ISSN 0280–5316
ISRN LUTFD2/TFRT--5463--SE

Styrning av missil med hjälp av multimodellering

Magnus Eriksson

Institutionen för Reglerteknik
Lunds Tekniska Högskola
Oktober 1992

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

Document name
MASTER THESIS

Date of issue
October 1992

Document Number
ISRN LUTFD2/TFRT--5463--SE

Author(s)
Magnus Eriksson

Supervisor
Rolf Johansson and Sven Lennart Wirkander (FOA)

Sponsoring organisation

Title and subtitle
Styrning av en missil med hjälp av multimodellering. (Controlling a missile by use of multimodelling.)

Abstract

This report investigates the possibility of simulating the cruising of a missile by using the program Matlab and SimuLink. Matlab is a mathematical package with a number of application areas, for instance statistics, signalling and control theory. SimuLink is an easy to-use program package for building models and for simulation of both linear and non-linear technical control systems.

The report describes a method of building a number of linear models by using process identification and at each point in time choosing the right model. The report also discusses how the model division should be done in order to obtain as good control as possible.

Many processes with realistic complexity cannot be described by simple linear models, and not either as SISO-(SingleInput-SingleOutput) system. But some systems can be modelled as a series of linear models. This is called multimodelling of non-linear systems.

Key words
Digital controlling, Process identification, Multimodelling.

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
Swedish

Number of pages
76

Recipient's notes

Security classification

Förord

Främst av allt vill jag tacka min handledare Sven-Lennart Wirkander på FOA 2 för hans utmärkta rådgivning och stora uppmuntran. Utan hans hjälp hade inte examensarbetet kommit så här långt.

Jag vill också tacka Åke Assarson som gick igenom modellen för missilen, och lämnade viktiga data och upplysningar.

Antonios Fokas förtjänar också ett stort tack för sin hjälp med datorsystemet och intressanta diskusioner.

Jag vill också tacka docent Rolf Johansson på Lunds Tekniska Högskola för hans synpunkter på rapporten och råd för regulatorn.

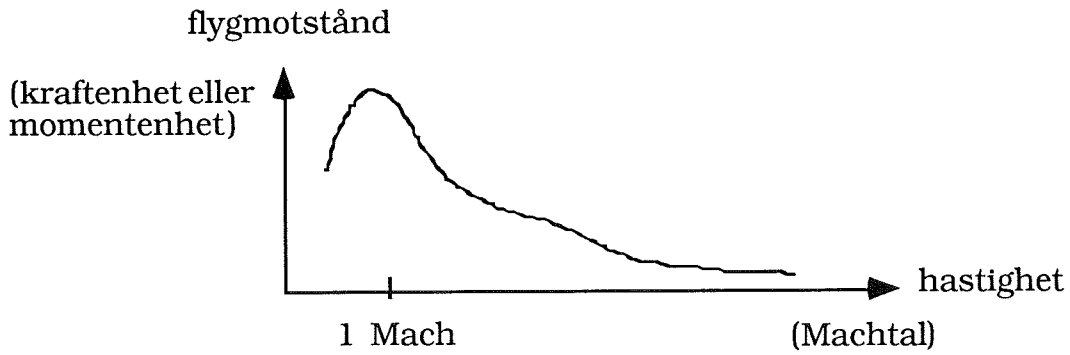
Innehållsförteckning

1. Inledning

Examensarbetet har utförts på Förvarets Forskningsanstalt (FOA 2), institutionen för styrda vapen och automation, i Stockholm. Institutionen har reglerteknik, i synnerhet för missiler, som specialitet. Provskjutningar av missiler är dyrbara och därför är man intresserad av att simulera missiler med hjälp av programvaran SimuLink. Uppgiften var också att ta fram lämpliga styrprogram för missilen.

1.1 Bakgrund

Vid konstruktionen av en missil är det en fördel om man kunde testa hur robotkroppens geometriska data, motor m m påverkar missilens prestanda utan att behöva göra många dyrbara provskjutningar. Det är alltså ganska viktigt att relativt enkelt kunna byta ut delsystem och styrprogram, och att ändra på parametrar. För detta ändamål är SimuLink tillsammans med Matlab ett bra verktyg.
Fysikaliska och tekniska processer är oftast olinjära till sin natur. Exempelvis är missilens flygmotstånd olinjärt beroende av flyghöjd och hastighet. T ex kan flygmotståndet på grund av Machtalet se ut enligt nedanstående figur:



Det finns två slags olinjära system, dels de som är olinjära i insignalen eller tillstånden, dels de som enbart är olinjära i parametrarna som t ex farten och höjden för missilen men annars är linjära i övrigt.
De metoder som man har för reglering och styrning förutsätter nästan alltid att processens dynamik kan beskrivas av linjära matematiska modeller. Ett vanligt sätt att automatiskt bygga upp modeller är rekursiv processidentifiering, som estimerar vissa systemparametrar med hjälp av en utsignal-insignal sekvens för systemet.
Denna metod kan även användas på olinjära system efter linjärisering kring en arbetspunkt. Information ska också kunna bevaras för eventuell senare återanvändning då arbetspunkten förändras. Rätt modell ska kunna väljas vid varje tidpunkt och modellindelningen ska vara "lämplig" så att styrningen blir tillräckligt bra.

1.2 Målsättning, syfte

Målsättningen är att skapa en systematisk och modulariserad uppbyggnad av simuleringsprogrammet. Institutionen vill också undersöka

* hur en regulator designad för en enda arbetspunkt, d v s given fart och höjd, fungerar över hela arbetsområdet

* hur mycket bättre en multimodellregulator som är baserad på processidentifiering on- och offline är jämfört med en regulator grundad på endast en model

* hur olika typer av process- och mätbrus i robotsystemet, mätbrus i arbetspunkten, variationer i samplingstider för styrprogrammet m m påverkar systemet

1.3 Metod

Simuleringen utföres på en Sun Sparc server med både Open-windows version 2.0 och X-windows (Unix). Modellen byggs upp som block-scheman i SimuLink medan den diskreta regulatorn skrives i programkod som m-filer (= Matlab-filer). I SimuLink finns den viktiga valfriheten att kunna välja olika integrationsmetoder, steglängder, toleranser m m.

Den digitala regulatorn skall arbeta med linjära matematiska modeller där observerare och återkoppling tillämpas. Ingen exakt kunskap om modellen får förekomma i verkligheten. Om det är möjligt bör adaptiva diskreta metoder användas, annars tillämpas multimodellering istället.

1.4 Övrigt

Examensarbetet är inte gjort genom systematisk genomgång eller teoriuppbyggnad på något avgränsat område utan efter hur framgångsrika simuleringarna för regulatorn varit.

