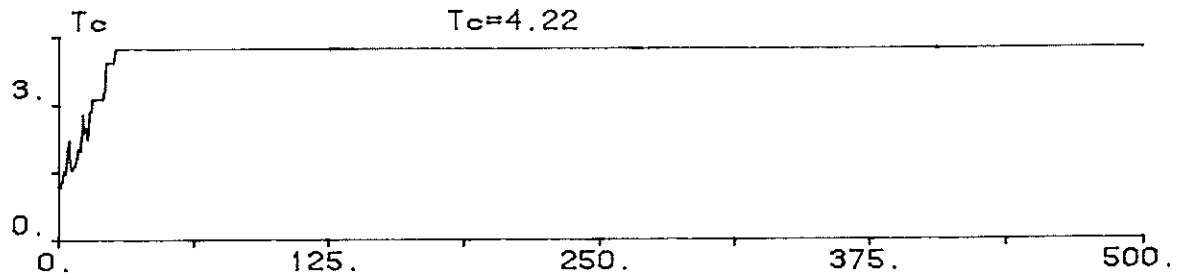
Simulering 3.4.4











8. SAMMANFATTNING

Ett generellt tredimensionellt paket har utvecklats.
Det utför:

- -perspektivtransformationer
- -hiddensurface-beräkningar
- -shading
- -ändring av betraktelsepunkten

Animering av en industrirobot har gjorts. Robotens rörelse beskrives i ett kartesiskt koordinatsystem och ledernas vinklar beräknas.

9. REFERENSER

Newman W., Sproull R.:
Principles of Interactive Computer Graphics
2:a uppl., McGraw-Hill, 1979
särskilt kapitlen 20, 22-24

Paul R.:
Robot Manipulators: Mathematics, Programming, and Control
The MIT Press
kapitlen 1-3
Observera att i denna bok skall alla matriser transponeras för att få samma beskrivning som i Newman&Sproull. Dessutom är alla rotationer definierade åt motsatt håll.

Ekman T., Karlsson J.:
Pascal för dig som kan programmera.
Studentlitteratur, 1981

Sparr G.:
Linjär algebra 1
Sigma-Tryck, TLTH Lund, 1978

Appendix 1

```
{threed.pak}

{Author:Mikael Rignell
 Date:1982-09-04}

{reference:Newman W., Sproull R.:
           Principles of Interactive Computer Graphics
           Second Edition, McGraw-Hill, 1979
           chapters 20,22-24}

.FORWARD

{These are the primitives used to make three-dimensional
images, to perform the hidden-surface algorithm and to move
the viewingpoint.}

procedure LineTo3(xm,ym,zm:real);forward;
procedure MoveTo3(xm,ym,zm:real);forward;

procedure StartPolygon(xm,ym,zm:real);forward;
procedure NextVertex(xm,ym,zm:real);forward;
procedure LastVertex(xm,ym,zm:real);forward;

procedure Translate3(Tx,Ty,Tz:real);forward;
procedure Rotate3x(fi:real);forward;
procedure Rotate3y(fi:real);forward;
procedure Rotate3z(fi:real);forward;

procedure Pan3(theta:real);forward;
procedure Tilt3(fi:real);forward;
procedure Zoom3(Dchange:real);forward;
procedure Roll3(Zchange:real);forward;

procedure NewViewport3(vxl,vxr,vyb,vyt:real);forward;

procedure FrameColor(colno:integer);forward;
procedure InternalColor(colno:integer);forward;
procedure Reset3;forward;
procedure Shade(shadeon:boolean);forward;
procedure BackFaceRemoval(bfrem:boolean);forward;

.TYPE
matrix=array[1..4,1..4]of real;
pointer=^point;
point=record
          x,y,z:real;
          pre,suc:pointer;
      end;

.VAR
K,M,V,P,S,ZERO,UNIT:matrix;
```

```
{M is the transformationmatrix from the model coordinate
system to the world coordinate system. V transforms from the
world coordinate system to the eye coordinate system. P
transforms from the eye c.s. to the homogeneous c.s.. S
choses the viewport. K=M*V*P.}
xh,yh,zh,wh, xh1,yh1,zh1,wh1, xs1,ys1,zs1, xs2,ys2,zs2:real;
xs2pre,ys2pre,zs2pre:real;
orderno:integer;
firstvertexlist,vertexlist:pointer;
depth:array[0..511,0..511] of real;
SP,DP,FP:real;
framecol,internalcol:integer;
shading:boolean;
shadeconst: real;
backfacerem:boolean;
xpixelscale3, ypixelscale3, xpixeloffset3, ypixeloffset3:real;

.PROCEDURE
procedure MultiplyVectMat(xi,yi,zi,wi:real;var Mx:matrix;
                          var xo,yo,zo,wo:real);

begin
   xo:=xi*Mx[1,1]+yi*Mx[2,1]+zi*Mx[3,1]+wi*Mx[4,1];
   yo:=xi*Mx[1,2]+yi*Mx[2,2]+zi*Mx[3,2]+wi*Mx[4,2];
   zo:=xi*Mx[1,3]+yi*Mx[2,3]+zi*Mx[3,3]+wi*Mx[4,3];
   wo:=xi*Mx[1,4]+yi*Mx[2,4]+zi*Mx[3,4]+wi*Mx[4,4];
end;

procedure MultiplyMatMat(A,B:matrix;var C:matrix);
var i,j:integer;
begin
   for i:=1 to 4 do
      for j:=1 to 4 do
         C[i,j]:=A[i,1]*B[1,j]+A[i,2]*B[2,j]+A[i,3]*B[3,j]
                                            +A[i,4]*B[4,j];

end;

procedure MultMVP;
begin
   MultiplyMatMat(M,V,K);
   MultiplyMatMat(K,P,K);
end;

procedure Clip3(var x1,y1,z1,w1,x2,y2,z2,w2:real;
                var outsideclipbox:boolean);
{From Newman&Sproull p. 360.}
   type edge=1..6;
        outcode=set of edge;
   var wc:array[1..2,1..6] of real;
       c1,c2:outcode;
       dx,dy,dz,dw,t,t1,t2:real;
       i:integer;
   procedure MakeWindowCoords(p:integer;x,y,z,w:real;
                                          var c:outcode);

      var i:integer;
      begin
```

```
        wc[p,1]:=w+x;
        wc[p,2]:=w-x;
        wc[p,3]:=w+y;
        wc[p,4]:=w-y;
        wc[p,5]:=z;
        wc[p,6]:=w-z;
        c:=[];
        for i:=1 to 6 do
           if wc[p,i]<0 then c:=c+[i];
    end;
begin
    MakeWindowCoords(1,x1,y1,z1,w1,c1);
    MakeWindowCoords(2,x2,y2,z2,w2,c2);
    if (c1*c2)=[] then
    begin
       outsideclipbox:=false;
       t1:=0;
       t2:=1;
       for i:=1 to 6 do
          if (wc[1,i]<0) or (wc[2,i]<0) then
          begin
             writeln('clipping has occured');
             t:=wc[1,i]/(wc[1,i]-wc[2,i]);
             if wc[1,i]<0 then
             begin
                if t>t1 then t1:=t;
             end
             else
             begin
                if t<t2 then t2:=t;
             end;
          end;
       if t2>=t1 then
       begin
          dx:=x2-x1;
          dy:=y2-y1;
          dz:=z2-z1;
          dw:=w2-w1;
          if t2<>1 then
          begin
             x2:=x1+t2*dx;
             y2:=y1+t2*dy;
             z2:=z1+t2*dz;
             w2:=w1+t2*dw;
          end;
          if t1<>0 then
          begin
             x1:=x1+t1*dx;
             y1:=y1+t1*dy;
             z1:=z1+t1*dz;
             w1:=w1+t1*dw;
          end;
       end;
    end
    else outsideclipbox:=true;
```

```
      {The line is completely outside the viewingbox.}
   end;

procedure Transform3(xm,ym,zm:real;var xs1,ys1,zs1,
                     xs2,ys2,zs2:real; var notwrite:boolean);
{The image is from the beginning described in the model
coordinate-system. Via the eye coordinate-system, the
homogenous coordinate-system and clipping its final
description is in the screen coordinate-system. This
procedure performs the transformation.}
var xh,yh,zh,wh,xh2,yh2,zh2,wh2,x1,y1,z1,w1,x2,y2,z2,w2:real;
begin
   MultiplyVectMat(xm,ym,zm,1,K,xh,yh,zh,wh);
   xh2:=xh;
   yh2:=yh;
   zh2:=zh;
   wh2:=wh;
   Clip3(xh1,yh1,zh1,wh1,xh,yh,zh,wh,notwrite);
   MultiplyVectMat(xh1,yh1,zh1,wh1,S,x1,y1,z1,w1);
   MultiplyVectMat(xh,yh,zh,wh,S,x2,y2,z2,w2);
   xh1:=xh2;
   yh1:=yh2;
   zh1:=zh2;
   wh1:=wh2;
   xs1:=x1/w1;
   ys1:=y1/w1;
   zs1:=z1/w1;
   xs2:=x2/w2;
   ys2:=y2/w2;
   zs2:=z2/w2;
end;

procedure BackFace;
var i:integer;
    helpp:pointer;
begin
   for i:=1 to orderno do
   begin
      helpp:=vertexlist;
      vertexlist:=vertexlist^.suc;
      PixelLine(round(helpp^.x),round(helpp^.y),
                round(vertexlist^.x),round(vertexlist^.y));
   end;
end;

procedure HiddenSurface;
{This procedure performs the depth-buffer algorithm
described in Newman&Sproull, p. 369. Lines and surfaces not
visible to the viewer are excluded. The procedure assumes
that the model consists of convex polygons. The verticies of
one polygon is inserted into a circular two way list, by the
procedures NextVertex and LastVertex. (The list is
initialized by the procedure StartPolygon.) Lastvertex calls
the hiddensurface procedure.}
label 10;
```

```
const maxintens = 15;
      shadeconst = 175;
var P,Q,R:pointer;
    PQ,QR:array[1..3] of real;
    A,B,C,D:real;
    left,up,down,helppointer:pointer;
    helporder,i,k,xhelpuptrunc,yhelpuptrunc,xhelpdowntrunc,
    Nextxtrunc,yhelpdowntrunc,xt,yt:integer;
    xmin,xhelpup,yhelpup,xhelpdown,yhelpdown,Nextx,Nexty,
    UpNumerator,UpDenominator,DeltaxUp,DeltayUp,
    DownNumerator,DownDenominator,DeltaxDown,DeltayDown,z,
    DeltaxMiddle,DeltayMiddle:real;
    DeltaUpChange,DeltaDownChange,UpNext,nextupsuc:boolean;
    intens: real;
    colorno,incolorno:integer;
    yhelpinternal:real;
    xhelpinternal:real;
    xscanup,yscanup,xscandown,yscandown:real;
    verticalline:boolean;
begin
   verticalline:=false;
   vertexlist:=firstvertexlist;
   helporder:=orderno-3;
   {orderno is the number of verticies.}
   if orderno < 3 then goto 10;
   {The equation of the polygon-plane in the form
   a*x+b*y+c*z+d=0, must be calculated. To do this you need
   one point in the plane (P), and two non-parallell vectors
   in the plane (PQ and QR).}
   P:=firstvertexlist;
   Q:=P^.suc;
   R:=Q^.suc;
   PQ[1]:=P^.x-Q^.x;
   PQ[2]:=P^.y-Q^.y;
   PQ[3]:=P^.z-Q^.z;
   QR[1]:=Q^.x-R^.x;
   QR[2]:=Q^.y-R^.y;
   QR[3]:=Q^.z-R^.z;
   C:=PQ[2]*QR[1]-PQ[1]*QR[2];
   if C < 0 then goto 10;
   {if C < 0 it means that the normal to the surface has a
   negative z-coordinate and this surface could not be seen
   finally. Due to the hardware, the positive z-direction is
   out from the screen. This is corrected when calling
   CmiDot.}
   if backfacerem then
   begin
      BackFace;
      goto 10;
   end;
   A:=PQ[3]*QR[2]-PQ[2]*QR[3];
   B:=PQ[1]*QR[3]-PQ[3]*QR[1];
   D:=-(A*P^.x+B*P^.y+C*P^.z);
   if (abs(A)<1.0E-6) and (abs(B)<1.0E-6)
                      and (abs(C)<1.0E-6) then
```

```
begin
   writeln('The first three verticies are
                                    on the same line.');
   Goto 10;
end;
for i:=1 to helporder do
begin
   vertexlist:=vertexlist^.suc;
   with vertexlist^ do
   begin
      if abs(A*x+B*y+C*z+D))0.01 then
      begin
         writeln('A point doesn''t lie in the plane.');
         Goto 10;
      end;
   end;
end;

{ shading }
if shading then
begin
   intens :=   C/shadeconst / sqrt(A*A+B*B+
                C/shadeconst*C/shadeconst);
   colorno := round(intens*maxintens);
   incolorno := round(intens*maxintens);
end
else
begin
   colorno := framecol;
   incolorno := internalcol;
end;

if abs(C))1.0E-6 then
{The plane is described in the form
     z=-(a/c)*x-(b/c)*y-d/c.}
begin
   A:=A/C;
   B:=B/C;
   D:=D/C;
end
else
begin
   A:=A*1.0E6;
   B:=B*1.0E6;
   D:=D*1.0E6;
end;
helppointer:=firstvertexlist;
left:=firstvertexlist;
xmin:=1000;
for i:=1 to orderno do
{The point with the smallest x-coordinate should be found.
If there are more than one, the first one found is chosen.}
begin
   if helppointer^.x(xmin then
   begin
```

```
        xmin:=helppointer^.x;
        left:=helppointer;
    end;
    helppointer:=helppointer^.suc;
end;
if left^.suc^.y>left^.pre^.y then
{Which way to go in the list to be on the upside
respectively on the downside of the polygon, is found out.}
begin
    up:=left^.suc;
    down:=left^.pre;
    nextupsuc:=true;
end
else
begin
    up:=left^.pre;
    down:=left^.suc;
    nextupsuc:=false;
end;
CmiCol(colorno);
DeltaUpChange:=true;
DeltaDownChange:=true;
for k:=1 to orderno-1 do
{Is the next vertex on the upside
                          or downside of the polygon?}
begin
    if up^.x<down^.x then
    begin
        Nextx:=up^.x;
        Nexty:=up^.y;
        UpNext:=true;
    end
    else
    begin
        Nextx:=down^.x;
        Nexty:=down^.y;
        UpNext:=false;
    end;
    if DeltaUpChange then
    {The slope of the upper edge have changed.}
    begin
        xhelpup:=left^.x{+0.5};
        xhelpuptrunc:=trunc(xhelpup);
        yhelpup:=left^.y{+0.5};
        yhelpuptrunc:=trunc(yhelpup);
        xscanup:=xhelpup;
        yscanup:=yhelpup;
        UpNumerator:=up^.y-left^.y;
        UpDenominator:=up^.x-left^.x;
        if abs(UpNumerator)>UpDenominator then
        begin
            DeltaxUp:=UpDenominator/abs(UpNumerator);
            DeltayUp:=UpNumerator/abs(UpNumerator);
        end
        else
```

```
    begin
       DeltaxUp:=1.0;
       if UpDenominator=0 then DeltayUp:=1 else
          DeltayUp:=UpNumerator/UpDenominator;
    end;
    DeltaUpChange:=false;
end;
if DeltaDownChange then
{The slope of the lower edge have changed.}
begin
    xhelpdown:=left^.x{+0.5};
    xhelpdowntrunc:=trunc(xhelpdown);
    yhelpdown:=left^.y{+0.5};
    yhelpdowntrunc:=trunc(yhelpdown);
    xscandown:=xhelpdown;
    yscandown:=yhelpdown;
    DownNumerator:=down^.y-left^.y;
    DownDenominator:=down^.x-left^.x;
    if abs(DownNumerator))DownDenominator then
    begin
       DeltaxDown:=DownDenominator/abs(DownNumerator);
       DeltayDown:=DownNumerator/abs(DownNumerator);
    end
    else
    begin
       DeltaxDown:=1.0;
       if DownDenominator=0 then DeltayDown:=1 else
          DeltayDown:=DownNumerator/DownDenominator;
    end;
    DeltaDownChange:=false;
end;
Nextxtrunc:=trunc(Nextx);
for xt:=trunc(left^.x) to Nextxtrunc do
{For each x-coordinate on an edge, (each xt), you have
a specific y-coordinate.But since the screen can only
take integer values, one x-value can have more than
one y-value. This happens if the slope of an edge is
more than 45 degrees. Then DeltaxUp or DeltaxDown are
less than one. For each point on the edge, the
z-coordinate is set and if the z-value is smaller than
the one in the depth-matrix,the point is set.}
begin
    if not(verticalline) then
    begin
       xscanup:=xhelpup;
       yscanup:=yhelpup;
       xscandown:=xhelpdown;
       yscandown:=yhelpdown;
    end;
    while xhelpuptrunc=xt do
    {If the slope is less than 45 degrees, these
    computations are made only once for each
    x-coordinate. If the slope is more than 45 degrees,
    they could be performed more than once.}
    begin
```

```
    z:=-A*xhelpup-B*yhelpup-D;
    if z<=depth[xt,yhelpuptrunc] then
    begin
       CmiDot(xt,yScreenMax-yhelpuptrunc);
       depth[xt,yhelpuptrunc]:=z;
    end;
    if yhelpup<yscanup then
    begin
       xscanup:=xhelpup;
       yscanup:=yhelpup;
    end;
    xhelpup:=xhelpup+DeltaxUp;
    xhelpuptrunc:=trunc(xhelpup);
    yhelpup:=yhelpup+DeltayUp;
    yhelpuptrunc:=trunc(yhelpup);
    if DeltaxUp<1 then
       if DeltayUp > 0 then
       begin
          if yhelpuptrunc > trunc(up^.y) then
                      xhelpuptrunc:=xhelpuptrunc+1;
       end
       else
       begin
          if yhelpuptrunc < trunc(up^.y) then
                      xhelpuptrunc:=xhelpuptrunc+1;
       end;
       {If we reach a vertex,
                we must stop at the right y-value.}
end;
while xhelpdowntrunc=xt do
{Same as above but with the lower edge.}
begin
    z:=-A*xhelpdown-B*yhelpdown-D;
    if z<=depth[xt,yhelpdowntrunc] then
    begin
       CmiDot(xt,yScreenMax-yhelpdowntrunc);
       depth[xt,yhelpdowntrunc]:=z;
    end;
    if yhelpdown>yscandown then
    begin
       xscandown:=xhelpdown;
       yscandown:=yhelpdown;
    end;
    xhelpdown:=xhelpdown+DeltaxDown;
    xhelpdowntrunc:=trunc(xhelpdown);
    yhelpdown:=yhelpdown+DeltayDown;
    yhelpdowntrunc:=trunc(yhelpdown);
    if DeltaxDown<1 then
       if DeltayDown < 0 then
       begin
          if yhelpdowntrunc < trunc(down^.y) then
                   xhelpdowntrunc:=xhelpdowntrunc+1;
       end
       else
       begin
```

```
            if yhelpdowntrunc ) trunc(down^.y) then
                    xhelpdowntrunc:=xhelpdowntrunc+1;
        end;
    end;
    if not(trunc(yscanup)=trunc(yscandown)) then
    begin
       DeltaxMiddle:=(xscanup-xscandown)/
                     (trunc(yscanup)-trunc(yscandown));
       DeltayMiddle:=(yscanup-yscandown)/
                     (trunc(yscanup)-trunc(yscandown));
    end;

    { scanconversion }

    if xt < Nextxtrunc then
    {Don't do scanconversion beyond the next vertex.}
    begin
       CmiCol(incolorno);
       xhelpinternal:=xscandown+DeltaxMiddle;
       yhelpinternal:=yscandown+DeltayMiddle;
       for yt:=trunc(yscandown)+1 to trunc(yscanup)-1 do
       begin
          z:=-A*xhelpinternal-B*yhelpinternal-D;
          if z<depth[xt,yt] then
          begin
             CmiDot(xt,yScreenMax-yt);
             depth[xt,yt]:=z;
          end;
          xhelpinternal:=xhelpinternal+DeltaxMiddle;
          yhelpinternal:=yhelpinternal+DeltayMiddle;
       end;
       CmiCol(colorno);
    end;
    if trunc(left^.x) = Nextxtrunc then
    begin
       if (trunc(left^.x) = trunc(left^.pre^.x)) or
          (trunc(left^.x) = trunc(left^.suc^.x))
                             then verticalline:=true
                             else verticalline:=false;
    end
    else
    begin
       verticalline:=false;
    end;
end;
if UpNext then
{The vertex with the next x-coordinate in turn
                                  is on the upside.}
begin
    left:=up;
    if nextupsuc then up:=up^.suc
                 else up:=up^.pre;
    DeltaUpChange:=true;
end
else
```

```
        {Next vertex on the downside.}
        begin
           left:=down;
           if nextupsuc then down:=down^.pre
                        down:=down^.suc;
           DeltaDownChange:=true;
        end;
        z:=-A*Nextx-B*Nexty-D;
        if z < depth[Nextxtrunc,trunc(Nexty)] then
        {Due to the truncation, you might miss a vertexpoint.}
        begin
           CmiDot(Nextxtrunc,yScreenMax-trunc(Nexty));
           depth[Nextxtrunc,trunc(Nexty)]:=z;
        end;
     end;
10:for i:=1 to orderno-1 do
     begin
        vertexlist:=vertexlist^.suc;
        dispose(vertexlist^.pre);
     end;
     dispose(vertexlist);
end;

procedure insert(e:pointer);
{Inserts a point into the cicular two way list.}
begin
     if firstvertexlist=nil then
     begin
        firstvertexlist:=e;
        firstvertexlist^.pre:=firstvertexlist;
        firstvertexlist^.suc:=firstvertexlist;
     end
     else
     begin
        firstvertexlist^.pre^.suc:=e;
        e^.pre:=firstvertexlist^.pre;
        e^.suc:=firstvertexlist;
        firstvertexlist^.pre:=e;
     end;
end;

procedure LineTo3(xm,ym,zm:real);
{A line is drawn from the point in the last MoveTo3
LineTo3 to the one given here.}
var xs1r,ys1r,xs2r,ys2r:integer;
     lineoutside:boolean;
begin
     Transform3(xm,ym,zm,xs1,ys1,zs1,xs2,ys2,zs2,lineoutside);
     if not(lineoutside) then
     begin
        xs1r:=round(xs1);
        ys1r:=round(ys1);
        xs2r:=round(xs2);
        ys2r:=round(ys2);
        PixelLine(xs1r,ys1r,xs2r,ys2r);
```

```
   end;
end;

procedure MoveTo3(xm,ym,zm:real);
{The next line should begin at this point.}
begin
   MultiplyVectMat(xm,ym,zm,1,K,xh,yh,zh,wh);
   xh1:=xh;
   yh1:=yh;
   zh1:=zh;
   wh1:=wh;
end;

procedure StartPolygon(xm,ym,zm:real);
{Initiates the list and performs the first steps (those
before the clipping), in the transformation from
model-coordinates to screen-coordinates. The
clipping-algorithm must have both the end-points of a line.}
begin
   new(firstvertexlist);
   firstvertexlist:=nil;
   orderno:=0;
   MultiplyVectMat(xm,ym,zm,1,K,xh,yh,zh,wh);
   xh1:=xh;
   yh1:=yh;
   zh1:=zh;
   wh1:=wh;
   xs2pre:=1000;
   ys2pre:=1000;
   zs2pre:=1000;
end;

procedure NextVertex(xm,ym,zm:real);
{Puts one (or sometimes two) point(s) into the list after
transformation. The coordinates are translated and scaled as
to be visible on the screen. The y-coordinates are also
corrected as not to be shown upside down.}
label 7;
var e:pointer;
    noline:boolean;
begin
   Transform3(xm,ym,zm,xs1,ys1,zs1,xs2,ys2,zs2,noline);
   if noline then goto 7;

   if (xs1<>xs2pre) or (ys1<>ys2pre) or (zs1<>zs2pre) then
 {If the startingpoint of the new line isn't the same as the
 last point in the previous line, the startingpoint is
 inserted.}
   begin
      new(e);
      with e^ do
      begin
         x:=xs1;
         y:=ys1;
         z:=zs1;
```

```
        end;
        insert(e);
        orderno:=orderno+1;
    end;
    new(e);
    with e^ do
    begin
        x:=xs2;
        y:=ys2;
        z:=zs2;
    end;
    insert(e);
    orderno:=orderno+1;
    xs2pre:=xs2;
    ys2pre:=ys2;
    zs2pre:=zs2;
 7:    ;
end;

procedure LastVertex(xm,ym,zm:real);
{Puts the last vertex into the list and calls HiddenSurface.}
begin
    NextVertex(xm,ym,zm);
    HiddenSurface;
end;

procedure Translate3(Tx,Ty,Tz:real);
{Translates the model coordinate-system.}
var T:matrix;
     i,j:integer;
begin
    T:=UNIT;
    T[4,1]:=Tx;
    T[4,2]:=Ty;
    T[4,3]:=Tz;
    MultiplyMatMat(T,M,M);
    MultMVP;
end;

procedure Rotate3x(fi:real);
{The model coordinate-system is rotated around the x-axis.}
var T:matrix;
     i,j:integer;
     cosfi,sinfi:real;
begin
    fi:=3.14159265*fi/180;
    cosfi:=cos(fi);
    sinfi:=sin(fi);
    T:=ZERO;
    T[1,1]:=1;
    T[2,2]:=cosfi;
    T[2,3]:=-sinfi;
    T[3,2]:=sinfi;
    T[3,3]:=cosfi;
    T[4,4]:=1;
```

```
   MultiplyMatMat(T,M,M);
   MultMVP;
end;

procedure Rotate3y(fi:real);
{Rotation of the model coordinatesystem around the y-axis.}
var T:matrix;
    i,j:integer;
    cosfi,sinfi:real;
begin
   fi:=3.14159265*fi/180;
   cosfi:=cos(fi);
   sinfi:=sin(fi);
   T:=ZERO;
   T[1,1]:=cosfi;
   T[1,3]:=sinfi;
   T[2,2]:=1;
   T[3,1]:=-sinfi;
   T[3,3]:=cosfi;
   T[4,4]:=1;
   MultiplyMatMat(T,M,M);
   MultMVP;
end;

procedure Rotate3z(fi:real);
{Rotates the model coordinate-system around the z-axis.}
var T:matrix;
    i,j:integer;
    cosfi,sinfi:real;
begin
   fi:=3.14159265*fi/180;
   cosfi:=cos(fi);
   sinfi:=sin(fi);
   T:=ZERO;
   T[1,1]:=cosfi;
   T[1,2]:=-sinfi;
   T[2,1]:=sinfi;
   T[2,2]:=cosfi;
   T[3,3]:=1;
   T[4,4]:=1;
   MultiplyMatMat(T,M,M);
   MultMVP;
end;

procedure SetViewBox(SP,DP,FP:real);
begin
   P[3,3]:=SP/(DP*(1-DP/FP));
   P[3,4]:=SP/DP;
   P[4,3]:=-SP/(1-DP/FP);
   MultMVP;
end;

procedure Pan3(theta:real);
{The viewer rotates around his vertical axis.}
var thetarad,Csin,Ccos:real; H:matrix;
```

```
begin
   thetarad:=3.141592654*theta/180;
   Csin:=sin(thetarad);
   Ccos:=cos(thetarad);
   H:=UNIT;
   H[1,1]:=Ccos;
   H[1,3]:=Csin;
   H[3,1]:=-Csin;
   H[3,3]:=Ccos;
   MultiplyMatMat(V,H,V);
   MultMVP;
end;

procedure Tilt3(fi:real);
{The viewer rotates around the axis parallell
                                        to the horizon.}
var firad,Ksin,Kcos:real; i:integer; H:matrix;
begin
   firad:=-3.141592654*fi/180;
   Ksin:=sin(firad);
   Kcos:=cos(firad);
   H:=UNIT;
   H[2,2]:=Kcos;
   H[2,3]:=-Ksin;
   H[3,2]:=Ksin;
   H[3,3]:=Kcos;
   MultiplyMatMat(V,H,V);
   MultMVP;
end;

procedure Roll3(Zchange:real);
{The viewer moves forwards or backwards.}
var H:matrix;
begin
   H:=UNIT;
   H[4,3]:=-Zchange;
   MultiplyMAtMat(V,H,V);
   MultMVP;
end;

procedure Zoom3(Dchange:real);
begin
   DP:=DP+Dchange;
   SetViewBox(SP,DP,FP);
end;

procedure NewViewPort3(vxl,vxr,vyb,vyt:real);
begin
   S[1,1]:=(vxr-vxl)/2*xpixelscale3;
   S[2,2]:=(vyt-vyb)/2*ypixelscale3;
   S[4,1]:=(vxl+vxr)/2+xpixeloffset3;
   S[4,2]:=(vyb+vyt)/2+ypixeloffset3;
end;

procedure FrameColor(colno:integer);
```

```
begin
   framecol:=colno;
end;

procedure InternalColor(colno:integer);
begin
   internalcol:=colno;
end;

procedure Shade(shadeon: boolean);
begin
   shading := shadeon;
   if shadeon then BlueShade
              else TwoPlanes;
end;

procedure BackFaceRemoval(bfrem:boolean);
begin
   backfacerem:=bfrem;
end;

procedure Reset3;
begin
   M:=UNIT;
   MultMVP;
end;

procedure Setdepth;
var i,j:integer;
begin
   for i:=0 to 511 do depth[0,i]:=10;
   for i:=1 to 511 do depth[i]:=depth[0];
end;

procedure DefineScreen3;
  const
  CmiXsize = 511;  CmiYsize = 511;
  screenwidth = 280;  screenheight = 200;

  begin
  xpixelscale3 := (CmiXSize + 1) / ScreenWidth;
  ypixelscale3 := (CmiYSize + 1) / ScreenHeight;

  xpixeloffset3 := CmiXSize / 2;
  ypixeloffset3 := CmiYSize / 2;
  end;

procedure InitThreeDim;
var i,j:integer;
begin
   for i:=1 to 4 do
   for j:=1 to 4 do
   begin
      ZERO[i,j]:=0;
   end;
```

```
    UNIT:=ZERO;
    for i:=1 to 4 do UNIT[i,i]:=1;
    M:=UNIT;
    V:=ZERO;
    P:=ZERO;
    S:=ZERO;
    SP:=75;
    DP:=300;
    FP:=800;
    V[1,1]:=1;
    V[2,3]:=1;
    V[3,2]:=1;
    V[4,4]:=1;
    Roll3(-400);
    P[1,1]:=1;
    P[2,2]:=1;
    SetViewBox(SP,DP,FP);
    DefineScreen3;
    S[3,3]:=1;
    S[4,4]:=1;
    NewViewPort3(-75,75,-75,75);
    MultMVP;
    shading:=false;
    backfacerem:=false;
    internalcol:=0;
    framecol:=1;
    CmiErase;
end;

.INIT
InitThreeDim;

.END
```

```
{robot.pak describes an industrial robot and
 its movement in a cartesian coordinate system}

{Author: Mikael Rignell
 Date: 1982-09-04}

{Reference:
   Paul R.:
   Robot Manipulators: Mathematics, Programming and Control
   The MIT Press}

.PROGRAM

program Robot(input,output,outfile);


.CONST
pi=3.1415926534;


.VAR
outfile:text;
ch:char;
number,newnumber,Znewnumber:real;
i,j,forhelp,cno:integer;
vxl,vxr,vyb,vyt:real;
joystickvalues:joysticktype;
px,py,pz,nx,ny,nz:real;
pxold,pyold,pzold, nxold,nyold,nzold:real;
a1,a2,a3,a4,a5,z0:real;
theta1,theta2,theta3,theta4,theta5,theta6:real;
model,notjumpout,notmoving:boolean;
shad:boolean;
bafare:boolean;


.PROCEDURE

function arctan2(y,x:real):real;
var fi:real;
begin
   if x=0 then
   begin
      if y>0 then fi:=pi/2
             else fi:=-pi/2;
   end
   else
   begin
      fi:=arctan(y/x);
      if (x<0) and (y>0) then fi:=fi+pi;
      if (x<0) and (y<0) then fi:=fi-pi;
   end;
   arctan2:=fi;
end;
```

```
procedure box(x2,y2,z2:real);
begin
    MoveTo3(0,0,0);
    LineTo3(0,0,z2);
    LineTo3(x2,0,z2);
    LineTo3(x2,0,0);
    LineTo3(0,0,0);
    LineTo3(0,y2,0);
    LineTo3(0,y2,z2);
    LineTo3(x2,y2,z2);
    LineTo3(x2,y2,0);
    LineTo3(0,y2,0);
    MoveTo3(0,y2,z2);
    LineTo3(0,0,z2);
    MoveTo3(x2,0,z2);
    LineTo3(x2,y2,z2);
    MoveTo3(x2,y2,0);
    LineTo3(x2,0,0);
end;

procedure Rectanglex(xb1,yb1,zb1,yb3,zb3:real);
begin
    StartPolygon(xb1,yb1,zb1);
    NextVertex(xb1,yb3,zb1);
    NextVertex(xb1,yb3,zb3);
    LastVertex(xb1,yb1,zb3);
end;

procedure Rectangley(xb1,yb1,zb1,xb3,zb3:real);
begin
    StartPolygon(xb1,yb1,zb1);
    NextVertex(xb1,yb1,zb3);
    NextVertex(xb3,yb1,zb3);
    LastVertex(xb3,yb1,zb1);
end;

procedure Rectanglez(xb1,yb1,zb1,xb3,yb3:real);
begin
    StartPolygon(xb1,yb1,zb1);
    NextVertex(xb3,yb1,zb1);
    NextVertex(xb3,yb3,zb1);
    LastVertex(xb1,yb3,zb1);
end;

procedure Block(length,width,height:real);
begin
    if (notmoving or bafare) then
    begin
        Rectanglex(0,0,0,width,height);
        Rectanglex(length,width,0,0,height);
        Rectangley(0,0,0,length,height);
        Rectangley(0,width,height,length,0);
        Rectanglez(0,0,0,length,width);
        Rectanglez(length,0,height,0,width);
    end
```

```
    else
    begin
       box(length,width,height);
    end;
end;

procedure NewAngles(var outsidereach:boolean);
{used when moving the robot with 6 degrees of freedom
 Inparameters : px py pz (position)
                nx ny nz (direction)
 Outparameters : theta1,theta2,theta3,theta4,theta5 }
label 5,6;
var pxp,pzp, pxpp,pzpp, pxh,pyh,pzh,nxh,nyh,nzh:real;
    norm:real;
    alfa1,alfa2,alfa3,alfa4,alfa5,alfa234,alfa23:real;
    C1,S1,C2,S2,C23,S23,C234,S234,C3,S3,S4,C4,C5,S5:real;
begin
   outsidereach:=false;
   norm:=sqrt(sqr(nx)+sqr(ny)+sqr(nz));
   nx:=nx/norm;
   ny:=ny/norm;
   nz:=nz/norm;

   alfa1:=arctan2((a5*ny-py),(px-a5*nx));
   S1:=sin(alfa1);
   C1:=cos(alfa1);
   {alfa1 is the robot's rotation around the foundation.
         -pi<= alfa1 <=+pi }

   S5:=-(S1*px+C1*py)/a5;
   if abs(S5) > 1 then goto 5;
   C5:=sqrt(1-sqr(S5));
   alfa5:=arctan2(S5,C5);
   {alfa5 is the angle between the two outer arms. (Rotation
    around an z-axis).      -pi/2 <= alfa5 <= +pi/2 }

   alfa234:=arctan2(nz,C1*nx-S1*ny);
   C234:=cos(alfa234);
   S234:=sin(alfa234);
   {alfa234=alfa2+alfa3+alfa4. -pi <= alfa234 <= +pi }

   pxp:=C1*px-S1*py-(C5*a5+a4)*C234;
   pzp:=pz-z0-(C5*a5+a4)*S234-a1;
   S3:=(sqr(pxp)+sqr(pzp)-sqr(a3)-sqr(a2))/(2*a2*a3);
   if abs(S3) > 1 then goto 5;
   C3:=sqrt(1-sqr(S3));
   alfa3:=arctan2(S3,C3);
   if alfa3 < -3*pi/8 then goto 5;
   {alfa3 is the angle between the biggest parts of the arm.
   If S3>1 it means that the point cannot be reached, at
   least not with the desired direction. -3*pi/8 <= alfa3 <=
   +pi/2 . The lower limit because the outer parts of the
   arm shouldn't fold into the first part.}

   pxpp:=C234*(C1*px-S1*py)+S234*(pz-z0-a1)-C5*a5-a4;
```

```
    pzpp:=-S234*(C1*px-S1*py)+C234*(pz-z0-a1);
    alfa4:=arctan2(pxpp*a2*C3-pzpp*(a2*S3+a3),
                                pzpp*a2*C3+pxpp*(a2*S3+a3));
    if abs(alfa4) > 9*pi/10 then goto 5;
    {alfa4 is is the angle between the second
     and the third part of the arm.
     -9*pi/10 <= alfa4 <= +9*pi/10 }

    alfa2:=alfa234-alfa4-alfa3;
    if alfa2<-pi/2 then alfa2:=alfa2+pi;
    if alfa2>pi/2 then alfa2:=alfa2-pi;
    S2:=sin(alfa2);
    C2:=cos(alfa2);
    {alfa2 is the angle between the body
     and the first part of the arm.
     -pi/2 <= alfa2 <= +pi/2 }

    C23:=cos(alfa2+alfa3);
    S23:=sin(alfa2+alfa3);
    {Calculate the exact expressions for the coordinates
                                              and the directions.}
    pxh:=C5*a5*C234*C1+a4*C234*C1+a3*C23*C1-a5*S5*S1-a2*S2*C1;
    pyh:=-C5*a5*C234*S1-a4*C234*S1-a3*C23*S1-a5*S5*C1+a2*S2*S1;
    pzh:=C5*a5*S234+a4*S234+a3*S23+a2*C2+a1+z0;
    nxh:=C5*C234*C1-S5*S1;
    nyh:=-C5*C234*S1-S5*C1;
    nzh:=C5*S234;

    if (abs(px-pxh) > 0.1) or (abs(py-pyh) > 0.1) or
       (abs(pz-pzh) > 0.1) or (abs(nx-nxh) > 0.05) or
       (abs(ny-nyh) > 0.05) or (abs(nz-nzh) > 0.05)
                                                  then goto 5;

    theta1:=180*alfa1/pi;
    theta2:=180*alfa2/pi;
    theta3:=180*alfa3/pi;
    theta4:=180*alfa4/pi;
    theta5:=180*alfa5/pi;
    goto 6;
  5:outsidereach:=true;
    {If we reach point 5 it means that the desired point
    cannot be reached. Then we must give back px,py,pz or
    nx,ny,nz values corresponding to a point that can be
    reached, as not to be trapped.}

  6:  ;
end;

procedure InitPos;
begin
    nx:=1;
    ny:=0;
    nz:=0;
    px:=64;
    py:=0;
```

```
    pz:=40;
    z0:=-40;
    a1:=40;
    a2:=40;
    a3:=30;
    a4:=18;
    a5:=16;
    theta1:=0;
    theta2:=0;
    theta3:=0;
    theta4:=0;
    theta5:=0;
    theta6:=0;
end;

procedure Robot6(fi1,fi2,fi3,fi4,fi5,fi6:real);
begin
    Reset3;
    Translate3(-21,-21,z0-13);
    Block(42,42,13);  {foundation}

    Translate3(21,21,13);
    Rotate3z(fi1);
    Translate3(-13,-13,0);
    Block(26,26,a1+5);  {body}

    Translate3(13,13,a1);
    Rotate3y(fi2);
    Translate3(-7,-7,-5);
    Block(14,14,a2+10);  {first arm}

    Translate3(7,7,a2+5);
    Rotate3y(fi3);
    Translate3(-5,-6,-6);
    Block(a3+9,12,12);  {second arm}

    Translate3(a3+5,6,6);
    Rotate3y(fi4);
    Translate3(-4,-5,-5);
    Block(a4+6,10,10);  {third arm}

    Translate3(a4+4,5,5);
    Rotate3z(fi5);
    Translate3(-2,-3,-3);
    Block(a5-4,6,6);  {fourth arm}
    Translate3(a5-4,3,3);
    Rotate3x(fi6);
    Translate3(0,-0.5,-0.5);
    Block(6,1,1);     {tip}
    Translate3(6,0.5,0.5);
end;

procedure MoveRobot(cx,cy,cz,dx,dy,dz:real);
{All motion of the robot must be done
 through this procedure.}
```

```
var oldtheta1,oldtheta2,oldtheta3,oldtheta4,
    oldtheta5,oldtheta6:real;
    outsidereachmr:boolean;
begin
   outsidereachmr:=false;
   oldtheta1:=theta1;
   oldtheta2:=theta2;
   oldtheta3:=theta3;
   oldtheta4:=theta4;
   oldtheta5:=theta5;
   px:=cx;
   py:=cy;
   pz:=cz;
   nx:=dx;
   ny:=dy;
   nz:=dz;
   NewAngles(outsidereachmr);
   if outsidereachmr then
   begin
      theta1:=oldtheta1;
      theta2:=oldtheta2;
      theta3:=oldtheta3;
      theta4:=oldtheta4;
      theta5:=oldtheta5;
      px:=pxold;
      py:=pyold;
      pz:=pzold;
      nx:=nxold;
      ny:=nyold;
      nz:=nzold;
   end;
   CmiErase;
   Robot6(theta1,theta2,theta3,theta4,theta5,theta6);
   SwapPlane;
end;

procedure Weld;
var deltax,deltay:array[1..4] of real;
    i,j:integer;
    cx,cy,cz1,cz2,dx,dy,dz:real;
begin
   deltax[1]:=0;
   deltax[2]:=-4;
   deltax[3]:=0;
   deltax[4]:=4;
   deltay[1]:=4;
   deltay[2]:=0;
   deltay[3]:=-4;
   deltay[4]:=0;
   cx:=40;
   cy:=-40;
   cz1:=-40;
   cz2:=-53;
   dx:=0.00;
   dy:=0.00;
```

```
    dz:=-1;
    for i:=1 to 4 do
    begin
       for j:=1 to 20 do
       begin
          cx:=cx+deltax[i];
          cy:=cy+deltay[i];
          MoveRobot(cx,cy,cz1,dx,dy,dz);
          MoveRobot(cx,cy,cz2,dx,dy,dz);
          MoveRobot(cx,cy,cz1,dx,dy,dz);
       end;
    end;
    MoveRobot(64,0,40,1,0,0);
end;

.MAIN
    begin
       TwoPlanes;
       InitPos;
       forhelp:=10;
       bafare:=false;
       shad:=false;
       notjumpout:=false;
       notmoving:=false;
       model:=true;
       CmiCol(1);
       DrawBackground;
       MoveRobot(px,py,pz,nx,ny,nz);
       while true do
       begin
          write(')');
          read(ch);
          if ch='l' then
          begin
             writeln('All commands are in the form : ',
                          'small character, real number');
             writeln('a : hardcopy');
             writeln('b : backfaceremoval');
             write('c : changed mode, the joysticks  ',
                          'changes the orientation');
             write('instead of the movement ',
                       'in the cartesian coordinate');
             writeln('system or vice versa');
             writeln('f : perform the hidden-surface ',
                                             'algorithm');
             writeln('h : the viewer moves horisontally');
             writeln('i : change the colour of the model');
             writeln('j : change the colour of the edges  ',
                                          'of the model');
             writeln('k : number of snapshots');
             writeln('l : gives the list of commands');
             writeln('m : the robot is controlled  ',
                                       'by the joysticks');
             write('o : the viewer rotates around the axis  ',
                                       'pointed at the ');
```

```
    writeln('model, number=degrees');
    writeln('p : panning motion of the viewer, ',
                                    'number=degrees');
    writeln('r : the viewer rolls  ',
                  '(moves backwards or forwards)');
    writeln('s : shading');
    writeln('t : the viewer tilts, number=degrees');
    writeln('v : the viewer moves vertically');
    writeln('w : a point-welding program is called');
    writeln('z : the viewer zooms');
end
else
if ch='a' then
begin
    DrawForeground;
    PrintExactScreen;
    DrawBackground;
end
else
if ch='f' then
begin
    bafare:=false;
    BackFaceRemoval(bafare);
    notmoving:=true;
    Setdepth;
    CmiErase;
    Robot6(theta1,theta2,theta3,theta4,theta5,theta6);
    SwapPlane;
    notmoving:=false;
end
else
if ch='s' then
begin
    shad:=not(shad);
    Shade(shad);
end
else
if ch='b' then
begin
    bafare:=not(bafare);
    BackFaceRemoval(bafare);
end
else
if ch='m' then
begin
    notjumpout:=true;
    while notjumpout do
    begin
        pxold:=px;
        pyold:=py;
        pzold:=pz;
        nxold:=nx;
        nyold:=ny;
        nzold:=nz;
        JoySticks(joystickvalues);
```

```
        With joystickvalues do
        begin
            if def[0] then
            begin
                if model then px:=px+values[0]/3000
                         else nx:=nx+values[0]/10000;
            end;
            if def[1] then
            begin
                if model then py:=py+values[1]/3000
                         else ny:=ny+values[1]/10000;
            end;
            if def[2] then
            begin
                if model then pz:=pz+values[2]/3000
                         else nz:=nz+values[2]/10000;
            end;
            if def[3] then theta6:=theta6+
                                   values[3]/3000;
            if def[17] then if values[17]=1 then
                                   notjumpout:=false;
            if def[18] then if values[18]=1 then
                                   notjumpout:=false;
            if def[19] then if values[19]=1 then
                                   notjumpout:=false;
            if def[20] then if values[20]=1 then
                                   notjumpout:=false;
        end;
        MoveRobot(px,py,pz,nx,ny,nz);
    end;
end
else
if ch='c' then
begin
    model:=not model;
end
else
if ch='w' then
begin
    Weld;
end
else
begin
    read(number);
    if ch='i' then
    begin
        cno:=trunc(number);
        InternalColor(cno);
    end
    else
    if ch='j' then
    begin
        cno:=trunc(number);
        FrameColor(cno);
    end
```

```
                else
                if ch='k' then
                begin
                   forhelp:=trunc(number);
                end
                else
                begin
                   newnumber:=number/forhelp;
                   Znewnumber:=sqr(number);
                   for i:=1 to forhelp do
                   begin
                      case ch of
                        'p' : Pan3(newnumber);
                        't' : Tilt3(newnumber);
                        'z' : Zoom3(Znewnumber);
                        'r' : Roll3(number);
                        'h' : begin
                                 Pan3(90);
                                 Roll3(number);
                                 Pan3(-90);
                              end;
                        'v' : begin
                                 Tilt3(90);
                                 Roll3(number);
                                 Tilt3(-90);
                              end;
                        'o' : begin
                                 Tilt3(90);
                                 Pan3(newnumber);
                                 Tilt3(-90);
                              end;
                      end;
                      Clearscreen;
                      Robot6(theta1,theta2,theta3,
                             theta4,theta5,theta6);
                      SwapPlane;
                   end;
                end;
             end;
             readln;
          end;
       end;

.END
```