Läsaren förutsättes vara bekant med programpaketet Matlab och ha grundläggande kunskaper i reglerteknik och numerisk analys.

2 Kortfattad beskrivning av SimuLink

2.1 Inledning

SimuLink är ett programpaket för modelluppbyggnad och analys av dynamiska system. SimuLink är en utvidgning av Matlab eftersom alla hjälpmedel i Matlab finns tillgängliga i SimuLink. Till skillnad från många andra programmeringsspråk användes grafiska symboler vid programmering i SimuLink.
Blockscheman är lättare att förstå än konventionell programkod vilket ökar förståelsen för och tillgängligheten av programmen. Programmeringen sker företrädesvis direkt i blockscheman eftersom simuleringarna går cirka 20 gånger fortare i fördefinierade program i blockscheman än i program skrivna i Matlab-språket. Vissa delar av modellen måste dock implementeras med hjälp av programkod.
Hur det går till att rita linjer, skapa block m m går jag inte in på. Mitt syfte med den kortfattade beskrivningen är innehållet i SimuLink, programkoden för dynamiska system (S-funktioner) och hur integrationen, d v s simuleringen, går till.

2.2 Blockbibliotek

Vid uppstart av SimuLink kommer ett fönster med blockbiblioteket fram. Blockbiblioteket innehåller en stor mängd fördefinierade block vilka är indelade i olika grupper:

* Signalkällor
* Signalmottagare
* Tidsdiskreta block
* Linjära block
* Olinjära block
* Anslutningar

Vidare finns en rullgardinmeny för bl a filhantering, editering, speciella funktioner, simulering och utformning.

2.3 Block

Originalblocken är enkla att hantera, flytta och modifiera för användning i egna modeller. För att ändra parametrarna i blocket dubbelklickar man på blocket och skriver in numeriska värden eller variabler vilkas värden hämtats från Matlabs arbetsminne. De olika grupperna innehåller följande block:

Signalkällor

Stegfunktion	Klocksignal	Från fil
Sinussignal	Konstant	Signalgenerator
Vitt brus	Från arbetsminne	

Signalmottagare

Oscilloskop	Till arbetsminne	Till fil

Tidsdiskreta block

Enhetsfördröjning	Filter	Tillståndsform
Pol-nollställe-placering	Överföringsfunktion	

Linjära block

Summator	Derivering	Tillståndsform
Förstärkare	Överföringsfunktion	
Integrator	Pol-nollställe-placering	

Olinjära block

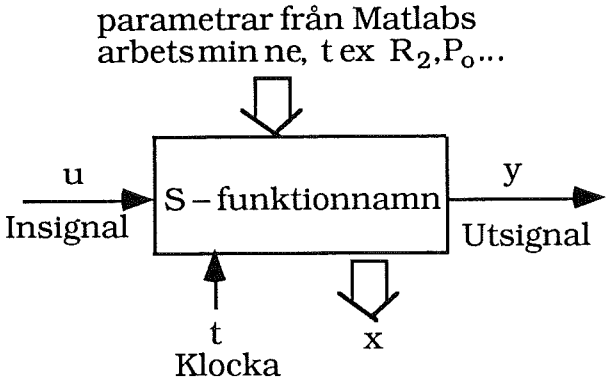
Absolutbelopp	Dödzon	Begränsare
Multiplikator	Lutningsbegränsare	Tidsfördröjning
Egen funktion	Insignalgraf	Matlab-funktion
Hysteres	Omkopplare	S-funktion

Anslutningar

Ingång	Multiplexer
Utgång	Demultiplexer

2.4 Programkod för S-funktioner

Syftet med S-funktioner (= SimuLink-funktioner) är att vanlig programkod för Matlab skall kunna skrivas också för dynamiska system. Därigenom kan gamla värden sparas i en tillståndsvariabel x för att sedan plockas fram vid nästa tidspunkt, d v s den har minne. Med S-funktioner kan man starta från vilket initialtillstånd x_0 som helst. Denna finess finns inte för grafiska SimuLinksblock. Tiden t styrs direkt från klockan. Insignalerna specificeras i u och utsignalerna i y. S-funktionblocket gör att S-funktioner blir tillgängliga som block enligt figuren nedan.



sys=S-funktionnamn(t,x,u,flag,inparametrar)

returnerar, beroende på *flag*, följande systemparametrar i *sys* :

flag	sys	Beskrivning
0	sizes	antal tillstånd, insignaler m m samt initialtillståndet x_0
1	dx	derivator dx/dt för kontinuerliga tillstånd
2	ds	nästa värde x(k+1) för diskreta tillstånd
3	y	systemutsignaler
4	tnext	nästa tidsintervall för uppdatering (bara diskreta system)

Man kan alltså blanda diskreta och kontinuerliga tillstånd. När S-funktionen initieras måste antalet in- och utgångar samt initialtillståndet x_0 anges. Denna kontroll görs genom att S-funktionen anropas med flag = 0 vilket ger följande värden på returvariabeln:

sys(1)=antal kontinuerliga tillstånd
sys(2)=antal diskreta tillstånd
sys(3)=antal utsignaler
sys(4)=antal insignaler
sys(5)=antal rötter
sys(6)=1 om systemet har direkt inmatning av insignalerna, annars = 0.
Även initialtillståndsvektorn x_0 erhålls vid detta anrop.

Om standardblock från SimuLink eller standardprogram från Matlab används behöver man inte tänka på implementeringsproblem. När egna program skrives måste däremot ordentliga tester göras separat och ändå kan det dyka upp oväntade problem. T ex måste programmet för den diskreta regulatorn skrivas så att samplingstidpunkterna inträffar när de verkligen skall och att programmet ger ifrån sig nya värden endast då samplingstidpunkterna inträffar. Det måste också klara av att hålla insignalen konstant mellan samplingarna.

2.5 Simulering i SimuLink

Som för alla integrationsmetoder måste några parametrar specificeras innan simuleringen kan börja. Nedan finns en kortfattad beskrivning av dessa parametrar:

StartTime	- tidpunkt vid vilken simuleringen skall starta
StopTime	- tidpunkt vid vilken simuleringen skall sluta
MinStepSize	- kortaste tillåtna steglängd
MaxStepSize	- längsta tillåtna steglängd
Tolerance	- gräns för relativa felet i varje integrationssteg
x_0	- vektor innehållande initialvärden för tillstånden (ej vid simulering från modellfönster)
IntegrationMethod	- val av integrationsmetod (linsim, rk45, rk23, adams, gear och euler)

Det finns två metoder att starta simuleringar, nämligen från modellfönster och från Matlabs kommandofönster. De beskrivs i det följande.