Appendix 2

Ur M-matrisen för robotarmens spets får man spetsens läge och orientering. I procedure Robot6 sker följande translationer och rotationer av modellkoordinatsystemet.

Translate3(-21,-21,z_0-13) Translate3(21,21,13)	Translate3(0,0,z_0)
Rotate3z(φ1)	
Translate3(-13,-13,0) Translate3(13,13,a1)	Translate3(0,0,a1)
Rotate3y(φ2)	
Translate3(-7,-7,-5) Translate3(7,7,a2+5)	Translate3(0,0,a2)
Rotate3y(φ3)	
Translate3(-5,-6,-6) Translate3(a3+5,6,6)	Translate3(a3,0,0)
Rotate3y(φ4)	
Translate3(-4,-5,-5) Translate3(a4+4,5,5)	Translate3(a4,0,0)
Rotate3z(φ5)	
Translate3(-2,-3,-3) Translate3(a5-4,3,3)	Translate3(a5-6,0,0)
Rotate3x(φ6)	
Translate3(0,-0.5,-0.5) Translate3(6,0.5,0.5)	Translate3(6,0,0)

Anm. De lokala variablerna i Robot6 kallas φ1-φ6 medan de globala kalla θ1-θ6.

Beteckningarna S1=sinθ1, C1=cosθ1, C234=cos(θ2+θ3+θ4) etc. används.
a1-a5 är avstånd mellan vridningsaxlar enl. fig 11.
z_0 är den första vridningspunktens z-koordinat. I och med att origo för wcs ligger på den andra axeln, blir z_0=-a1.

I matrisform ger translationerna och rotationerna:

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 6 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C6 & -S6 & 0 \\ 0 & S6 & C6 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ a5-6 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} C5 & -S5 & 0 & 0 \\ S5 & C5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ a4 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C4 & 0 & S4 & 0 \\ 0 & 1 & 0 & 0 \\ -S4 & 0 & C4 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ a3 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C3 & 0 & S3 & 0 \\ 0 & 1 & 0 & 0 \\ -S3 & 0 & C3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & a2 & 1 \end{bmatrix}$$

$$\begin{bmatrix} C2 & 0 & S2 & 0 \\ 0 & 1 & 0 & 0 \\ -S2 & 0 & C2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & a1 & 1 \end{bmatrix} \begin{bmatrix} C1 & -S1 & 0 & 0 \\ S1 & C1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & z_0 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C4 & -S4 & 0 \\ 0 & S4 & C4 & 0 \\ a5 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C5 & -S5 & 0 & 0 \\ S5 & C5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ a4 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C4 & 0 & S4 & 0 \\ 0 & 1 & 0 & 0 \\ -S4 & 0 & C4 & 0 \\ a3 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} C3 & 0 & S3 & 0 \\ 0 & 1 & 0 & 0 \\ -S3 & 0 & C3 & 0 \\ 0 & 0 & a2 & 1 \end{bmatrix} \begin{bmatrix} C2 & 0 & S2 & 0 \\ 0 & 1 & 0 & 0 \\ -S2 & 0 & C2 & 0 \\ 0 & 0 & a1 & 1 \end{bmatrix} \begin{bmatrix} C1 & -S1 & 0 & 0 \\ S1 & C1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & z_0 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} C5 & -S5 & 0 & 0 \\ C6*S5 & C6*C5 & -S6 & 0 \\ S6*S5 & S6*C5 & C6 & 0 \\ a5*C5+a4 & -a5*S5 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} C4*C3-S4*S3 & 0 & C4*S3+S4*C3 & 0 \\ 0 & 1 & 0 & 0 \\ -S4*C3-C4*S3 & 0 & -S4*S3+C4*C3 & 0 \\ a3*C3 & 0 & a3*S3+a2 & 1 \end{bmatrix}$$

$$\begin{bmatrix} C2*C1 & -C2*S1 & S2 & 0 \\ S1 & C1 & 0 & 0 \\ -S2*C1 & S2*S1 & C2 & 0 \\ 0 & 0 & a1+z_0 & 1 \end{bmatrix} = \begin{bmatrix} \text{Trigonometriska} \\ \text{regler ger:} \\ C4*C3-S4*S3=C34 \\ C4*S3+S4*C3=S34 \end{bmatrix}$$

$$= \begin{bmatrix} C5 & -S5 & 0 & 0 \\ C6*S5 & C6*C5 & -S6 & 0 \\ S6*S5 & S6*C5 & C6 & 0 \\ a5*C5+a4 & -a5*S5 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} C34*C2*C1-S34*S2*C1 & -C34*C2*S1+S34*S2*S1 \\ S1 & C1 \\ -S34*C2*C1-C34*S2*C1 & S34*C2*S1+C34*S2*S1 \\ a3*C3*C2*C1- & -a3*C3*C2*S1+ \\ (a3*S3+a2)*S2*C1 & (a3*S3+a2)*S2*S1 \end{bmatrix}$$

$$\begin{bmatrix} C34*S2+S34*C2 & 0 \\ 0 & 0 \\ -S34*S2+C34*C2 & 0 \\ a3*C3*S2+ & 1 \\ (a3*S3+a2)*C2+ & \\ a1+z_0 & \end{bmatrix} =$$

$$\begin{bmatrix} C5*C234*C1-S5*S1 & -C5*C234*S1-S5*C1 \\ C6*S5*C234*C1+C6*C5*S1+S6*S234*C1 & -C6*S5*C234*S1+C6*C5*C1-S6*S234*S1 \\ S6*S5*C234*C1+S6*C5*S1-C6*S234*C1 & -S6*S5*C234*S1+S6*C5*C1+C6*S234*S1 \\ (a5*C5+a4)*C234*C1-a5*S5*S1+ & -(a5*C5+a4)*C234*S1-a5*S5*C1- \\ +a3*C23*C1-a2*S2*C1 & -a3*C23*S1+a2*S2*S1 \end{bmatrix}$$

$$\begin{bmatrix} -C5*C234*S1-S5*C1 & C5*S234 & 0 \\ -C6*S5*C234*S1+C6*C5*C1-S6*S234*S1 & C6*S5*S234-S6*C234 & 0 \\ -S6*S5*C234*S1+S6*C5*C1+C6*S234*S1 & S6*S5*S234+C6*C234 & 0 \\ -(a5*C5+a4)*C234*S1-a5*S5*C1- & (a5*C5+a4)*S234+ & 1 \\ -a3*C23*S1+a2*S2*S1 & +a3*S23+a2*C2+a1+z_0 & \end{bmatrix}$$

Denna matris skall vara lika med matrisen:

$$\begin{bmatrix} n_x & n_y & n_z & 0 \\ o_x & o_y & o_z & 0 \\ a_x & a_y & a_z & 0 \\ p_x & p_y & p_z & 1 \end{bmatrix}$$

Detta ger villkoren:

$$p_x = a5*C5*C234*C1+a4*C234*C1+a3*C23*C1-a5*S5*S1-a2*S2*C1$$

$$p_y = -a5*C5*C234*S1-a4*C234*S1-a3*C23*S1-a5*S5*C1-a2*S2*S1$$

$$p_x = a5*C5*S234+a4*S234+a3*S23+a2*C2+a1+z_0$$

$$n_x = C5*C234*C1-S5*S1$$

$$n_y = -C5*C234*S1-S5*C1$$

$$n_z = C5*S234$$

Härav framgår att vinkeln φ6 ej har någon inverkan på p och n.
För att förenkla fortsatta räkningar sättes φ6=0°.
Då kan M skrivas som produkten av fem matriser:

$$M = A_5 A_4 A_3 A_2 A_1$$

$$A_1 = \begin{bmatrix} C1 & -S1 & 0 & 0 \\ S1 & C1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & z_0 & 1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} C2 & 0 & S2 & 0 \\ 0 & 1 & 0 & 0 \\ -S2 & 0 & C2 & 0 \\ 0 & 0 & a1 & 1 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} C3 & 0 & S3 & 0 \\ 0 & 1 & 0 & 0 \\ -S3 & 0 & C3 & 0 \\ 0 & 0 & a2 & 1 \end{bmatrix}$$

$$A_4 = \begin{bmatrix} C4 & 0 & S4 & 0 \\ 0 & 1 & 0 & 0 \\ -S4 & 0 & C4 & 0 \\ a3 & 0 & 0 & 1 \end{bmatrix}$$

$$A_5 = \begin{bmatrix} C5 & -S5 & 0 & 0 \\ S5 & C5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ C5*a5+a4 & -S5*a5 & 0 & 1 \end{bmatrix}$$

Inversen av en transformationsmatris

$$T = \begin{bmatrix} n_x & n_y & n_z & 0 \\ o_x & o_y & o_z & 0 \\ a_x & a_y & a_z & 0 \\ p_x & p_y & p_z & 1 \end{bmatrix}$$

$$T^{-1} = \begin{bmatrix} n_x & o_x & a_x & -p \cdot n \\ n_y & o_y & a_y & -p \cdot o \\ n_z & o_z & a_z & -p \cdot a \\ p_x & p_y & p_z & 1 \end{bmatrix}$$

p·o betecknar skalärprodukten mellan vektorn p och o etc.