2.5.1 Simulering från modellfönster

I modellfönstrets övre del finns rullgardinsmenyer och där väljes menyn "*Simulation*". Tyvärr finns det här ingen möjlighet att specificera initialtillståndsvektorn x_0, eftersom SimuLink automatiskt sätter $x_0 = 0$. Denna simuleringsmetod är den mest användarvänliga av de två och är lämplig att använda vid enstaka simuleringar, felsökning, tidiga stadier under modellbyggnadsfasen m m.

Fördelar:
* simuleringarna kan enkelt stoppas
* insignalerna och utsignalerna består enbart av standardblock vilket inte medför några problem p g a dålig implementering

Nackdelar:
* det går inte att direkt komma åt tillstånden x
* simuleringarna kan inte automatiseras, d v s upprepas av sig själva med olika insignaler eller parametrar

2.5.2 Simulering från kommandofönster

Med denna simuleringsmetod skriver man i kommandofönstret

[t,x,y]=integrationmethod('blocknamn',[StartTime StopTime],x_0,options);

Vänsterledet är de svarsvariabler som blir tillgängliga i Matlabs arbetsminne efter simuleringens slut. Tiden *t* innehåller de tidspunkter vid vilka värdena har sparats för tillstånden och utsignalerna. Denna lagring görs radvis i matriserna *x* resp *y*. "*blocknamn*" är det SimuLink-system som anropas och *options = [Tolerance, MinStepSize, MaxStepSize]* specifierar vissa integrationsparametrar.
Det är också möjligt att skriva anropet i en m-fil och att i kommandofönstret anropa m-filen istället. Då kan data bearbetas, sparas o s v.

Fördelar: anrop kan skrivas i en m-fil
insignalen kan väljas antingen som en signalkälla i form av ett block eller som en tabellvektor u = [t, u_1, u_2,..., u_n] där det finns n st insignaler som funktion av tiden t.
x_0 kan specifieras för hela blocksystemet. Den förra metodkunde specificera endast tillstånden x_0 för varje S-funktion, inte de tillstånd som finns i de grafiska blocken

Nackdelar: vid implementeringen är det besvärligt att veta exakt vilka tillstånd som motsvarar varje element i x och x_0 för hela blocksystemet eftersom SimuLink blandar alla delblock
det är omöjligt att ändra på eventuella tidsvariabla parametrar inne i ett SimuLink-block under exekveringen på det sätt som den knapphändiga SimuLink-manualen påstår.
En i endast vissa fall möjlig motåtgärd är att göra om dessa parametrar $P_1(t)$, $P_2(t)$,... som extra insignalvektorer u=[t, P_1, P_2,...] vilka kan matas in direkt eller från Matlabs arbetsminne före exekveringen. I varje fall har detta faktum försvårat programmeringen.

På sista sidan i denna rapport finns ett enkelt program av processidentifieraren. Syftet är att underlätta förståelsen av S-funktioner och att ge ett programexempel.

2.6 Integrationsmetoder

I SimuLink finns möjligheten att välja mellan olika integrationsmetoder. Det är mycket viktigt att välja integrationsmetod och integrationsparametrar som är anpassade till den typ av modell som skall simuleras. I olyckliga fall kan små ändringar av parametrarna eller olika integrationsmetoder för samma parametrar ge helt olika resultat vilket avsevärt försvårar felsökningen. Därigenom minskar möjligheterna att få med den rätta dynamiken för systemet och att minimera simuleringstiden. Dessa problem beror på att det inte finns någon metod som kan klara av alla modelltyper. Den modell som används är ganska komplicerad p g a att tillstånden är blandade (= innehåller både kontinuerliga och diskreta tillstånd) och att den innehåller delsystem som är starkt olinjära medan andra delsystem är relativt linjära. Nedan beskrivs de olika integrationsmetoder som SimuLink tillhandahåller:

Linsim används för modeller som är relativt linjära men kan också användas i olinjära modeller som bara innehåller ett fåtal olinjära komponenter. Metoden har konstant steglängd. Denna metod och rk45 är de två vanligaste.

Runge-Kutta-metoderna **rk23** och **rk45** kan användas för de flesta problem men är bäst för mycket olinjära och/eller diskontinuerliga system. De kan användas i blandade system. Dock fungerar de inte bra för styva system, d v s system som innehåller dynamik av olika snabbhet. Steglängden är variabel.

Gear används för mjuka olinjära system, alltså system som inte innehåller singulariteter eller vars insignaler inte varierar för kraftigt. Bäst är den för styva system. Steglängden är variabel och går ej att styra.

Adams metod är anpassad för mjuka olinjära system. Metoden är ej lämplig för styva system. Steglängden är variabel och går ej att styra.

Eulers metod tar längre tid än de andra metoderna och ger sämre noggrannhet varför den bör användas enbart för verifiering av resultat. Metoden har konstant steglängd.

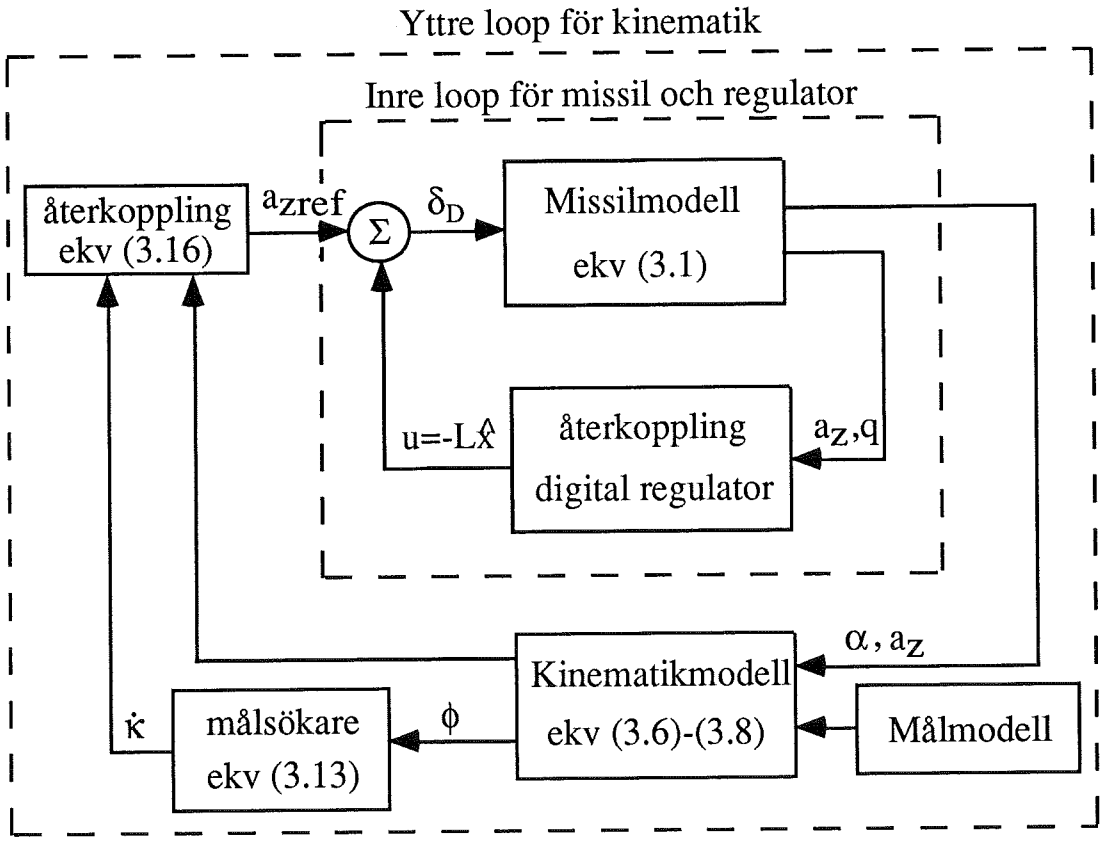
3. Beskrivning av roboten

I detta kapitel beskivs och härleds modellen för hela robotsystemet. För att förenkla den verkliga 3-dimensionella modellen begränsar vi oss till det plana fallet, i vilket missilen rör sig i ett vertikalt plan (x,z) kallat tippled. Den totala modellen består huvudsakligen av två stora block:

* det öppna systemet, d v s själva missilen, samt regulatorn vilka innehåller en inre loop med snabba tillstånd

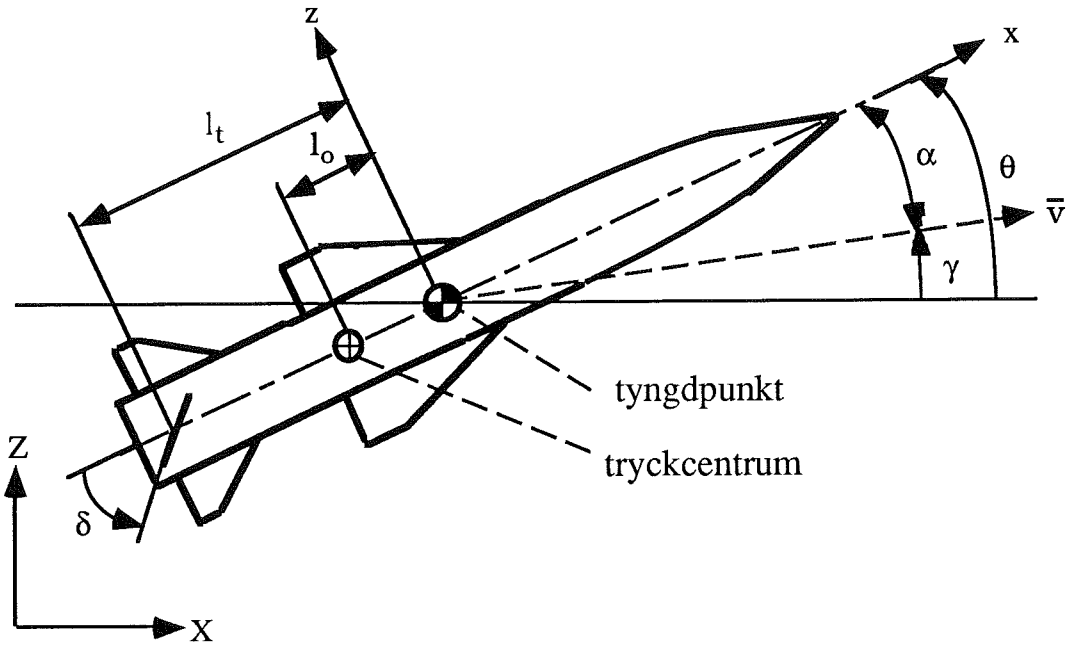
* kinematiken (samspelet mellan missilen och målobjektet) vilken innehåller en yttre loop med långsamma tillstånd

De övriga blocken är målmodell, målsökarfunktion och motor. Se översiktsskissen i figuren nedan och i blockschemat i Appendix D.



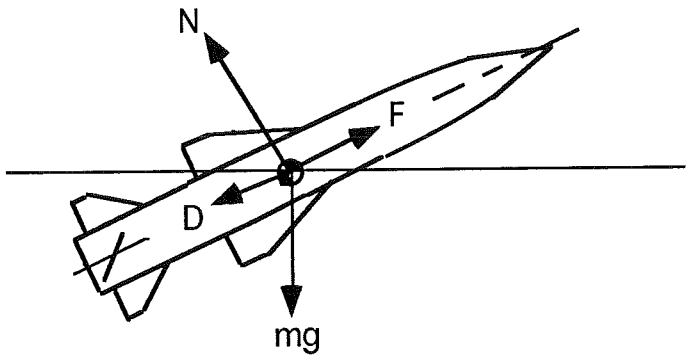
3.1 Robotmodell

För en utförlig härledning av robotens mekanik hänvisas läsaren till speciallitteratur eller robotexperter eftersom en sådan härledning kan bli mycket omfattande. Missilen skissas enligt figuren på nästa sida.



Två koordinatsystem införes, nämligen ett inertiellt koordinatsystem (X,Z) med origo i någon punkt på markytan (egentligen havsytan) och ett med missilen åtföljande koordinatsystem (x,z).
Robotkroppens längdriktning sammanfaller med x-axeln och robotens hastighetsvektor är $\bar{v}$. Roboten rör sig alltså inte i sin egen axelriktning. Vinkeln mellan hastighetsvektorn $\bar{v}$ och robotens axelriktning kallas anfallsvinkeln α.
Härigenom utövar luften ett vridmoment på roboten runt dennas tyngdpunkt. Syftet är att roboten ska få en tväracceleration vinkelrätt mot x-axeln för att kunna styra sig mot målet. För att kunna parera vridmomentet används ett roder baktill i robotkroppen.
Vinkeln mellan robotens längdriktning och rodret benämnes rodervinkeln δ. Vinkeln mellan x-axeln och den horisontella X-axeln kallas för θ. γ är vinkeln mellan hastighetsvektorns riktning och X-axeln.
När roboten under påverkan av motorkraften rör sig genom luften sammanfaller inte masscentrum och kraftcentrum. Ligger tryckcentrum bakom tyngdpunkten blir roboten stabil i luften, annars blir den instabil. Avståndet mellan dessa två centra kallas l_o. (Se figuren upptill). Avståndet mellan rodret och masscentrum benämes l_t.