Kommentar

Filtreringstiden har valts till 15 enheter,för att få ett bättre värde på amplituden i början.Glömskefaktorn,LD, som är ett mått på hur snabbt de gamla data glömmes bort och som får ligga mellan noll och ett,har satts till 0.9.Ju mindre glömskefaktorn är,desto snabbare glömmes de gamla data bort.Samplingsperioden har valts till 0.3.Ju mindre samplingsperioden väljs,desto snabbare konvergerar parametrarna inom det bestämda tidsintervallet då skattningarna sker på grund av att flera värden har kunnat noteras,men i gengäld får man en större spridning på de skattade parametrarna,om man t.ex väljer den till 0.1,som är lika med mätbrusets samplingsperiod.Genom ett lämpligt och noggrannare val av glömskefaktorn och samplingsperioden, kan man alltså få en bra samverkan mellan dem.

På grund av mätbrusets inverkan,har skattningsalgoritmen modifierats något på följande sätt : Bara K_c-värden,som är mindre än det förra,är noterade,eftersom en mindre proportionalverkan alltid är säkrare.Se bilaga prog.10.Bara T_c-värden,som är större men inte större än 110% av det förra,är noterade,eftersom för stora perioder möjligen kan dyka upp och K_c-värdena är känsliga för T_c-värdets variationer.Se bilaga prog.9.För de fall då perioden inte är växande,får man notera och acceptera alla T_c-värdena som ligger inom ett band t.ex mellan 90% och 110% av det gamla värdet.

Det visar sig att T_c-värdet minskas med ökande brusnivå, och K_c-värdet,som är mycket beroende av T_c-värdet i skattningsalgoritmen,ökar.Det är därför viktigt att se till att T_c-värdet inte får vara för litet,så att för stor proportional- och integralverkan kan undvikas.Detta kan t.ex lösas genom att låta skattningarna fortsätta under flera perioder,så att flera värden kommer att värderas.Se avsnitt 3.5.Man kan också göra om modifikationen på T_c-värdets skattning genom att acceptera större T_c-värde varje gång. Men det får inte förbli störst inom de närmaste fem andra skattningar,om inget större T_c-värde dyker upp.Då måste det anta det näst största värdet.

Tabell 3.4 visar en jämförelse mellan de "riktiga" och skattade processparametrarna och regulatorparametrarna för process 1 utan störningar och med tre olika brusnivåer.Det visar sig att stegsvaren svänger in sig snabbare när

brusnivån ökar.Detta beror på den ökade förstärkningen samt den minskade integraltiden.

(a).

STDEV1	K_c		T_c	
	Riktig	Skattad	Riktig	Skattad
Utan	4.0	3.51	6.3	6.44
0.03	4.0	3.11	6.3	7.53
0.09	4.0	3.67	6.3	5.28
0.1	4.0	4.7	6.3	4.22

(b).

STDEV1	Regulator					
	Riktig			Skattad		
	K	T_I	T_D	K	T_I	T_D
Utan	1.0	3.34	0.82	0.88	3.41	0.84
0.03	1.0	3.34	0.82	0.78	3.99	0.98
0.09	1.0	3.34	0.82	0.92	2.8	0.69
0.1	1.0	3.34	0.82	1.18	2.24	0.55

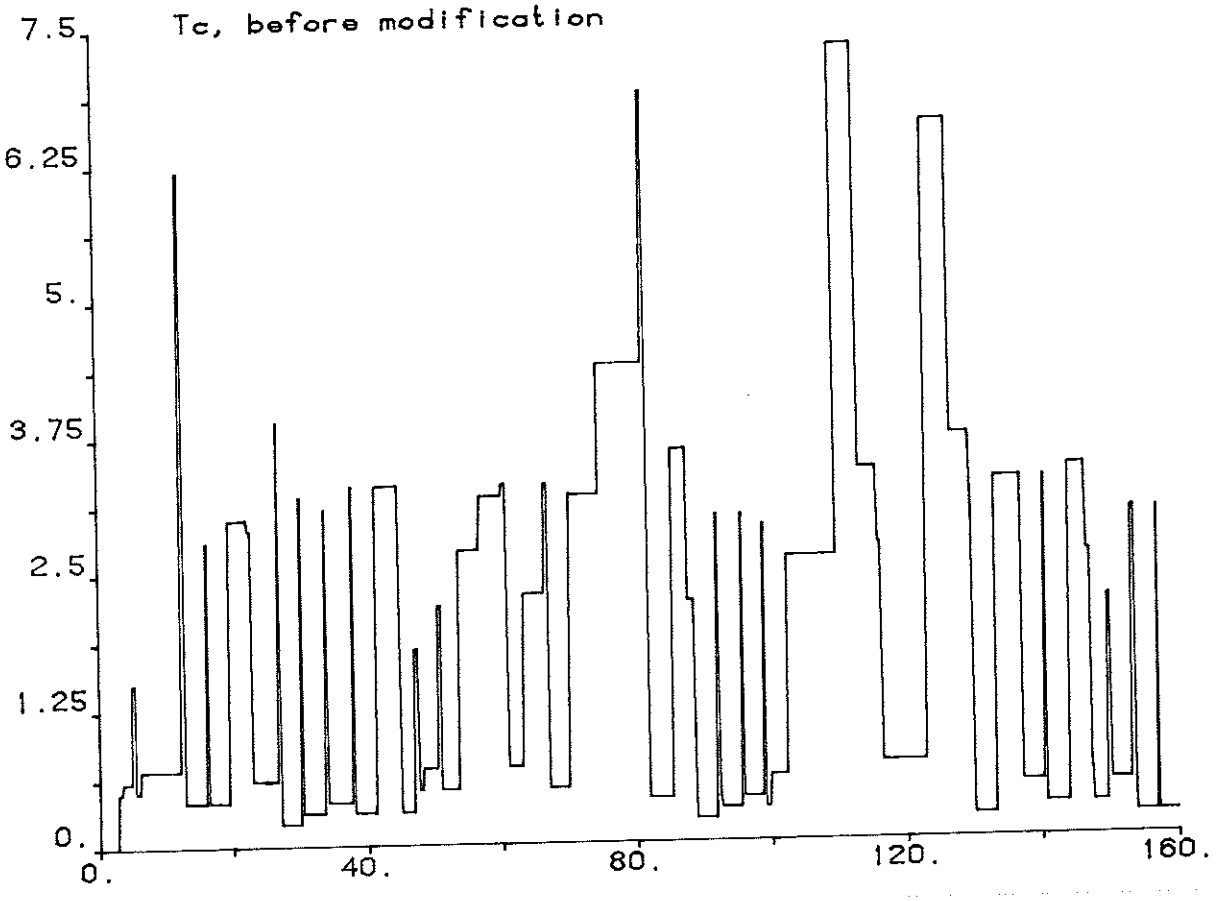
Tabell 3.4 "Riktiga" och skattade värdena på processparametrarna och regulatorparametrarna.

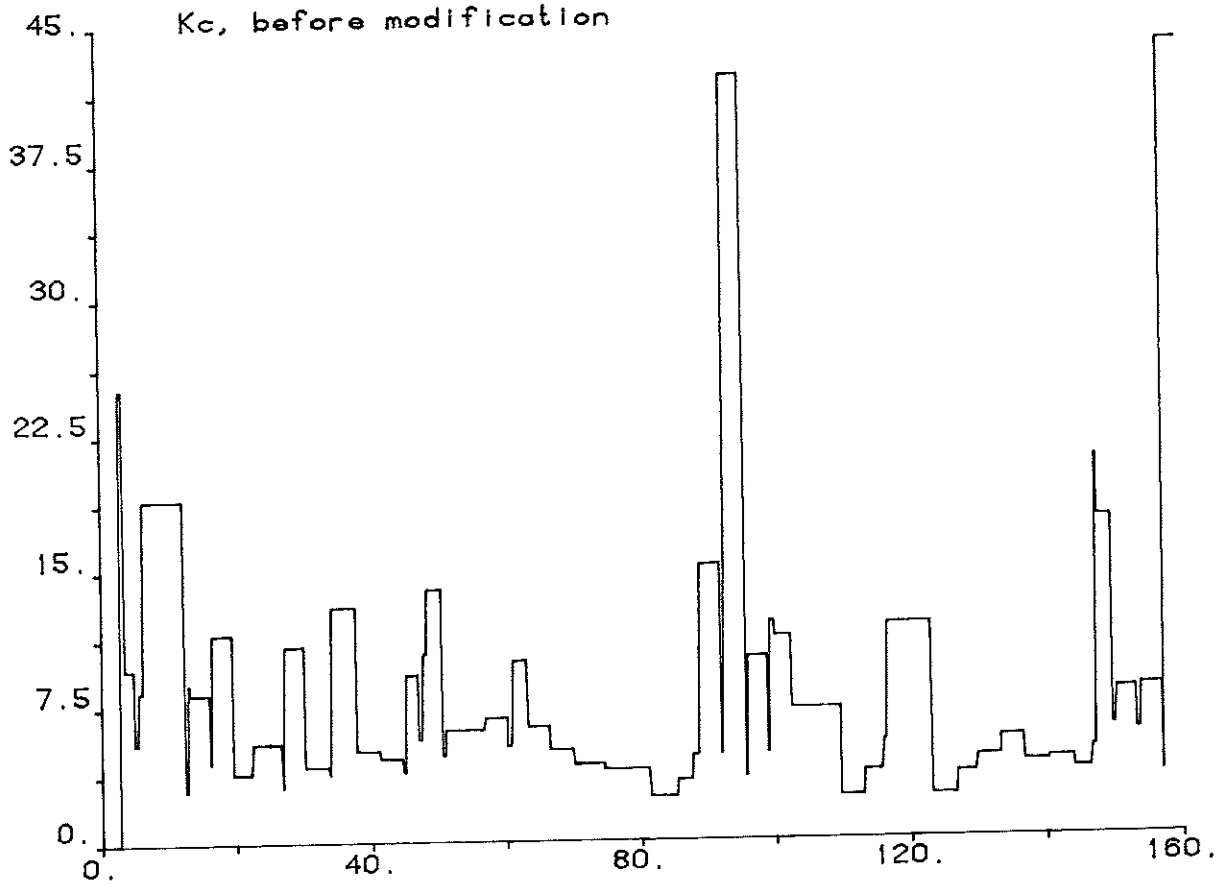
Metoden har i stort sett visat sig att fungera bra med mätbrusets inverkan efter modifikationen.

3.5 Metodernas tillförlitlighet

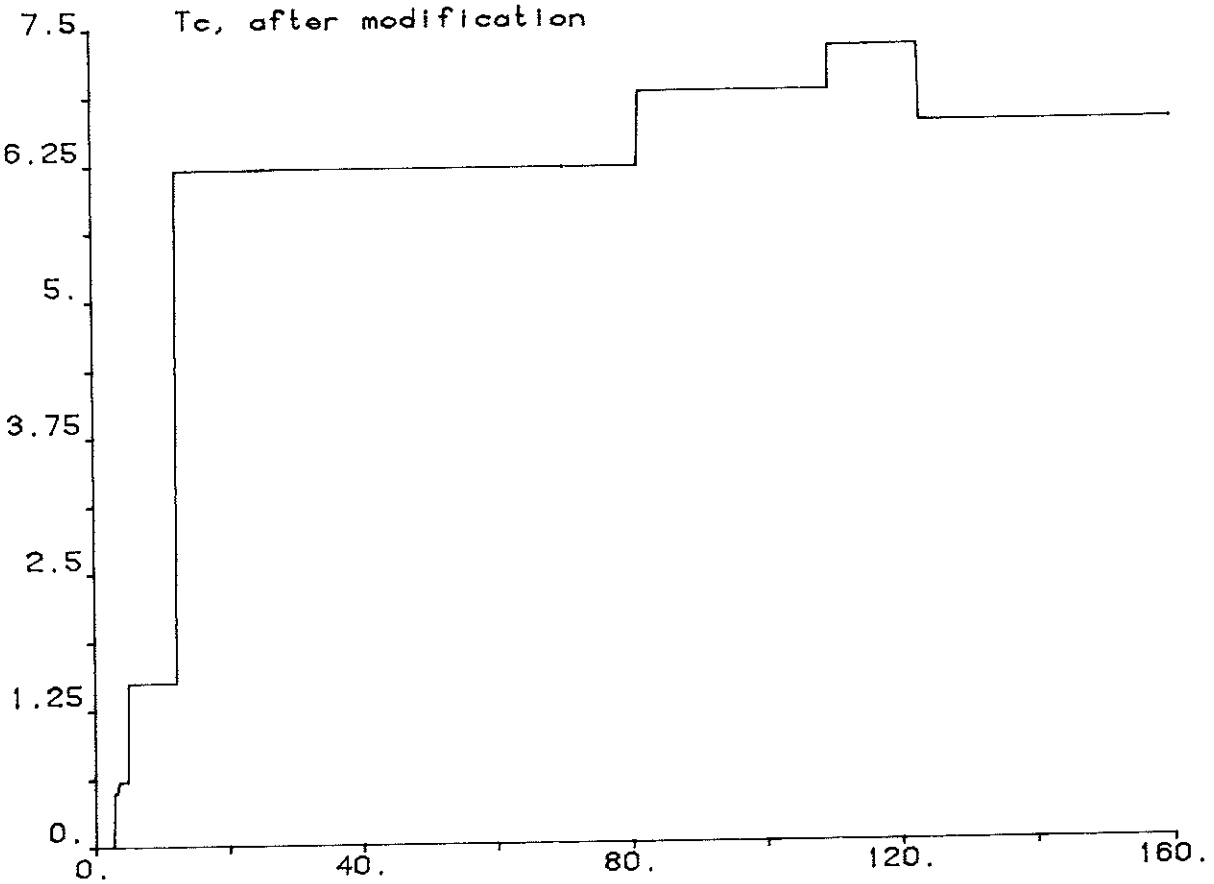
En undersökning om hur känsliga de två metoderna är för mätbruset,ska studeras i detta avsnitt genom att låta skattningarna fortsätta i en längre tid,dvs under många fler perioder.Skattningsalgoritmerna ska testas både med och utan modifikationerna före mätbrusets inverkan. Parametern,STDEV1 för mätbruset sätts här till 0.1,och hysteresbredden sätts som tidigare till 0.1.