Krafter: de krafter som påverkar roboten är (se figuren nedtill)



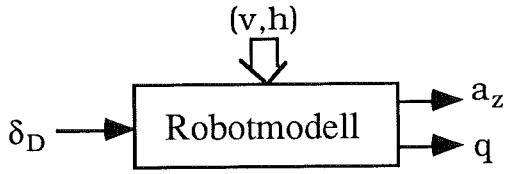
* motorns dragkraft F
* bromskraft p g a luftmotstånd D
* normalkraft N
* tyngdkraft mg

Massa: i verkligheten minskar robotens massa väsenligt under den första fasen i luften men här användes ett konstant medelvärde för massan så att robotmodellen blir tidsinvariant för annars går det inte att använda processidentifiering.

Avstånd: eftersom massan i verkligheten minskar under flygningens gång kommer robotens tyngdpunkt att förflytta sig vilket medför att robotens mekanik kommer att förändras. Vår modell av robotens mekanik förutsätter emellertid konstant massa och därmed konstanta l_o och l_t.

Målsökare: roboten har ett mål som den vill nå fram till. Målsökaren är en förutsättning för s k syftbäringsreglering vilket innebär att robothastighetens vridningshastighet skall hållas proportionell mot vridningshastigheten hos den s k syftlinjen, d v s förbindelselinjen mellan missil och mål. Målsökarens uppgift är här att uppskatta denna vridningshastighet.

Robotmodellen skulle kunna skissas av nedanstående figur:



där parametrarna (v,h) varierar olinjärt utmed hela missilbanan. Det är de som gör att en enda modell inte räcker om man önskar goda regulatorprestanda.

En robot med roderservo kan i 2 dimensioner beskrivas med följande enkla 4-tillståndsmodell:

$$(3.1)\quad \begin{cases} \dot{\alpha} = q + \dfrac{\frac{1}{2}S\rho(h)v^2}{mv}\left(C_{N\alpha}(v)\cdot\alpha + C_{N\delta}(v)\cdot\delta\right) & \text{(kraftekvation)} \\ \dot{q} = \dfrac{\frac{1}{2}S\rho(h)v^2 d}{J_0}\left(C_{m\alpha}(v)\cdot\alpha + \dfrac{d}{2v}C_{mq}(v)\cdot q + C_{m\delta}(v)\cdot\delta\right) & \text{(momentekvation)} \\ \dot{\delta} = \vartheta \\ \dot{\vartheta} = -2\zeta_0\omega_0\vartheta - \omega_0^2\delta + \omega_0^2\delta_D \end{cases}$$

Tillståndsvektorn består av

α: anfallsvinkel (vinkel mellan robotkroppens och hastighetsvektorns riktningar)

q: rotationshastighet kring tyngdpunkten

ϑ: rodervinkelhastighet

δ: rodervinkel

Missilens styrsignal är

δ_D: börvärde till roderservot

Missilen har 2 utsignaler, nämligen

$a_z = (q - \dot{\alpha})\cdot v$ (robotens acceleration vinkelrätt mot hastighetsriktningen)

q (robotens rotationshastighet)

Beträffande övriga i modellen ingående beteckningar hänvisas till appendix A. I nästa kapitel beskrives konstruktion av en styrautomat för roboten. Dess uppgift är att med hjälp av rodervinkelbörvärdet δ_D ställa in önskat värde på tvärsaccelerationen a_z.

3.2 Målmodell

Den målmodell vi använder är enkel. Målet rör sig med konstant fart v_T och konstant vinkel γ_T (se figur nedtill på nästa sida). Vi kan då skriva (x,y)-koordinaterna för målobjektet i förhållande till det inertiella koordinatsystemet som

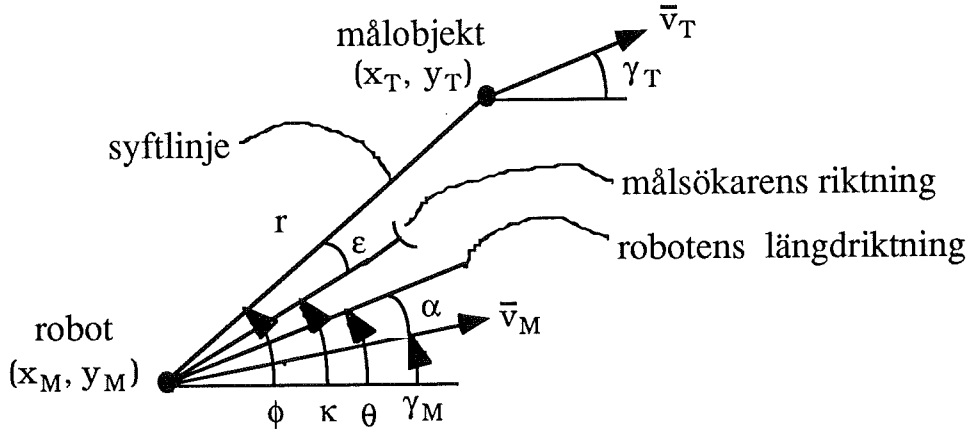
$$(3.2) \qquad \begin{aligned} x_T &= x_{To} + v_{To}\cos(\gamma_{To})\cdot t \\ y_T &= y_{To} + v_{To}\sin(\gamma_{To})\cdot t \end{aligned}$$

Givetvis är det enkelt att byta ut målmodellen ovan mot en godtycklig kontinuerlig flygbana men p g a tidsbrist begränsar vi oss till endast ett målfall, nämligen det ovannämda med följande värden

$v_{To} = 0.8$ Mach
$\gamma_{To} = 180\ °$
$x_{To} = 5$ km
$y_{To} = 5$ km

3.3 Kinematik

Kinematiken är det mekaniska samspelet mellan roboten och målobjektet. Figuren nedan underlättar förståelsen av de parametrar som beskriver kinematiken.



Vi använder här samma inertiella koordinatsystem (X,Z) som under sektion 3.1. Indexet M står för "Missil" och T för "Target". Avståndet mellan missilen och målet är r, d v s syftlinjens längd, och syftlinjens vinkel mot X-axeln är ϕ. Vinkeln för missilens hastighetvektorns riktning är γ_M. Initialt skall anfallsvinkeln α vara lika med noll. Eftersom ϕ inte kan mätas måste ett estimat av ϕ beräknas. Detta estimat kallas för κ.

När roboten skjuts iväg mot målet bör vinklarna κ och γ_M förinställas på ett optimalt sätt. Av figuren inses att ε bör vara så liten som möjligt och i starttillståndet kan ε sättas lika med noll vilket ger att $\kappa = \phi$. Eftersom α är noll i starttillståndet blir $\gamma_M = \theta$.

Enkla geometriska samband ger följande formler:

$$r = \sqrt{(x_T - x_M)^2 + (y_T - y_M)^2} \tag{3.3}$$

$$\phi = \arctan\left(\frac{y_T - y_M}{x_T - x_M}\right) \tag{3.4}$$

$$\begin{aligned} x_M &= x_T - r\cos\phi \\ y_M &= y_T - r\sin\phi \end{aligned} \tag{3.5}$$

Genom elementära mekaniska samband fås närmandehastigheten (kan mätas med dopplerradar) $v_{rel} = -\dot{r}$ och hastighetskomponenten vinkelrätt syftlinjen är $r\dot{\phi}$. De kinematiska ekvationerna blir då:

$$\dot{r} = -v_M\cos(\phi - \gamma_M) + v_T\cos(\phi - \gamma_T) \tag{3.6}$$

$$r\dot{\phi} = v_M\sin(\phi - \gamma_M) + v_T\sin(\phi - \gamma_T) \tag{3.7}$$

Det finns fyra krafter nämligen lyftkraften, motorkraften, tyngdkraften och luftmotståndet som i vårt fall bestämts semiempiriskt (se figuren på sid 17). Kraftekvationen för missilen i hastighetsvektorns riktning kan skrivas enligt Newtons lag:

$$m\dot{v}_M = F(t) - \frac{1}{2}Sv_M^2\rho(y_M)C_D(v_M,\alpha) - mg\sin(\gamma_M) \tag{3.8}$$

Motståndskoefficienten C_D kan semiempiriskt formuleras

$$C_D(v_M,\alpha) = \begin{cases} C_{Ao} + C_{N\alpha}(v_M)\alpha^2 \\ C_A + C_{N\alpha}(v_M)\alpha^2 \end{cases} \tag{3.9}$$

där det första uttrycket används då motorn är påslagen och det andra då motorn är frånslagen.

Den olinjära modellen för kinematiken blir alltså ekv (3.6)-(3.8). Vi vill konstuera en styrlag för kinematikloopen genom att tillämpa syftbärings-styrning. Utsignalen för styrlagen ska vara tvärsaccelerationen a_z. Accelerationen vinkelrätt hastighetsriktningen v_M blir:

$$a_z = v_M\dot{\gamma}_M \tag{3.10}$$

Det ideala syftbäringsvillkoret lyder:

(3.11) $\dot{\gamma}_M = c\dot{\phi}$

där c är en konstant. Ekv (3.10) och (3.11) ger då

(3.12) $a_z = cv_M\dot{\phi}$

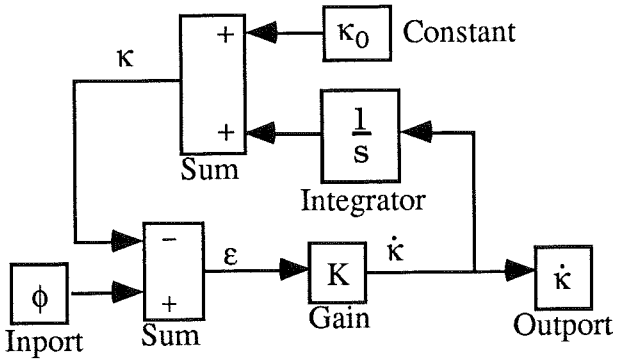
Emellertid finns endast utsignalen $\dot{\kappa}$ från målsökaren som ett estimat av $\dot{\phi}$, där κ bestäms av ϕ genom följande 1:a ordningens filter:

(3.13) $\dot{\kappa} = K(\phi - \kappa)$ där $T = \frac{1}{K}$ är tidskonstanten för målsökaren

Den allmänna lösningen till alla första ordningens filter med godtyckligt begynnelsetillstånd kan enkelt beräknas genom en vanlig integrering

(3.14) $$\kappa(t) = \kappa_0 + K\int_0^T (\phi(\tau) - \kappa(\tau))d\tau$$

där κ_0 (och ϕ_0) är initialvärden som kan enkelt beräknas m h a ekv (3.4). Felavvikelsen $\varepsilon=\kappa-\phi$ bör vara så liten som möjligt. Målsökningen blir således optimal om $\kappa_0=\phi_0$ och tidskonstanten T bör vara "lagom" liten. Målsökaren vilken formuleras av ekv (3.13) kan implementeras som ett SimuLink-block vars innehåll visas enligt skissen nedan



Genom att byta ut $\dot{\phi}$ mot $\dot{\kappa}$ fås styrlagen som

(3.15) $a_z = cv_M\dot{\kappa}$

Emellertid är v_M vansklig att bestämma i verkligheten varför ytterligare en approximation måste göras. Det gäller att

$$\alpha_k = c\frac{v_M}{v_{rel}}\cos(\phi_0 - \gamma_{M0}) \approx 3 \div 4$$

$$\Leftrightarrow \quad cv_M = \frac{\alpha_k v_{rel}}{\cos(\phi_0 - \gamma_{M0})} \approx \alpha_k v_{rel}$$

Approximationen är tillåten eftersom roboten kan ställas in så att $\gamma_{M0} \approx \phi_0$. Vår slutliga styrlag för den olinjära kinematikmodellen blir

$$(3.16) \qquad a_z = -\alpha_k \dot{r}\dot{\kappa}$$

I SimuLink-programmen i Appendix D benämnes denna a_z som a_{zref} eftersom det är denna signal som är börvärdet till den inre regulatorloopen.

4. Regulatorns uppbyggnad

Vi vill konstruera en datorimplementerad styrlag för den inre tillståndsloopen. P g a att modellen har tidsvariabla inparametrar (v,h) bör hela regulatorn automatiseras vilket innebär att den själv tar reda på modellstrukturen, observerar de icke mätbara tillstånden och beräknar själv styrlagen för den aktuella modellen.