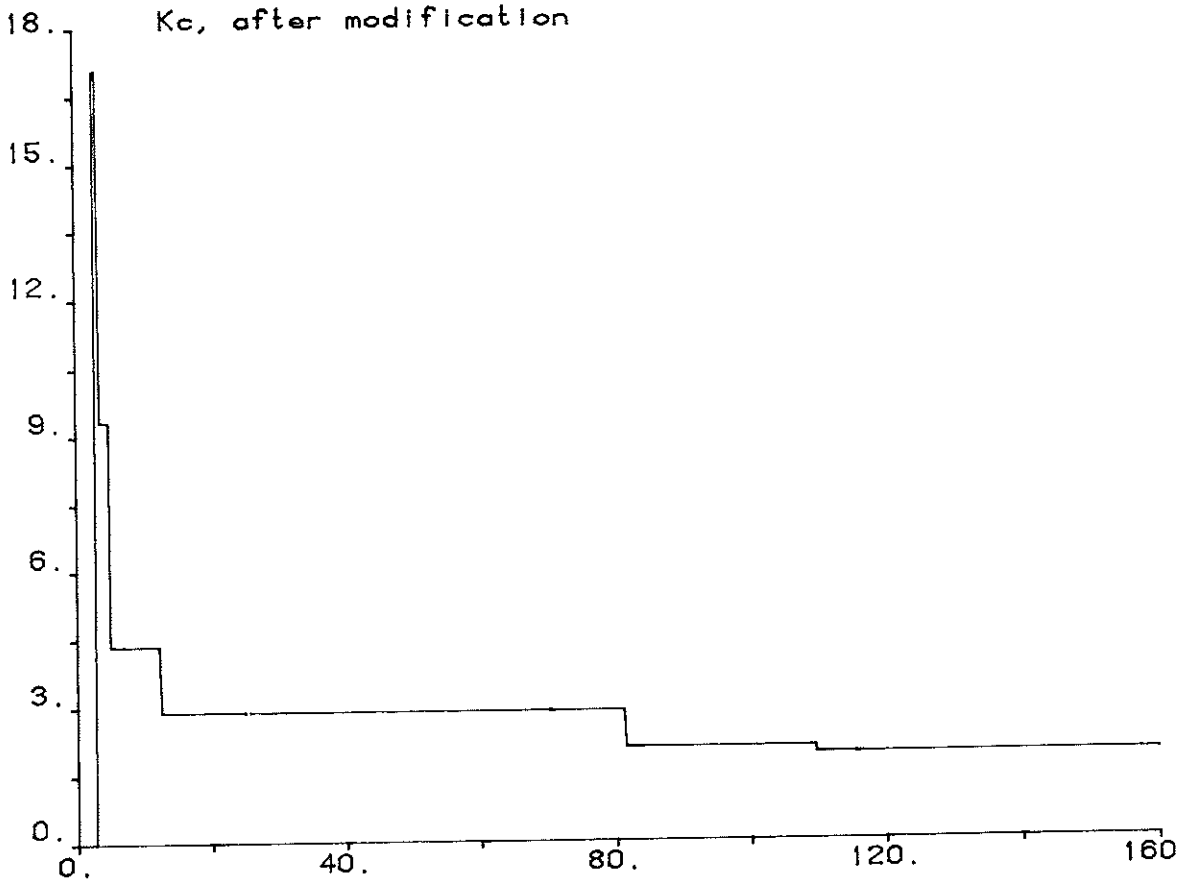
Simulering 3.5.1



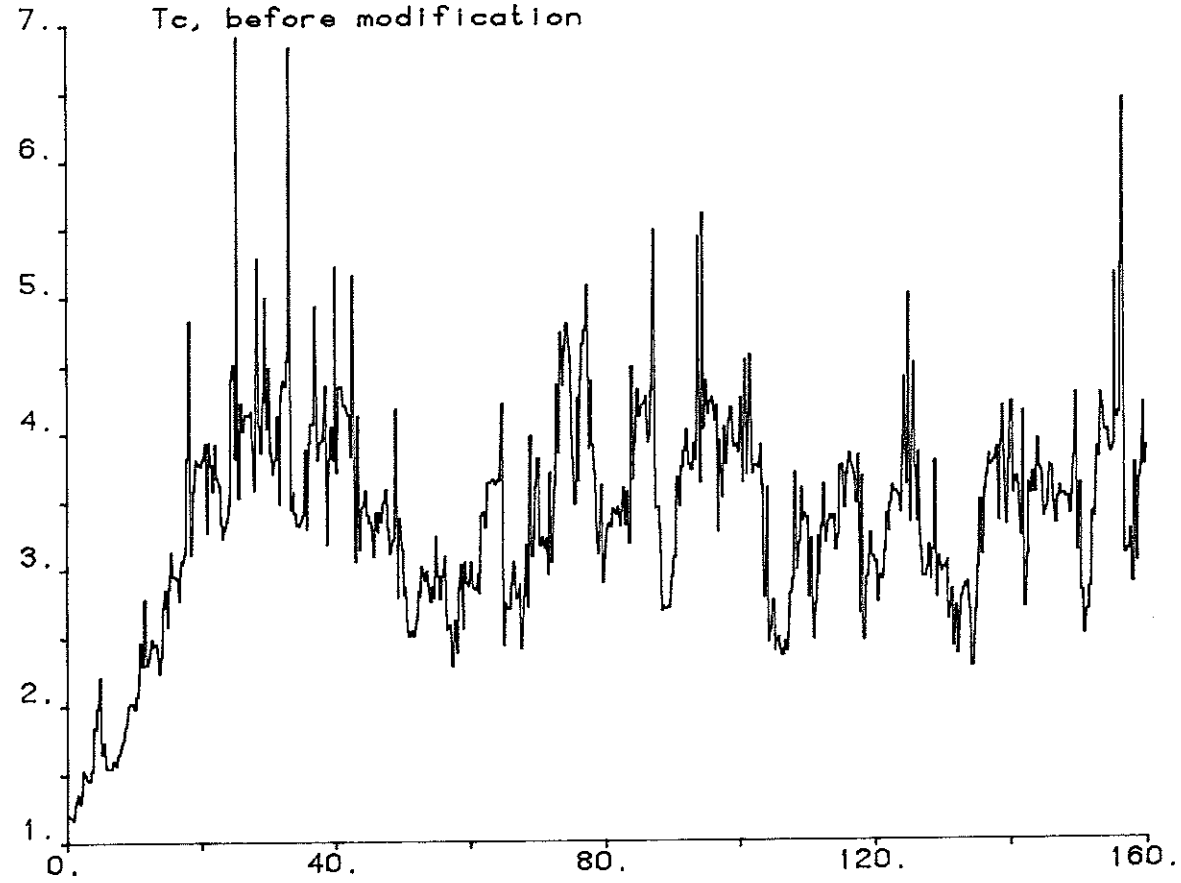


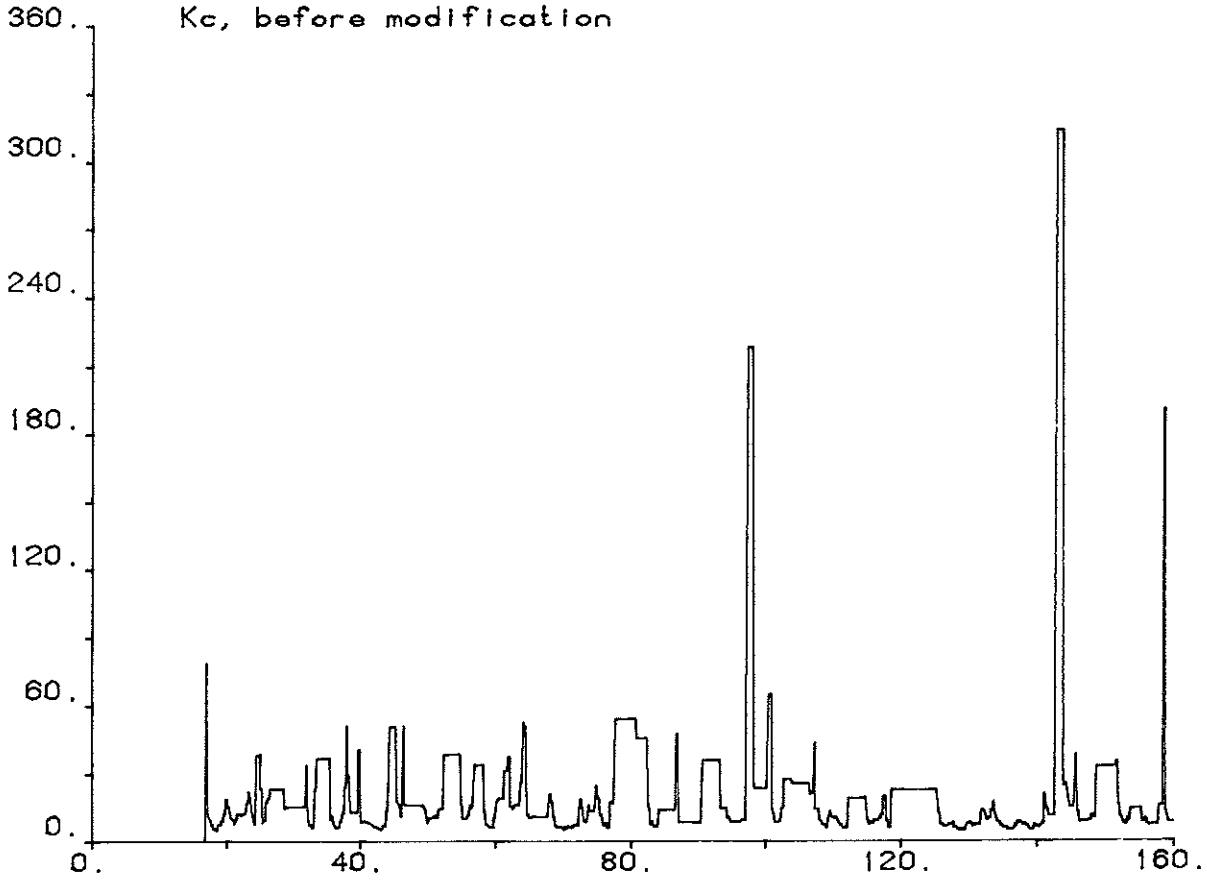
Simulering 3.5.2



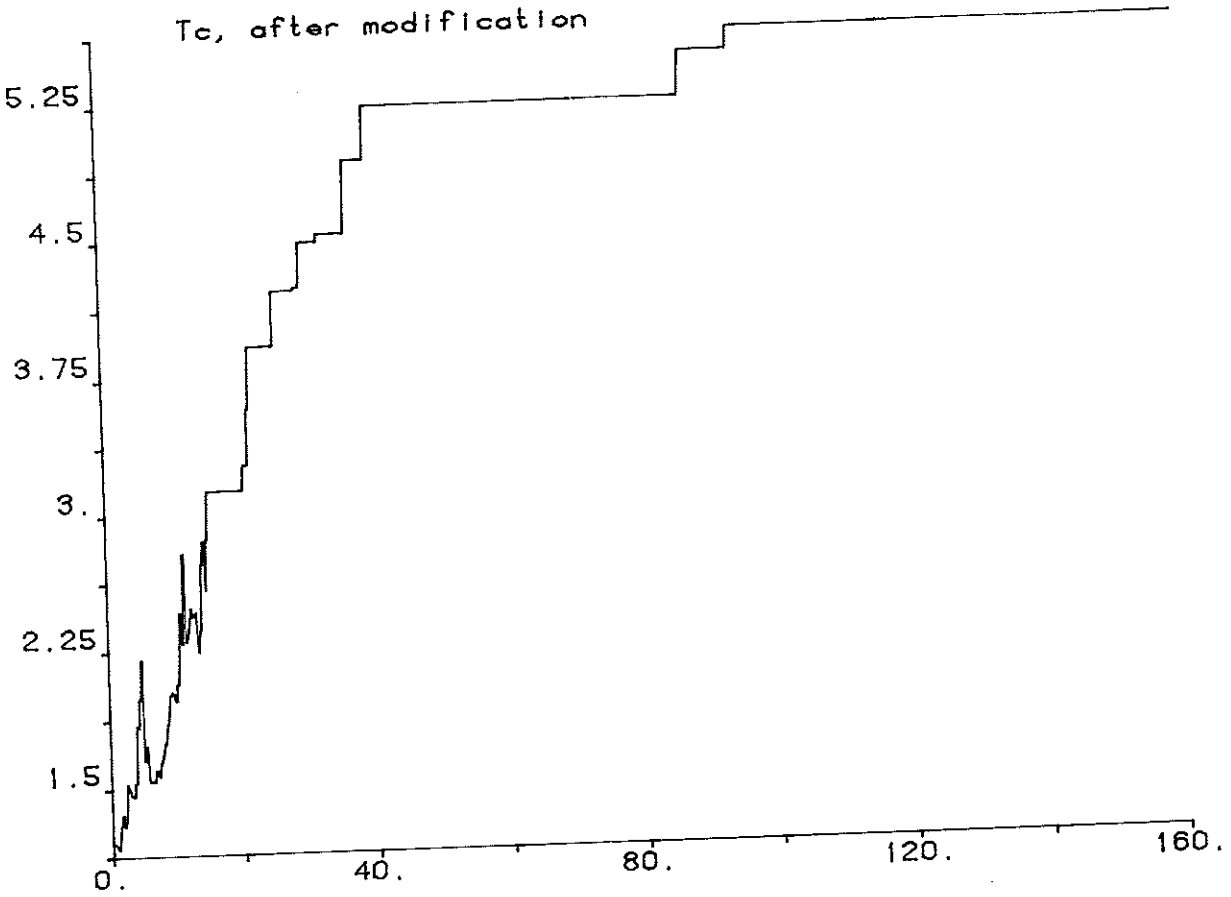


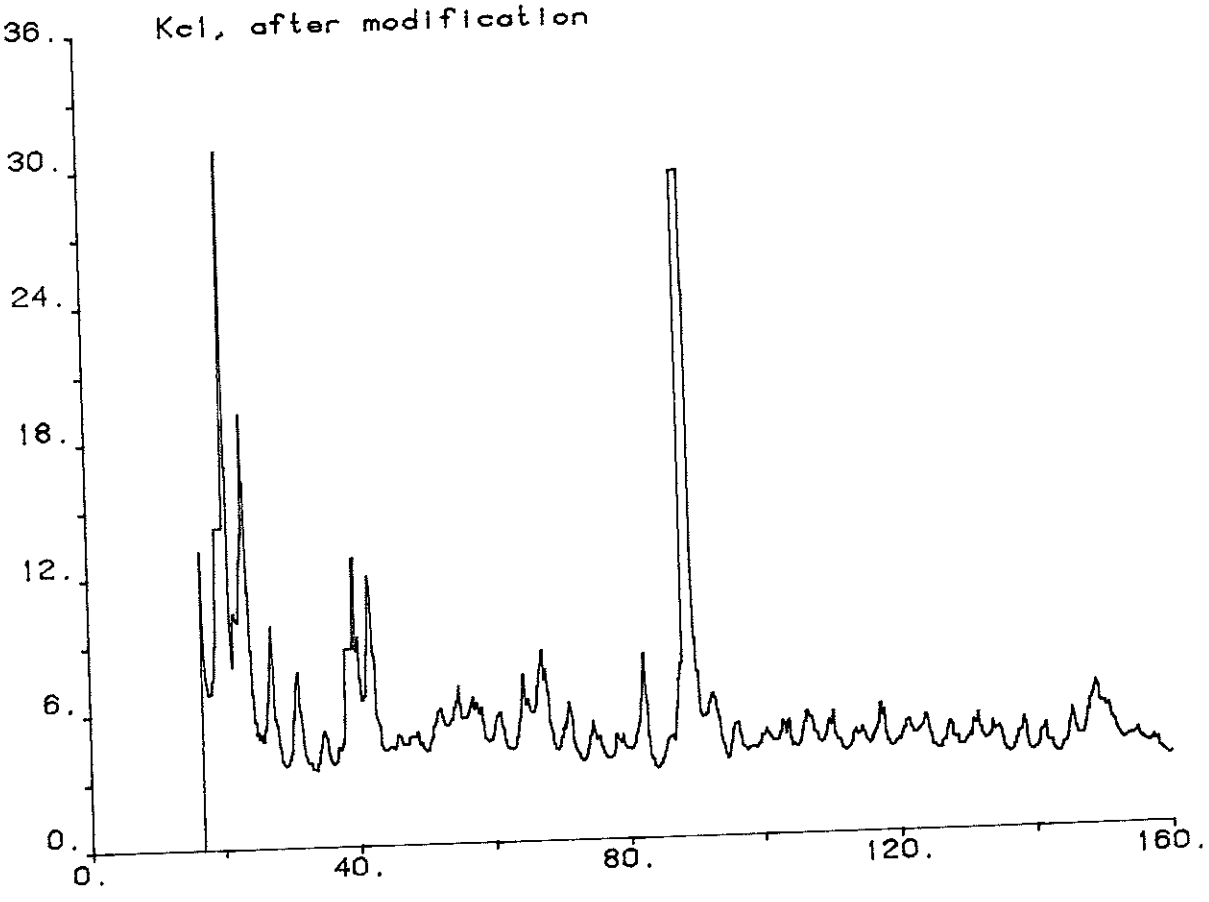
Simulering 3.5.3



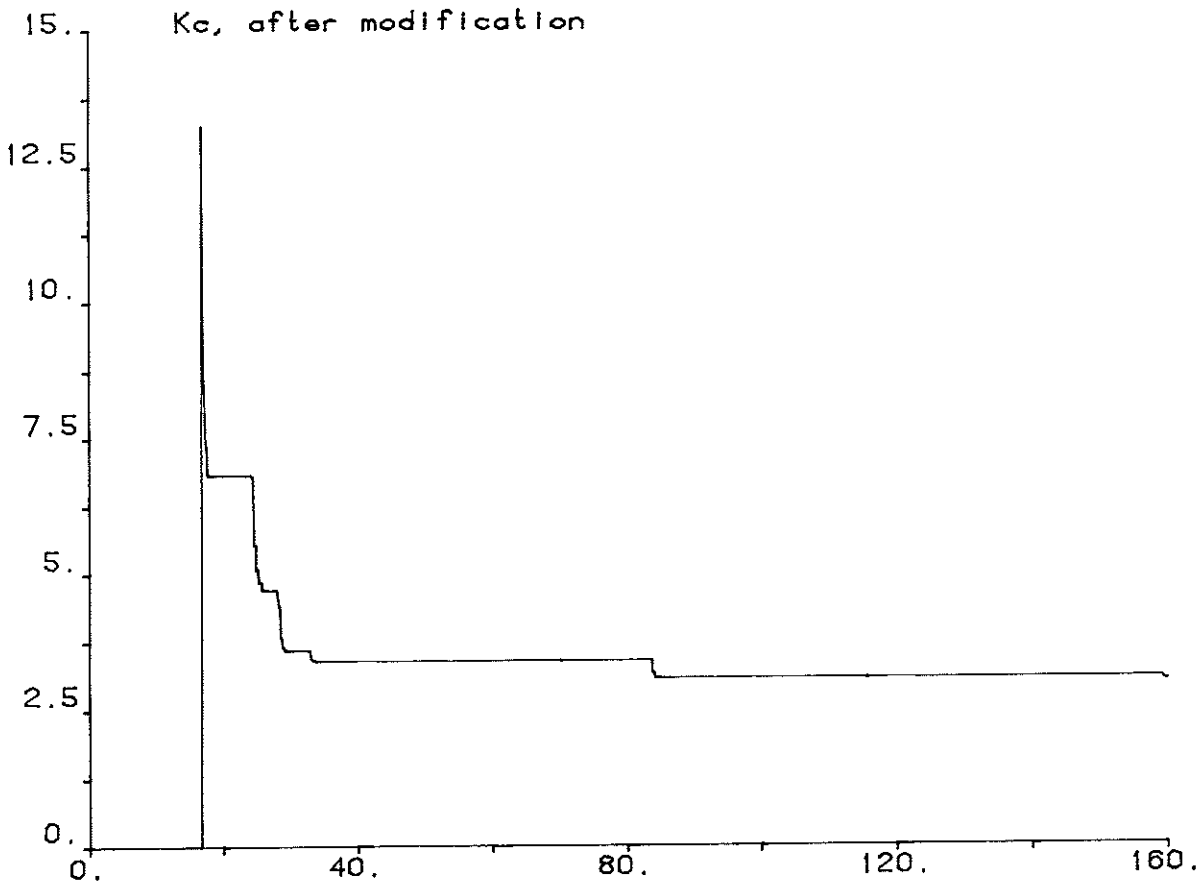


Simulering 3.5.4





Simulering 3.5.4(forts.)



Kommentar

Det visade sig att båda metoderna är känsliga för mätbruset före modifikationerna.För metoden med toppar och nollgenomgångar,är båda parametrarna oberoende av varandra. Man ser att skattningarna på perioden påverkas mycket av mätbruset,detta på grund av känsligheten vid nollgenomgångar,som leder till att räkningen av antalet nollgenomgångar blir felaktig.Många felaktiga skattningar sker vid en och samma nollgenomgång eller redan vid halva perioden.Amplituden blir också felaktigt beräknad.Se simuleringen 3.5.1.Känsligheten kan delvis minskas genom att öka samplingsperioden hos systemen till t.ex 0.3.Men det är bäst att digitalt filtrera utsignalen,s.k.lågpassfiltrering. Efter modifikationen,erhålles ett mindre K_c-värde samt ett större och riktigare T_c-värde.Se simuleringen 3.5.2.Detta innebär en säkrare inställning.Metoden har visat sig att fungera bra.

För den rekursiva minsta kvadratmetoden,är amplituden beroende av skattningen av perioden.Man ser att skattningarna av K_c-värdet varierar kraftigt då T_c-värdet hämtas direkt från skattningarna.Se simuleringen 3.5.3.Däremot är variationerna mindre då det modifierade T_c-värdet används.Se parametern,K_{c1} i simuleringen 3.5.4.Man ser också att T_c-värdena varierar omkring ett mindre värde,vilket kan bero dels på den höga brusnivå,dels på den låga glömskefaktorn,LD.Se simuleringen 3.5.3 igen. Detta kan lösas genom att LD först sätts till ett tillräckligt litet värde,t.ex 0.9 så att de oacceptable värdena i början snabbare blir bortglömda,och därefter sätts den till ett större värde,t.ex 0.99 när skattningarna börjar efter filtreringen.Metoden har visat sig att fungera bra.