4.1 Modell för regulatorn

En stor klass av dynamiska system kan generellt skrivas på tillståndsform enligt följande:

$$\begin{aligned} \dot{x} &= f(x,u) \\ y &= g(x) \end{aligned} \tag{4.1}$$

$x \in R^n$ är tillståndsvektorn
$y \in R^p$ är utsignalvektorn
$u \in R^r$ är insignalvektorn

I vårt fall är n=4, p=2 och r=1. Enligt kapitel 3 är robotmodellen linjär och tidsinvariant om (v,h) antar fixa värden, vilket innebär att den istället för ekv (4.1) kan skrivas

$$\begin{aligned} \dot{x} &= Ax + Bu + Gw(t) \\ y &= Cx + Du + e(t) \end{aligned} \tag{4.2}$$

där w(t) och e(t) är vita brus vilkas fördelningar antas vara normalfördelade, d v s $w(t) \in N(0,\sigma_w)$ och $e(t) \in N(0,\sigma_e)$. Brusförstärkningsmatrisen G beror av (v,h), d v s G=G(v,h) och enligt följande definition blir därmed $R_{1c}=R_{1c}(v,h)$. Kovariansen för brusen w(t) och e(t) definieras:

$$E\left[Gw(t)w^T(t+\tau)G^T\right] = R_{1c}\delta(\tau) \text{ och } E\left[e(t)e^T(t+\tau)\right] = R_{2c}\delta(\tau)$$

De vita brusen är okorrelerade vilket innebär att $E[e(t)w^T(t+\tau)]=0$. Emellertid har vi bara den samplade versionen av modellen tillgänglig.

$$\begin{aligned} x_{t+1} &= \Phi x_t + \Gamma u_t + w_t \\ y_t &= Cx_t + Du_t + e_t \end{aligned} \tag{4.3}$$

där Φ och Γ beräknas ur A och B enligt

$$\Phi = e^{Ah} \qquad h = \text{samplingstidslängden (benämnes också } t_s\text{)}$$

$$\Gamma = \int_0^h e^{Ah} ds \cdot B \tag{4.4}$$

Derivering av ekv (4.4) och därefter integrering av både Φ och Γ i matrisform ger en mera kompakt form av ekv (4.4) enligt följande:

$$(4.5) \qquad \begin{pmatrix} \Phi & \Gamma \\ 0 & I \end{pmatrix} = \exp\left\{ \begin{pmatrix} A & B \\ 0 & 0 \end{pmatrix} h \right\}$$

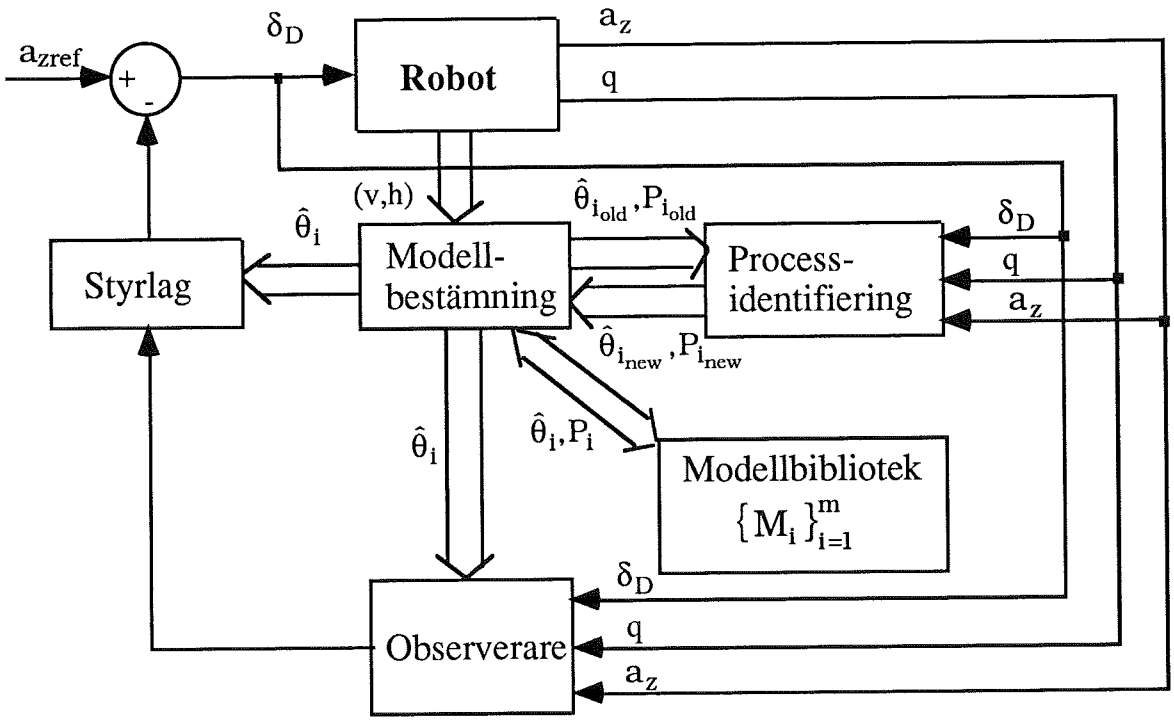
Ekv (4.5) kan alltså användas istället för ekv (4.4) om man vill integrera över en exponentialmatris vilket underlättar programmeringen avsevärt.

De vita brusen samplas också varvid kovariansmatriserna blir

$$E\left[Gw_t w_t^T G^T\right] = R_{1d} \text{ och } E\left[e_t e_t^T\right] = R_{2d}$$

där R_{1d} kan vara teoretiskt exakt eller modelleras som νI där I är enhetsmatrisen och ν ett godtyckligt reellt positiv tal. I appendix B beskrives bruset. Dock måste vara R_{1d} positivt semidefinit och R_{2d} positivt definit.

Principskissen (med vissa modifikationer för processidentifieraren) för regulatorn blir



Inkluderas processidentifieraren i regulatorn så kallas den för indirekt självinställande adaptiv regulator (indirect selftuning adaptive regulator). Mera om detta beskrives i referens [6] och i sektion 5.3. Utelämnas processidentifieraren så kallas metodiken för multimodellering av ett olinjärt system.

4.2 Observerare

Eftersom tillstånden i modellen inte kan mätas måste de estimeras vilket innebär att regulatorn behöver en bra observerare. Om modellen inte är korrekt så ger observeraren felaktiga estimat av x.
Eftersom det kontinuerliga systemet är dåligt dämpat med polerna nära imaginära axeln och förorenas med mycket process- och mätbrus bör därför en optimal observerare konstrueras, alltså ett kalmanfilter. Men enligt simuleringarna kan också polplaceringsobserveraren göra bra estimat genom lämpliga val av observerarpolerna.

En generell observerare ger uppdateringen av tillståndsestimaten

$$\hat{x}_{t+1} = \Phi\hat{x}_t + \Gamma u_t + K_t[y_t - C\hat{x}_t] \tag{4.6}$$

För polplaceringsobserveraren är K_t konstant=K och beräknas m h a Ackermanns formel vilken lyder

$$K = P(\Phi)W_o^{-1}[0,...,0,1] \tag{4.7}$$

där $P(z)=z^n+p_1z^{n-1}+...+p_n$ är observerarens karakteristiska polynom, alltså det(zI-Φ-KC) och

$$W_o = \begin{bmatrix} C \\ C\Phi \\ \vdots \\ C\Phi^{n-1} \end{bmatrix}$$

är observerbarhetsmatrisen för systemet. Polerna [0.02, -0.01, 0, 0] ger snabba insvängningsförlopp och små överslängar för robotmodellen.