Möjligheter till kombination av de två metoderna finns.Man kan,t.ex skatta amplituden med den ena och perioden med den andra.Man kan också t.ex skatta parametrarna grovt med den ena och sedan göra en finare skattning med hjälp av den andra.

3.6 Problem och svårigheter

Två praktiska problem ska behandlas här i detta avsnitt.

Den högfrekventa stabila självsvängningen

Det kan hända att Nyquistdiagrammet av processens överföringsfunktion,G skär den negativa reella axeln i G-planet fler än en gång.Så sker t.ex med process 4.Vid skärningspunkterna då funktionen argG(iω) är avtagande,är motsvarande självsvängningar stabila.Man undrar därför om metoden med beskrivande funktionen möjligen kan sätta ett sådant slutet system i oscillation i högre frekvenser. Figur 12 visar hur undersökningar har utförts med ett sådant reglersystem.Med hjälp av Bode-diagrammet,vet man vilka frekvenser de stabila skärningspunkterna ligger i.

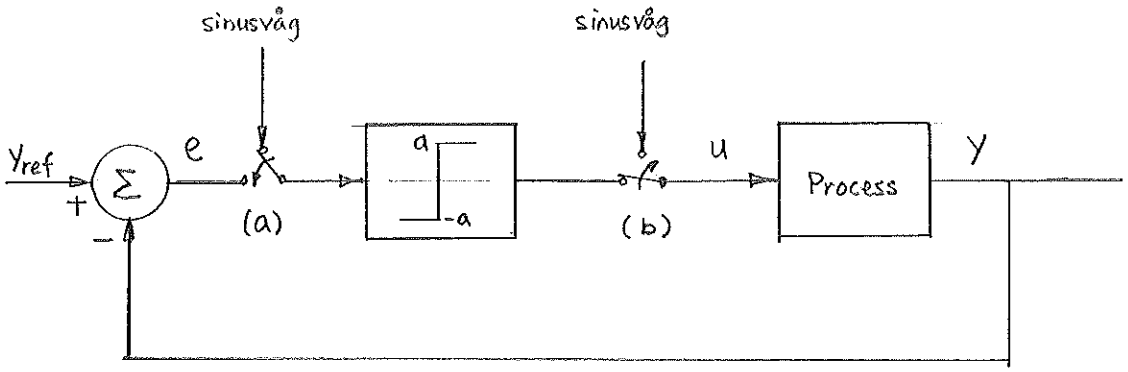


Fig.12 Blockschema för ett reglersystem.

Undersökningarna sker med ett relä och utan störningar på följande sätt : I Figur 12.a,kopplas en högfrekvenssinusvåg först till reläet så att en högfrekvensfyrkantvåg erhålls som styrsignalen till processen.Efter en tid kopplas sinusvågen bort och reglerfelet kopplas omedelbart till reläet.Två undersökningar gjordes på detta sätt,med en högfrekvensfyrkantvåg av mycket stor amplitud före bortkoppling av sinusvågen vid den ena undersökningen.Två undersökningar gjordes också enligt Figur 12.b,då en högfrekvenssinusvåg först kopplas till processen.Efter en lång tid,kopplas sinusvågen bort och utsignalen från reläet kopplas omedelbart till processen.Den ena undersökningen görs med en högfrekvenssinusvåg av mycket stor amplitud,och den andra med samma amplitud som reläet.

Vid alla fyra undersökningar har det visat sig att den högfrekventa stabila skärningspunkten har passerats,och den lågfrekventa stabila självsvängningen har uppnåtts.Som slutsats kan man säga att det verkar vara grundtonens

amplitud som dominerar.Man ser också att det finns en fördel för användandet av reläet med hysteres,eftersom de högfrekventa stabila självsvängningarna möjligen kan undvikas genom att välja bredden på reläet tillräckligt stor.

Justering av referensvärdet

Vid alla undersökningar,som hittills har utförts,har utsignalens referensvärde,Yr antagits vara lika med noll under parameterskattningarna.Avsikten med detta har varit att underlätta undersökningarna på följande sätt : Om processen har en ändlig statisk förstärkning,och Yr är skild från noll, kommer styrsignalen att ligga på en viss nivå i stationärt tillstånd.Om Yr=0,innebär det att reläet direkt kan kopplas in i det slutna systemet för parameterskattningarna utan att ta hänsyn till processens statiska förstärkning.Efter skattningarna ställs regulatorn in och därefter sätts det riktiga referensvärdet och reglering sker.Ofta är det opraktiskt att Yr ligger på nollnivån.Detta innebär att parameterskattningarna ska ske på en viss nivå.För att de tidigare beskrivna metoderna ska fungera,så måste hänsyn tagas till processens statiska förstärkning,vilken kommer att påverka styrsignalens referensnivå.Reläets position måste flyttas till den rätta nivån,så att en riktig fyrkantvåg,som styrsignalen till processen erhålls på andra sidan reläet,och utsignalen blir nästan en sinusvåg som svänger kring sitt referensvärde,Yr.Figur 13 visar den nya positionen för reläet.

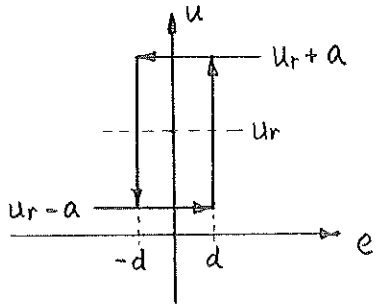
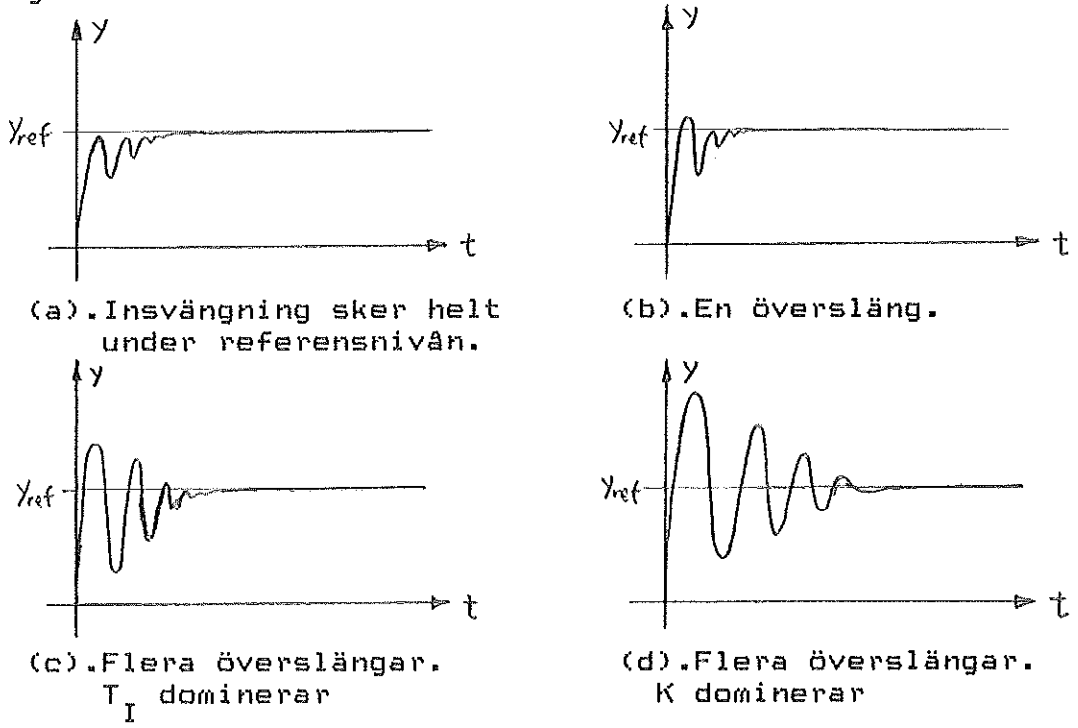


Fig.13 Karakteristik för reläet med hysteres efter nivåjusteringen

I praktiken,brukar man inte veta processens statiska förstärkning på förhand,så man måste antingen bestämma den statiska förstärkningen först eller styrsignalens referensvärde,Ur direkt.Genom att skicka ett enhetsteg till processen,kan man lätt bestämma processens statiska förstärkning ur stegsvaret,och därefter tillsammans med Yr, bestäms Ur enligt insignal-utsignal sambandet.Men en nackdel med denna metod är att utsignalen bara växer då processen,som t.ex process 4 innehåller en integrator.För

att direkt bestämma styrsignalens referensvärde,kan man t.ex använda tekniken som beskrivs i ref.(3),eller genom att använda en integrator med en lämplig integraltid,kan den nivån på vilken styrsignalen till processen ligger i stationärt tillstånd,bestämmas.Se ref.(6). En nackdel med denna metod är att utsignalen svänger in sig långsammare på grund av att skärningspunkten egentligen ligger på den negativa imaginära axeln och motsvarar en lägre frekvens.En annan nackdel är att utsignalen bara växer,om processen redan innehåller en integrator.Som slutsats kan man säga att metoden fungerar om och endast om Nyquistdiagrammet av processens överföringsfunktion skär den negativa imaginära axeln.Ett tredje sätt är att PI-verkan istället för endast I-verkan införs för att dels förbättra utsignalens insvängningstid,dels också kunna fungera för processer som innehåller en integrator.Valet av PI-parametrar faller tillbaka till problemet med inställningen,men genom en lämplig modifikation,kan metoden troligen fungera tillfredställande.

En modifikation har tänkts på följande sätt : Det är alltid säkert att starta med en liten PI-verkan,dvs liten förstärkning och stor integraltid,och låta utsignalen sakta svänga in sig under referensnivån. Se Figur 14.a.Styrsignalen börjar noteras då absolutvärdet på reglerfelet är mindre än 20% av Yr. Om detta inte är fallet,kan utsignalen uppträda i tre olika situationer.Se Figur 14.b,c och d.



(a).Insvängning sker helt under referensnivån.

(b).En översläng.

(c).Flera överslängar. T_I dominerar

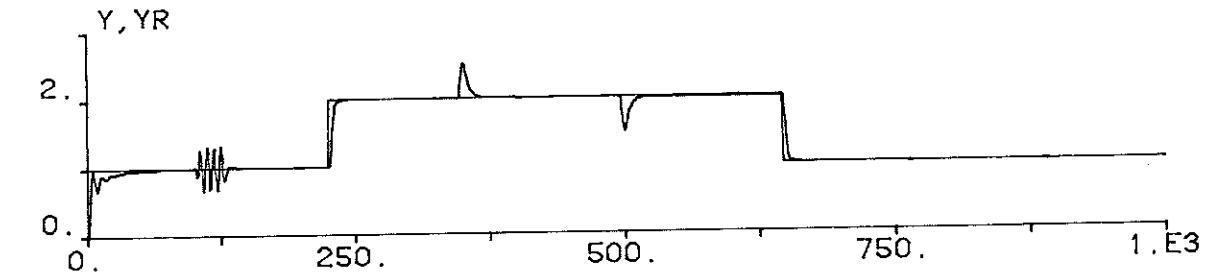
(d).Flera överslängar. K dominerar

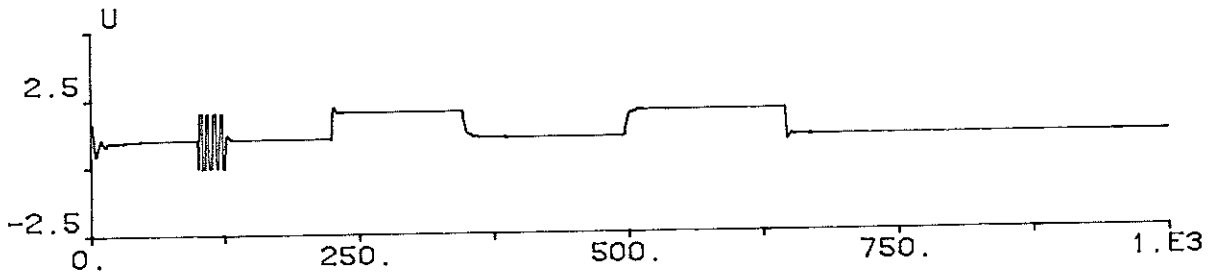
Fig.14 Olika situationer på utsignalen.

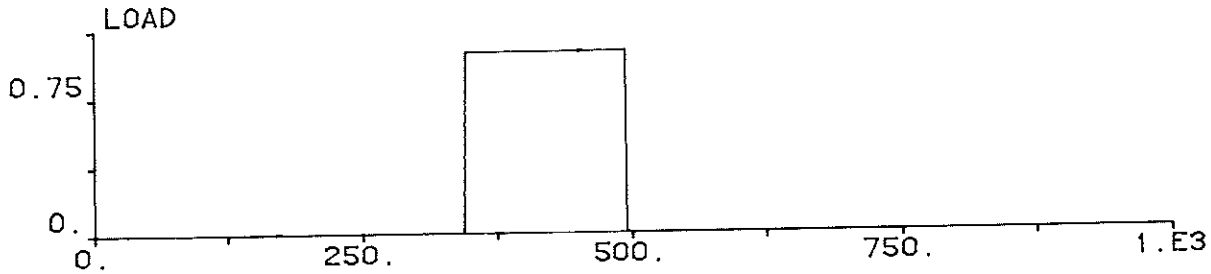
Den första är att det bara finns en översläng och de andra två är att det finns flera än en översläng.I de första två situationerna,ligger insvängningen mer eller mindre under referensnivån på grund av att det är en större integraltid,som dominerar och den andra situationen har en större förstärkning än den första.I den tredje situationen, finns det relativt större överslängar och detta är på grund av att det är en större förstärkning som dominerar. I fallet 14.b,börjar styrsignalen noteras då absolutvärdet på reglerfelet är mindre än 20% av Yr efter överslängen. Modifikationerna görs bara i de två sist nämnda situationerna och de går ut på att jämföra varje par slängar (en över- och undersläng).Principen för metoden med toppar och nollgenomgångar tillämpas också här.En stor översläng är inte tillåten,och dess tillåtna värde varierar med tillämpningen.Det räcker med att testa underslängen och modifikationerna görs då den är större än 20%,eftersom överslängen måste vara tillräckligt stor så att underslängen blir stor,eller det kan hända att utsignalen växer.Om det är den andra situationen,dvs underslängen är större än överslängen,minskas förstärkningen,K med 25% och samtidigt halveras integraltiden.Observera att minskningen av integraltiden bara sker en gång.Om det är den tredje situationen,dvs underslängen är mindre än överslängen, minskas bara K med 25%.I både fall 14.c och 14.d,noteras styrsignalen då ingen modifikation behövs,dvs utsignalen svänger in sig redan efter det första paret slängar.Om den första modifikationen behövs,noteras styrsignalen då absolutvärdet på reglerfelet är mindre än 20% av Yr efter den andra överslängen.Om fler än en modifikation behövs,noteras styrsignalen då absolutvärdet på reglerfelet är mindre än 20% av Yr efter det andra paret slängar,tills utsignalen svänger in sig.De andra modifikationerna,som är identiska för båda situationerna i fortsättningen,går ut på att minska K med 25% och öka samtidigt T_I med 50%.

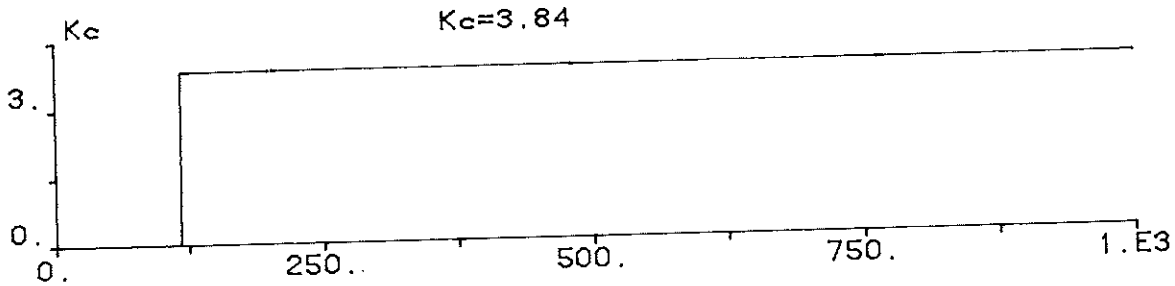
För att alla möjliga fall ska kunna övervakas i ett och samma program,kombineras alla olika noteringssätt och modifikationer. Se bilaga prog.17.Risken för alldeles för stor översläng, t.ex 50% i början finns,och man vill helst undvika den. Detta kan lösas t.ex genom att låta referensvärdet anta t.ex halva sitt riktiga värde i början före någon eller några modifikationer eller inom en viss tid.

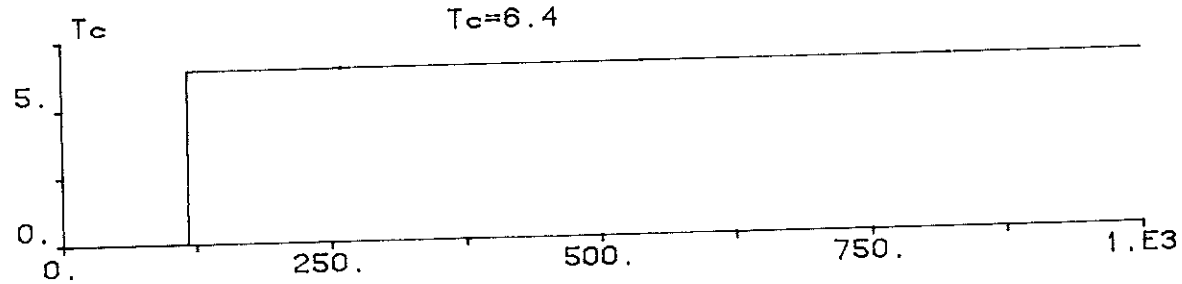
Simulering 3.6.1











4. SLUTSATSER

Det har visat sig att metoden med toppar och nollgenomgångar ger låg förstärkning och långa integral- och derivatatider vid närvaro av mätbrus.Vid användande av rekursiva minsta kvadratmetoden får man en högre förstärkning och kortare integral- och derivatatider i detta fall.Metoden med toppar och nollgenomgångar är mao robustare med avseende på mätbrus.

För tröga processer,är det bättre att filtrera bort helt oacceptabla värden i början,genom att vänta en viss tid istället för att räkna perioder då metoden med toppar och nollgenomgångar används,på grund av mätbrusets inverkan.Då rekursiva minsta kvadratmetoden används,är det bra att använda en mindre glömskefaktor under väntetiden,dvs filtreringen,och större då utsignalen har uppnått sin rätta period,så att man får bättre skattningar på T_c-värdet och

därefter bättre K_c-värdena,som är beroende på T_c-värden.

Det är bra om bredden på reläet med hysteres kan väljas stor,för att dels en säkrare inställning kan erhållas,dels den högfrekventa konvergenspunkten möjligen kan undvikas. Känsligheten vid nollgenomgångar kan minskas genom att digitalt filtrera utsignalen,den s.k.lågpassfiltreringen,så att bättre skattningar erhålls.

Det bör nämnas här att de modifikationerna,som har gjorts på de olika skattningsalgoritmerna bara är några av de många, som kan tänkas,och avsikten med dem har varit att en säkrare inställning eftersträvas.Möjligheter finns också att kombinera olika metoder så att den ena grovt skattar parametrarna och låta den andra förfina dem.

För justeringen av referensnivån på reläet,har det visat sig bra att använda PI-verkan.Valet av PI-parametrar sammanfaller dock med det ursprungliga problemet,nämligen valet av PID-parametrar.Efter modifikationerna,har det lyckats fungera tillfredställande.Insvängningstid beror mycket på initialvärdena på K_c och T_I.Det är alltid säkert

att börja med en mindre PI-verkan.Annars kan man justera utsignalens riktiga referensvärde i början för att undvika för stor översläng.

För övrigt,har det visat sig att självinställaren av PID-typ,som har beskrivits i kapitlet 2,ser ut att fungera bra.

5. REFERENSER

(1) Åström,K J (1981)
More stupid.
REPORT CODEN:LUTFD2/(TFRT-7214)/1-005/(1981).
Dept of Automatic Control,Lund Institute of Technology,
Lund,Sweden.

(2) Åström,K J (1982)
REGLERTEKNIK-en elementär introduktion.
Kapitel 5,PID reglering.
REPORT CODEN:LUTFD2/(TFRT-3166)/1-050/(1982).
Dept of Automatic Control,Lund Institute of Technology,
Lund,Sweden.

(3) Hägglund,T (1981)
A PID tuner based on phase margin specification.
REPORT CODEN:LUTFD2/(TFRT-7224)/1-020/(1981).
Dept of Automatic Control,Lund Institute of Technology,
Lund,Sweden.

(4) Åström,K J (1968)
(Eklund,K och Lindahl,S reviderat 1970)
Föreläsning i Reglerteknik FK,LTH 1968.Olinjära system.
Kapitel 5,Metoden med beskrivande funktionen.
Dept of Automatic Control,Lund Institute of Technology,
Lund,Sweden.

(5) Ogata,K
Modern control engineering.
Chapter 11,Describing function analysis of nonlinear
control systems.
University of Minnesota.

(6) Åström,K J (1982)
Ziegler-Nichols Auto-Tuner.
REPORT CODEN:LUTFD2/(TFRT-3167)/01-025/(1982).
Dept of Automatic Control,Lund Institute of Technology,
Lund,Sweden.

(7) Åström,K J and Wittenmark,B (1982)
Computer Control Theory.
Chapter 13,Modeling and identification.
Dept of Automatic Control,Lund Institute of Technology,
Lund,Sweden.

(8) Wieslander,J (1979)
(Andersson,L reviderat 1981)
REPORT User's Manual.
Dept of Automatic Control,Lund Institute of Technology,
Lund,Sweden.

BILAGOR

```
DISCRETE SYSTEM RELA
"
" * * * * RELAY * * * *
"
INPUT YR Y UR
OUTPUT U
TIME T
TSAMP TS
TS=T+H
E=YR-Y
U=IF E<0 THEN UR-A ELSE UR+A
"
"
"
H:0.1
A:1.0
END
```

```
DISCRETE SYSTEM REHYS
"
" * * * * RELAY WITH HYSTERESIS * * * *
"
INPUT Y YR UR
OUTPUT U
STATE UO
NEW NUO
TIME T
TSAMP TS
TS=T+H
E=YR-Y
U=IF E<-D THEN UR-A ELSE IF (E<D AND UO<UR) THEN UR-A ELSE UR+A
NUO=U
D=KD*A
"
"
"
H:0.1
A:1.0
KD:0.1
"
"
"
UO:-1
END
```

```
DISCRETE SYSTEM ESTT10
INPUT Y YR
OUTPUT TC
STATE MAO M TNO W LAG LO JO
NEW NMAO NM NTNO NW NLAG NLO NJO
TIME T
TSAMP TS
TS=T+H
E=YR-Y
"
" * * * * DETERMINATION OF PERIOD : ZERO CROSSINGS * * * *
"
MA=IF E)0 THEN 0 ELSE IF E(0 THEN 1 ELSE MAO
TV=IF ((MA(MAO OR MA)MAO) AND MAO(1.5) THEN 1 ELSE 0
N=IF (TV)0.5 AND LAG(0.5) THEN M+1 ELSE M
L=IF (N)M AND T)TH) THEN LO+1 ELSE LO
NLO=IF L)2.5 THEN 1 ELSE L
J=IF L)2.5 THEN JO+1 ELSE JO
NJO=J
TN=IF (L)2.5 AND J)-0.5) THEN TS-0.5*H ELSE IF L(0.5 THEN TS+0.5*H ELSE TNO
NTNO=TN
TC=IF (L)2.5 AND J)0.5) THEN TN-TNO ELSE W
NW=TC
NM=N
NMAO=MA
SET=IF J(IP THEN LAG ELSE 1
NLAG=SET
"
"
"
H:0.1
IP:2
TH:100
"
"
"
LAG:0
MAO:2
M:0
TNO:0
W:0
LO:0
JO:-2
END
```

```
DISCRETE SYSTEM ESTK10
INPUT Y YR U UR TC
OUTPUT KC TSW
STATE MAXA MINA AMPO KCO MAO M LO JO TKO TSWO GO
NEW NMAXA NMINA NAMPO NKCO NMAO NM NLO NJO NTKO NTSWO NGO
TIME T
TSAMP TS
TS=T+H
E=YR-Y
A=ABS(U-UR)
D=KD*A
"
" * * * * DETERMINATION OF AMPLITUDE : PEAK TO PEAK * * * *
"
MA=IF E>0 THEN 0 ELSE IF E<0 THEN 1 ELSE MAO
TV=IF ((MA<MAO OR MA>MAO) AND MAO<1.5) THEN 1 ELSE 0
N=IF (TV>0.5 AND GO<0.5) THEN M+1 ELSE M
L=IF (N>M AND T>TH) THEN LO+1 ELSE LO
NLO=IF L>2.5 THEN 1 ELSE L
J=IF L>2.5 THEN JO+1 ELSE JO
NJO=J
MB=IF (MA<0.5 AND L>0.5 AND J>-0.5) THEN MAX(E,MAXA) ELSE MAXA
NMAXA=IF L<2.5 THEN MB ELSE -100
MC=IF (MA>0.5 AND L>0.5 AND J>-0.5) THEN MIN(E,MINA) ELSE MINA
NMINA=IF L<2.5 THEN MC ELSE 100
AMP=IF (L>2.5 AND J>0.5) THEN (MB-MC)/2 ELSE AMPO
NAMPO=AMP
CAL=IF (L>2.5 AND J>0.5 AND AMP>D) THEN 1 ELSE 0
KC=IF CAL>0.5 THEN 4*A/(PI*SQRT(AMP*AMP-D*D)) ELSE KCO
NKCO=KC
NM=N
NMAO=MA
TK=IF J<IP THEN TS ELSE TKO
NTKO=TK
TSW=IF GO>0.5 THEN TK+0.25*TC ELSE TSWO
NTSWO=TSW
HT=IF J<IP THEN GO ELSE 1
NGO=HT
"
"
H:0.1
KD:0
PI:3.142
IP:2
TH:100
"
"
GO:0
MAXA:-100
MINA:100
AMPO:1
KCO:0
MAO:2
M:0
LO:0
JO:-2
TKO:0
TSWO:1000
END
```

```
DISCRETE SYSTEM ESTT1
INPUT Y YR
OUTPUT TC TCO
STATE MAO M TNO W WO LAG LO JO
NEW NMAO NM NTNO NW NWO NLAG NLO NJO
TIME T
TSAMP TS
TS=T+H
E=YR-Y
"
" * * * * DETERMINATION OF PERIOD : ZERO CROSSINGS * * * *
"
MA=IF E)0 THEN 0 ELSE IF E(0 THEN 1 ELSE MAO
TV=IF ((MA(MAO OR MA)MAO) AND MAO(1.5) THEN 1 ELSE 0
N=IF (TV)0.5 AND LAG(0.5) THEN M+1 ELSE M
L=IF N)M THEN LO+1 ELSE LO
NLO=IF L)2.5 THEN 1 ELSE L
J=IF L)2.5 THEN JO+1 ELSE JO
JT=IF TCO(0.9*W THEN 0 ELSE 1
NJO=IF JT(0.5 THEN JO ELSE J
TN=IF (L)2.5 AND J)-0.5) THEN TS-0.5*H ELSE IF L(0.5 THEN TS+0.5*H ELSE TNO
NTNO=TN
TCO=IF (L)2.5 AND J)0.5) THEN TN-TNO ELSE WO
NWO=TCO
TC=IF (L)2.5 AND J)0.5 AND JT)0.5) THEN TCO ELSE W
NW=TC
NM=N
NMAO=MA
SET=IF (J(IP OR JT(0.5) THEN LAG ELSE 1
NLAG=IF T(200 THEN SET ELSE 1
"
"
"
H:0.1
IP:3
"
"
"
LAG:0
MAO:2
M:0
TNO:0
W:0
WO:0
LO:0
JO:-3
END
```

```
DISCRETE SYSTEM ESTK1
INPUT Y YR U UR TC TCO
OUTPUT KC TSW
STATE MAXA MINA AMPO KCO MAO M LO JO W TKO TSWO GO TPO
NEW NMAXA NMINA NAMPO NKCO NMAO NM NLO NJO NW NTKO NTSWO NGO NTPO
TIME T
TSAMP TS
TS=T+H
E=YR-Y
A=ABS(U-UR)
D=KD*A
"
" * * * * DETERMINATION OF AMPLITUDE : PEAK TO PEAK * * * *
"
MA=IF E>0 THEN 0 ELSE IF E<0 THEN 1 ELSE MAO
TV=IF ((MA<MAO OR MA>MAO) AND MAO<1.5) THEN 1 ELSE 0
N=IF (TV>0.5 AND GO<0.5) THEN M+1 ELSE M
L=IF N>M THEN LO+1 ELSE LO
NLO=IF L>2.5 THEN 1 ELSE L
J=IF L>2.5 THEN JO+1 ELSE JO
JT=IF TCO<0.9*W THEN 0 ELSE 1
NJO=IF JT<0.5 THEN JO ELSE J
MB=IF (MA<0.5 AND L>0.5 AND J>-0.5) THEN MAX(E,MAXA) ELSE MAXA
NMAXA=MB
MC=IF (MA>0.5 AND L>0.5 AND J>-0.5) THEN MIN(E,MINA) ELSE MINA
NMINA=MC
AMP=IF (L>2.5 AND J>0.5 AND JT>0.5) THEN (MB-MC)/2 ELSE AMPO
NAMPO=AMP
CAL=IF (L>2.5 AND J>0.5 AND JT>0.5 AND AMP>D AND AMP>AMPO) THEN 1 ELSE 0
KC=IF CAL>0.5 THEN 4*A/(PI*SQRT(AMP*AMP-D*D)) ELSE KCO
NKCO=KC
NM=N
NMAO=MA
NW=TC
TK=IF GO<0.5 THEN TS ELSE TKO
NTKO=TK
TSW=IF GO>0.5 THEN TK+0.25*TP ELSE TSWO
NTSWO=TSW
TP=MAX(TC,TPO)
NTPO=TP
HT=IF (J<IP OR JT<0.5) THEN GO ELSE 1
NGO=IF T<200 THEN HT ELSE 1
"
"
"
H:0.1
KD:0.1
PI:3.142
IP:3
"
"
"
GO:0
TPO:0
MAXA:-100
MINA:100
AMPO:0
KCO:0
MAO:2
M:0
```

```
LO:0
JO:-3
W:0
TKO:0
TSWO:1000
END
```

```
DISCRETE SYSTEM ESTT20
INPUT Y YR
OUTPUT TC
STATE LAG W TKO J YO1 YO2 YO3
STATE PO11 DTO KO11
NEW NLAG NW NTKO NJ NYO1 NYO2 NYO3
NEW NPO11 NDTO NKO11
TIME T
TSAMP TS
TS=T+H
E=YR-Y
"
" * * * * DETERMINATION OF PERIOD : RECURSIVE LEAST SQUARES * * * *
"
I=J+1
NJ=I
Y1=IF LAG(0.5 THEN YO2 ELSE YO1
NYO1=Y1
Y2=IF LAG(0.5 THEN YO3 ELSE YO2
NYO2=Y2
Y3=IF LAG(0.5 THEN E ELSE YO3
NYO3=Y3
KO1=IF LAG(0.5 THEN PO11*Y2/(LD+PO11*Y2*Y2) ELSE KO11
NKO11=KO1
P11=IF LAG(0.5 THEN PO11*(1-KO1*Y2)/LD ELSE PO11
NPO11=P11
DT=IF LAG(0.5 THEN DTO+KO1*(Y3-Y2*DTO+Y1) ELSE DTO
NDTO=DT
TC=IF (DT(2 AND LAG(0.5) THEN 2*PI*H/ATAN2(SQRT(4-DT*DT),DT) ELSE W
NW=TC
TK=IF T)TH THEN TH+AN*TC ELSE TKO
NTKO=TK
SET=IF TS)TK THEN 1 ELSE LAG
NLAG=SET
"
"
H:0.3
PI:3.142
TH:15
AN:2
LD:0.9
"
"
LAG:0
W:0
TKO:1000
J:0
YO1:0
YO2:0
YO3:0
PO11:10
DTO:0
KO11:0
END
```

```
DISCRETE SYSTEM ESTK20
INPUT Y YR U TC
OUTPUT KC TSW
STATE GO TKO TSWO L SNO CSO
STATE PO11 PO12 PO22 CNO KO11 KO22 DTO1 DTO2 AMPO KCO
NEW NGO NTKO NTSWO NL NSNO NCSO
NEW NPO11 NPO12 NPO22 NCNO NKO11 NKO22 NDTO1 NDTO2 NAMPO NKCO
TIME T
TSAMP TS
TS=T+H
E=YR-Y
A=ABS(U)
D=KD*A
"
" * * * * DETERMINATION OF AMPLITUDE : RECURSIVE LEAST SQUARES * * * *
"
K=L+1
NL=K
SN=IF GO<0.5 THEN SIN(2*PI*K*H/TC) ELSE SNO
NSNO=SN
CS=IF GO<0.5 THEN COS(2*PI*K*H/TC) ELSE CSO
NCSO=CS
CN=IF GO<0.5 THEN LD+PO11*SN*SN+2*PO12*SN*CS+PO22*CS*CS ELSE CNO
NCNO=CN
KO1=IF GO<0.5 THEN (PO11*SN+PO12*CS)/CN ELSE KO11
NKO11=KO1
KO2=IF GO<0.5 THEN (PO12*SN+PO22*CS)/CN ELSE KO22
NKO22=KO2
P11=IF GO<0.5 THEN (PO11*(1-KO1*SN)-PO12*KO1*CS)/LD ELSE PO11
NPO11=P11
P12=IF GO<0.5 THEN (PO12*(1-KO1*SN)-PO22*KO1*CS)/LD ELSE PO12
NPO12=P12
P22=IF GO<0.5 THEN (-PO12*KO2*SN+PO22*(1-KO2*CS))/LD ELSE PO22
NPO22=P22
DT1=IF GO<0.5 THEN DTO1+KO1*(E-SN*DTO1-CS*DTO2) ELSE DTO1
NDTO1=DT1
DT2=IF GO<0.5 THEN DTO2+KO2*(E-SN*DTO1-CS*DTO2) ELSE DTO2
NDTO2=DT2
AMP=IF (T)TH AND GO<0.5) THEN SQRT(DT1*DT1+DT2*DT2) ELSE AMPO
NAMPO=AMP
KC=IF (T)TH AND AMP)D AND GO<0.5) THEN 4*A/(PI*SQRT(AMP*AMP-D*D)) ELSE KCO
NKCO=KC
TK=IF T)TH THEN TH+AN*TC ELSE TKO
NTKO=TK
TSW=IF GO)0.5 THEN TK+TC/4 ELSE TSWO
NTSWO=TSW
HT=IF TS)TK THEN 1 ELSE GO
NGO=HT
"
"
"
H:0.3
KD:0
PI:3.142
TH:15
AN:2
LD:0.9
"
"
"
```

```
GO:0
TKO:1000
TSWO:1000
L:0
PO11:10
PO12:0
PO22:10
CNO:0
KO11:0
KO22:0
DTO1:0
DTO2:0
AMPO:0
KCO:0
SNO:0.7
CSO:0.7
END
```

```
DISCRETE SYSTEM ESTT2
INPUT Y YR
OUTPUT TC
STATE LAG W WO TKO J YO1 YO2 YO3
STATE PO11 DTO KO11
NEW NLAG NW NWO NTKO NJ NYO1 NYO2 NYO3
NEW NPO11 NDTO NKO11
TIME T
TSAMP TS
TS=T+H
E=YR-Y
"
" * * * * DETERMINATION OF PERIOD : RECURSIVE LEAST SQUARES * * * *
"
I=J+1
NJ=I
Y1=IF LAG(0.5 THEN YO2 ELSE YO1
NYO1=Y1
Y2=IF LAG(0.5 THEN YO3 ELSE YO2
NYO2=Y2
Y3=IF LAG(0.5 THEN E ELSE YO3
NYO3=Y3
KO1=IF LAG(0.5 THEN PO11*Y2/(LD+PO11*Y2*Y2) ELSE KO11
NKO11=KO1
P11=IF LAG(0.5 THEN PO11*(1-KO1*Y2)/LD ELSE PO11
NPO11=P11
DT=IF LAG(0.5 THEN DTO+KO1*(Y3-Y2*DTO+Y1) ELSE DTO
NDTO=DT
TCO=IF (DT(2 AND LAG(0.5) THEN 2*PI*H/ATAN2(SQRT(4-DT*DT),DT) ELSE WO
NWO=TCO
TC=IF (T)TH AND (TCO(W OR TCO)1.1*W)) THEN W ELSE TCO
NW=TC
TK=IF T)TH THEN TH+AN*TC ELSE TKO
NTKO=TK
SET=IF TS)TK THEN 1 ELSE LAG
NLAG=SET
"
"
H:0.3
PI:3.142
TH:15
AN:3
LD:0.9
"
"
LAG:0
W:0
WO:0
TKO:1000
J:0
YO1:0
YO2:0
YO3:0
PO11:10
DTO:0
KO11:0
END
```

```
DISCRETE SYSTEM ESTK2
INPUT Y YR U TC
OUTPUT KC TSW
STATE GO TKO TSWO L SNO CSO
STATE PO11 PO12 PO22 CNO KO11 KO22 DTO1 DTO2 AMPO KCO1 KCO
NEW NGO NTKO NTSWO NL NSNO NCSO
NEW NPO11 NPO12 NPO22 NCNO NKO11 NKO22 NDTO1 NDTO2 NAMPO NKCO1 NKCO
TIME T
TSAMP TS
TS=T+H
E=YR-Y
A=ABS(U)
D=KD*A
"
" * * * * DETERMINATION OF AMPLITUDE : RECURSIVE LEAST SQUARES * * * *
"
K=L+1
NL=K
SN=IF GO(0.5 THEN SIN(2*PI*K*H/TC) ELSE SNO
NSNO=SN
CS=IF GO(0.5 THEN COS(2*PI*K*H/TC) ELSE CSO
NCSO=CS
CN=IF GO(0.5 THEN LD+PO11*SN*SN+2*PO12*SN*CS+PO22*CS*CS ELSE CNO
NCNO=CN
KO1=IF GO(0.5 THEN (PO11*SN+PO12*CS)/CN ELSE KO11
NKO11=KO1
KO2=IF GO(0.5 THEN (PO12*SN+PO22*CS)/CN ELSE KO22
NKO22=KO2
P11=IF GO(0.5 THEN (PO11*(1-KO1*SN)-PO12*KO1*CS)/LD ELSE PO11
NPO11=P11
P12=IF GO(0.5 THEN (PO12*(1-KO1*SN)-PO22*KO1*CS)/LD ELSE PO12
NPO12=P12
P22=IF GO(0.5 THEN (-PO12*KO2*SN+PO22*(1-KO2*CS))/LD ELSE PO22
NPO22=P22
DT1=IF GO(0.5 THEN DTO1+KO1*(E-SN*DTO1-CS*DTO2) ELSE DTO1
NDTO1=DT1
DT2=IF GO(0.5 THEN DTO2+KO2*(E-SN*DTO1-CS*DTO2) ELSE DTO2
NDTO2=DT2
AMP=IF (T)TH AND GO(0.5) THEN SQRT(DT1*DT1+DT2*DT2) ELSE AMPO
NAMPO=AMP
KC1=IF (T)TH AND AMP)D AND GO(0.5) THEN 4*A/(PI*SQRT(AMP*AMP-D*D)) ELSE KC(
NKCO1=KC1
KC=IF (T)TH AND KCO)0 AND KC1)KCO) THEN KCO ELSE KC1
NKCO=KC
TK=IF T)TH THEN TH+AN*TC ELSE TKO
NTKO=TK
TSW=IF GO)0.5 THEN TK+TC/4 ELSE TSWO
NTSWO=TSW
HT=IF TS)TK THEN 1 ELSE GO
NGO=HT
"
"
H:0.3
KD:0.1
PI:3.142
TH:15
AN:3
LD:0.9
"
"
```

```
GO:0
TKO:1000
TSWO:1000
L:0
PO11:10
PO12:0
PO22:10
CNO:0
KO11:0
KO22:0
DTO1:0
DTO2:0
AMPO:0
KCO1:0
KCO:0
SNO:0.7
CSO:0.7
END
```

```
CONTINUOUS SYSTEM PID1
"
"
"
INPUT Y YR KC TC
OUTPUT U
STATE X1 X2
DER DX1 DX2
E=YR-Y
"
"
"
K=KO*KC
TI=TIO*TC
DE=K/TI*E
DX1=IF (X1)10 AND DE)0) THEN 0 ELSE IF (X1(-10 AND DE(0) THEN 0 ELSE DE
UPI=X1+K*E
"
"
"
TD=TDO*TC
DX2=-N/TD*X2+(-Y)
UD=-N*N/TD*X2+N*(-Y)
U1=UPI+UD
U=IF U1)10 THEN 10 ELSE IF U1(-10 THEN -10 ELSE U1
"
"
KO:0.24
TIO:0.53
TDO:0.13
N:3
"
"
X1:0
X2:0
END
```

```
CONTINUOUS SYSTEM PID
"
" * * * * REGULATORN AR UPPDELAD I EN PI- OCH EN D-DEL * * * *
"
INPUT Y YR UR KC TC
OUTPUT U
STATE X1 X2
DER DX1 DX2
TIME T
E=YR-Y
"
" * * * * PI-DELEN:          * * * *
" * * * * G1(S)=K+K/(TI*S)  * * * *
" * * * * X1=K/(TI*S)*E      * * * *
"
K=KO*KC
TI=TIO*TC
DE=K/TI*E
DX1=IF (X1)10 AND DE)0) THEN 0 ELSE IF (X1(-10 AND DE(0) THEN 0 ELSE DE
UPI=UR+X1+K*E
"
" * * * * D-DELEN:                                              * * * *
" * * * * G2(S)=S*TD/(1+S*TD/N)=N-N*N/(TD*(S+N/TD))            * * * *
" * * * * TD/N:TIDKONSTANT TILL FILTER;N=3-30                  * * * *
" * * * * X2=1/(S+N/TD)*E                                      * * * *
"
TD=TDO*TC
DX2=-N/TD*X2+(UR-Y)
UD=-N*N/TD*X2+N*(UR-Y)
U1=UPI+UD
U=IF U1)10 THEN 10 ELSE IF U1(-10 THEN -10 ELSE U1
"
"
"
KO:0.24
TIO:0.53
TDO:0.13
N:3
"
"
"
X1:0
X2:0
END
```

```
CONTINUOUS SYSTEM PROC1
"
"     G(S)=K/     4
"           /(1+S)
"
INPUT U LOAD
OUTPUT Y
STATE X1 X2 X3 X4
DER DX1 DX2 DX3 DX4
V=U+LOAD
DX1=-4*X1-6*X2-4*X3-X4+V
DX2=X1
DX3=X2
DX4=X3
Y=K*X4
"
"
K:1
"
"
X1:0
X2:0
X3:0
X4:0
END
```

```
CONTINUOUS SYSTEM PROC2
"
"                                        (-S*TD)
"    PROCESS MED DELAY : G(S)=K*EXP        /
"                                        /(1+S*T)
"
INPUT U LOAD
OUTPUT Y
STATE X
DER DX
V=U+LOAD
DX=-X/T+K*V
Y=X/T
"
"
T:1
K:1
"
"
X:0
END
```

```
CONTINUOUS SYSTEM PROC3
"
"     G(S)=K*(1-S)/      3
"                  /(1+TS)
"
INPUT U LOAD
OUTPUT Y
STATE X1 X2 X3
DER DX1 DX2 DX3
V=U+LOAD
DX1=(-3*T*X1-3*X2-X3/T)/(T*T)+V
DX2=X1
DX3=X2
Y=K*(-X2+X3)/(T*T*T)
"
"
K:1
T:2
"
"
X1:0
X2:0
X3:0
END
```

```
CONTINUOUS SYSTEM PROC4
"
"                    2
"      G(S)=K*(S+5) /          2       2
"                   /(S*(S+0.5) *(S+50) )
"
INPUT U LOAD
OUTPUT Y
STATE X1 X2 X3 X4 X5
DER DX1 DX2 DX3 DX4 DX5
V=U+LOAD
DX1=-101*X1-2600.25*X2-2525*X3-625*X4+V
DX2=X1
DX3=X2
DX4=X3
DX5=X4
Y=K*(X3+10*X4+25*X5)
"
"
K:10
"
"
X1:0
X2:0
X3:0
X4:0
X5:0
END
```

```
DISCRETE SYSTEM PI
INPUT Y YR
OUTPUT U UR FD
STATE IO KO TIO LO BFO M1O M2O URO JO MO
NEW NIO NKO NTIO NLO NBFO NM1O NM2O NURO NJO NMO
TIME T
TSAMP TS
TS=T+H
E=YR-Y
EA=ABS(E)
I=IO+K*H/TI*E
U=K*E+I
V=IF U(UL THEN UL ELSE IF U)UH THEN UH ELSE U
NIO=I+V-U
BF=IF E)O THEN O ELSE IF E(O THEN 1 ELSE BFO
TV=IF ((BF(BFO OR BF)BFO) AND BFO(1.5) THEN 1 ELSE O
L=IF TV)0.5 THEN LO+1 ELSE LO
CH=IF (L)2.5 AND TV)0.5) THEN 1 ELSE O
NLO=IF (CH)0.5 AND M2)0.2*YR) THEN 1 ELSE L
NBFO=BF
M1=IF (L)0.5 AND L(1.5) THEN MAX(EA,M1O) ELSE M1O
NM1O=IF (CH)0.5 AND M2)0.2*YR) THEN -1 ELSE M1
M2=IF (L)1.5 AND L(2.5) THEN MAX(EA,M2O) ELSE M2O
NM2O=IF (CH)0.5 AND M2)0.2*YR) THEN -1 ELSE M2
K=IF (CH)0.5 AND M2)0.2*YR) THEN 0.75*KO ELSE KO
NKO=K
J=IF K(KO THEN JO+1 ELSE JO
NJO=J
TI=IF (J(1.5 AND CH)0.5 AND M2)M1 AND M2)0.2*YR) THEN 0.5*TIO ELSE TI1
TI1=IF (CH)0.5 AND M2)0.2*YR AND J)1.5) THEN 1.5*TIO ELSE TIO
NTIO=TI
M=IF (L)2.5 AND TV)0.5) THEN MO+1 ELSE MO
NMO=IF (L)2.5 AND L(3.5 AND TV)0.5) THEN O ELSE M
UR=IF (L)2.5 AND M2(0.2*YR AND T(TH) THEN U ELSE UR1
UR1=IF ((L(0.5 OR (L)1.5 AND M(0.5)) AND EA(0.2*YR AND T(TH) THEN U ELSE UR
UR2=IF (J)1.5 AND EA(0.2*YR AND T(TH) THEN U ELSE URO
NURO=UR
FD=IF T)TH THEN 1 ELSE O
"
"
H:0.1
UL:-10
UH:10
TH:100
"
"
IO:O
KO:1.5
TIO:10
LO:O
BFO:2
M1O:-1
M2O:-1
URO:O
JO:O
MO:O
END
```

```
CONNECTING SYSTEM KSMH13
TIME T
TD1[DELAY]=T-5
TEST=T-TSW[ESTK10]
YR[RELA]=0
YR[ESTT10]=0
YR[ESTK10]=0
Y[RELA]=Y[PROC2]
Y[ESTT10]=Y[PROC2]
Y[ESTK10]=Y[PROC2]
U[ESTK10]=U[RELA]
UR[RELA]=0
TC[ESTK10]=TC[ESTT10]
Y[PID1]=IF TEST>0 THEN Y[PROC2] ELSE 0
YR[PID1]=IF (TEST>75 AND TEST<225) THEN 1 ELSE 0
LOAD[PROC2]=0
U1[DELAY]=IF TEST<0 THEN U[RELA] ELSE U[PID1]
U[PROC2]=IF T<5 THEN 0 ELSE Y1[DELAY]
KC[PID1]=IF TEST>0 THEN KC[ESTK10] ELSE KC0
TC[PID1]=IF TEST>0 THEN TC[ESTT10] ELSE TC0
"
"
KC0:1.14
TC0:11.6
END
```

```
CONNECTING SYSTEM KSMHO
TIME T
TEST=T-TSW[ESTK1]
YR[REHYS]=0
YR[ESTT1]=0
YR[ESTK1]=0
YR[PID1]=IF (TEST)75 AND TEST(325) THEN 1 ELSE 0
Y[REHYS]=Y[PROC1]+E1[NOISE1]
Y[ESTT1]=Y[PROC1]+E1[NOISE1]
Y[ESTK1]=Y[PROC1]+E1[NOISE1]
Y[PID1]=IF TEST)0 THEN Y[PROC1]+E1[NOISE1] ELSE 0
U[ESTK1]=U[REHYS]
UR[REHYS]=0
UR[ESTK1]=0
TC[ESTK1]=TC[ESTT1]
TCO[ESTK1]=TCO[ESTT1]
LOAD[PROC1]=IF (TEST)145 AND TEST(235) THEN 1 ELSE 0
U[PROC1]=IF TEST(0 THEN U[REHYS] ELSE U[PID1]
KC[PID1]=IF TEST)0 THEN KC[ESTK1] ELSE 4
TC[PID1]=IF TEST)0 THEN TC[ESTT1] ELSE 6.3
END
```

```
CONNECTING SYSTEM KSMH4
TIME T
TEST=T-TSW[ESTK20]
YR[RELA]=0
YR[ESTT20]=0
YR[ESTK20]=0
Y[RELA]=Y[PROC1]
Y[ESTT20]=Y[PROC1]
Y[ESTK20]=Y[PROC1]
U[ESTK20]=U[RELA]
UR[RELA]=0
TC[ESTK20]=TC[ESTT20]
Y[PID1]=IF TEST>0 THEN Y[PROC1] ELSE 0
YR[PID1]=IF (TEST)75 AND TEST<225) THEN 1 ELSE 0
LOAD[PROC1]=0
U[PROC1]=IF TEST<0 THEN U[RELA] ELSE U[PID1]
KC[PID1]=IF TEST>0 THEN KC[ESTK20] ELSE KC0
TC[PID1]=IF TEST>0 THEN TC[ESTT20] ELSE TC0
"
"
KC0:4
TC0:6.3
END
```

```
CONNECTING SYSTEM KSMH2
TIME T
TEST=T-TSW[ESTK20]
YR[REHYS]=0
YR[ESTT20]=0
YR[ESTK20]=0
YR[PID1]=IF (TEST)75 AND TEST(325) THEN 1 ELSE 0
Y[REHYS]=Y[PROC1]+E1[NOISE1]
Y[ESTT20]=Y[PROC1]+E1[NOISE1]
Y[ESTK20]=Y[PROC1]+E1[NOISE1]
Y[PID1]=IF TEST)0 THEN Y[PROC1]+E1[NOISE1] ELSE 0
U[ESTK20]=U[REHYS]
UR[REHYS]=0
TC[ESTK20]=TC[ESTT20]
LOAD[PROC1]=IF (TEST)145 AND TEST(235) THEN 1 ELSE 0
U[PROC1]=IF TEST(0 THEN U[REHYS] ELSE U[PID1]
KC[PID1]=IF TEST)0 THEN KC[ESTK20] ELSE 4
TC[PID1]=IF TEST)0 THEN TC[ESTT20] ELSE 6.3
END
```

```
CONNECTING SYSTEM KSMH5
TIME T
TEST=T-TSW[ESTK10]
YR[PI]=YR1
YR[REHYS]=YR1
YR[ESTT10]=YR1
YR[ESTK10]=YR1
YR[PID]=IF (TEST)100 AND TEST(520) THEN YR2 ELSE YR1
Y[PI]=Y[PROC1]
Y[REHYS]=IF FD[PI](0.5 THEN YR1 ELSE Y[PROC1]
Y[ESTT10]=IF FD[PI](0.5 THEN YR1 ELSE Y[PROC1]
Y[ESTK10]=IF FD[PI](0.5 THEN YR1 ELSE Y[PROC1]
Y[PID]=IF TEST)0 THEN Y[PROC1] ELSE YR1
UR[REHYS]=IF FD[PI](0.5 THEN 0 ELSE UR[PI]
UR[ESTK10]=IF FD[PI](0.5 THEN 0 ELSE UR[PI]
UR[PID]=UR[PI]
U[ESTK10]=U[REHYS]
TC[ESTK10]=TC[ESTT10]
LOAD[PROC1]=IF (TEST)220 AND TEST(370) THEN 1 ELSE 0
U[PROC1]=IF FD[PI](0.5 THEN U[PI] ELSE U1
U1=IF TEST(0 THEN U[REHYS] ELSE U[PID]
KC[PID]=IF TEST)0 THEN KC[ESTK10] ELSE KC0
TC[PID]=IF TEST)0 THEN TC[ESTT10] ELSE TC0
"
"
YR1:1
YR2:2
KC0:4
TC0:6.3
END
```

```
MACRO KSDMH
LET N1.DELAY=0
,N2.DELAY=1
,SPACE.DELAY=4500
SYST DELAY RELA ESTT10 ESTK10 PID1 PROC2 KSMH13
STORE Y[PROC2] U[PROC2] KC[ESTK10] TC[ESTT10] YR[PID1]
STORE U1[DELAY]-ADD
PLOT Y[PROC2]
SPLIT 1 1
AXES H 0 300 V -0.5 1.6
END
```

```
MACRO KSTMH
"
" * * * * PEAK TO PEAK AND ZERO CROSSINGS * * * *
"
LET N.NOISE1=1
,NODD.NOISE1=13
SYST NOISE1 REHYS ESTT1 ESTK1 PID1 PROC1 KSMHO
PAR DT:0.1
,STDEV1:0.03
STORE U[PROC1] Y[REHYS] KC[ESTK1] TC[ESTT1] YR[PID1] LOAD[PROC1]
PLOT Y[REHYS]
SPLIT 1 1
AXES H 0 400 V -0.5 1.65
END
```

```
MACRO KSLMH
SYST RELA ESTT20 ESTK20 PID1 PROC1 KSMH4
STORE Y[PROC1] U[PROC1] KC[ESTK20] TC[ESTT20] YR[PID1]
PLOT Y[PROC1]
SPLIT 1 1
AXES H 0 400 v -0.5 1.65
END
```

```
MACRO KSKMH
"
" * * * * RECURSIVE LEAST SQUARES * * * *
"
LET N.NOISE1=1
,NODD.NOISE1=13
SYST NOISE1 REHYS ESTT20 ESTK20 PID1 PROC1 KSMH2
PAR DT:0.1
,STDEV1:0.03
STORE U[PROC1] Y[REHYS] KC[ESTK20] TC[ESTT20] YR[PID1] LOAD[PROC1]
SPLIT 1 1
PLOT Y[REHYS]
AXES H 0 500 V -0.5 1.65
END
```

```
MACRO KSRMH
"
" * * * * PEAK TO PEAK AND ZERO CROSSINGS * * * *
"
SYST PI REHYS ESTT10 ESTK10 PID PROC1 KSMH5
STORE U[PROC1] Y[PI] KC[ESTK10] TC[ESTT10] YR[PID] LOAD[PROC1]
STORE UR[PI] U[PID]-ADD
PLOT Y[PROC1]
SPLIT 1 1
AXES H 0 1000 V -0.1 2.5
END
```