Kalmanobserveraren ger ett bättre estimat när systemet innehåller brus, därför att den minimerar estimeringsfelets varians. Algoritmen för beräkningen av K_t lyder här

$$K_t = \Phi P_t C^T\left(R_{2d} + CP_tC^T\right)^{-1} \tag{4.8}$$

$$P_{t+1} = \Phi P_t\Phi^T + R_{1d} - \Phi P_tC^T\left(R_{2d} + CP_tC^T\right)^{-1}CP_t\Phi^T \tag{4.9}$$

Alla observerare kräver att (Φ,C) är observerbar, d v s att W_o har full rang.

4.3 Styrlag för regulatorn

För att styrlagen ska kunna fungera måste systemet (Φ,Γ) vara styrbart d v s styrbarhetsmatrisen $W_s=[\Gamma\ \Phi\Gamma\ ...\ \Phi^{n-1}\Gamma]$ måste ha full rang. Här tillämpas två olika styrlagar, nämligen polplacering och LQ-reglering (Linear Quadratic control).

Den generella styrlagen har formen

$$u_t = -L_t x_t \tag{4.10}$$

där L_t är den s k återkopplingsvektorn.

Eftersom processidentifieraren direkt ger en tillståndsbeskrivning på styrbar kanonsk form, d v s med matriser på formen

$$\Phi = \begin{bmatrix} -a_1 & -a_2 & \cdots & -a_{n-1} & -a_n \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \qquad \Gamma = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{4.11}$$

$$C = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \end{bmatrix} \tag{4.12}$$

kan styrlagen enkelt implementeras med polpacering. Givet det önskade slutna systemets poler z_1, ...,z_n blir den konstanta återkopplingsvektorn

$$L = [p_1 - a_1 \quad p_2 - a_2 \ .... \ p_n - a_n] \tag{4.13}$$

där p_1,p_2,...,p_n är koefficienterna i det slutna systemets karakteristiska polynom

$$P(z) = z^n + p_1 z^{n-1} + ... + p_n = \prod_{i=1}^{n}(z - z_i) = \det(zI - \Phi + \Gamma L) \tag{4.14}$$

Om regulatorn ska reagera snabbt på förändringar i processen bör de önskade polerna för det slutna systemet placeras i origo. Emellertid ger dessa poler stora överslängar för utsignalen. I stället beräknas därför två av dessa poler genom att sätta dämpningen $\zeta_o = 0.7$ och egenvinkelfrekvensen $\omega_o = 20$ 1/s för det kontinuerliga systemet. Sedan är det enkelt att med hjälp av dessa två kontinuerliga poler räkna fram den diskreta motsvarigheten. De kontinuerliga polerna blir

$$(4.15) \qquad \begin{cases} s_1 = \alpha + i\beta \\ s_2 = \alpha - i\beta \end{cases}$$

där α och β ges av

$$(4.16) \qquad \begin{cases} \alpha = -\omega_o \zeta_o \\ \beta = \omega_o \sqrt{1-\zeta_o^2} \end{cases}$$

och därigenom fås de diskreta polerna

$$(4.17) \qquad z_1 = e^{s_1 h} \quad \text{och} \quad z_2 = e^{s_2 h}$$

De två andra polerna z_3 och z_4 bör väljas mindre än 0.1 på den positiva reella axeln.

Den andra styrlagen är LQ-reglering (Linear Quadratic Control). Om systemet modelleras som brusfritt, så finns en entydig styrlag

$$(4.18) \qquad u_t = -L_t x_t$$

sådan att straffunktionen

$$(4.19) \qquad J = \sum_{t=0}^{N-1} x_t^T Q x_t + u_t^T R u_t$$

minimeras under bivillkoren

$$(4.20) \qquad x_{t+1} = \Phi x_t + \Gamma u_t$$

där Q är en straffmatris för tillstånden x_t (dim Q=4x4)
och R en straffskalär för insignalen.
Ibland kan det vara lämpligare att straffa utsignalerna y istället för tillstånden x varför vi istället minimerar straffunktionen

$$(4.21) \qquad J = \sum_{t=0}^{N-1} y_t^T Q_y y_t + u_t^T R_y u_t$$

med bivillkoren

$$(4.22) \qquad \begin{cases} x_{t+1} = \Phi x_t + \Gamma u_t \\ y_t = C x_t \end{cases}$$

För det <u>första</u> kriteriet (4.19) finns då en entydig styrlag sådan att

$$(4.23) \qquad L_t = \left(R + \Gamma^T S_{t+1} \Gamma\right)^{-1} \Gamma^T S_{t+1} \Phi$$

där S_{t+1} ges av den s k Riccati-ekvationen

$$(4.24) \qquad S_t = \Phi^T S_{t+1}\Phi + Q - L_t^T(R+\Gamma^T S_{t+1}\Gamma)L_t$$

För det andra kriteriet (4.21) byts Q ut mot C^TQC. Emellertid är det endast möjligt att lösa den rekursiva Riccati-ekvationen bakåt i tiden, d v s givet S_N vid tidpunkten t=N fås lösningarna $S_{N-1},...,S_0$ i ovannämnda ordning. Därför beräknas istället den stationära lösningen till Riccati-ekvationen varför S ges av ekvationen nedan

$$(4.25) \qquad 0 = S - \Phi^T S\Phi + \Phi^T S\Gamma(R+\Gamma^T S\Gamma)^{-1}\Gamma^T S\Phi - Q$$

I sektion (4.5) beskrivs hur ekv (4.25) kan lösas.

4.4 Justering av styrsignalen

Givet ekvation (4.3) och styrlagen $u_t=-L_tx_t$ blir insignal-utsignal-relationen med avseende på den första utsignalen (y_{1t}):

$$(4.26) \qquad y_{1t} = C_1(qI - \Phi + \Gamma L)^{-1}\Gamma r_t$$

Här är C_1=första raden i C-matrisen och r_t är börvärdet. Stationärt (d v s q=1) skall om börvärdet hålls konstant=r, gälla $y_{1t}\equiv r$ varför börvärdet skalas med skalfaktorn

$$(4.27) \qquad \frac{1}{C_1\cdot(I-\Phi+\Gamma L)^{-1}\Gamma}$$

För kontinuerliga system eller diskreta system på deltaoperator kan enhetsmatrisen I i ekv (4.27) strykas.

4.5 Deltaoperatorn

Vid simulering av det inre tillståndsloopen förekom kraftiga oscillationer. En möjlig orsak kunde vara att systemet är dåligt konditionerat. Dessutom när samplingstidlängden h måste vara tillräckligt kort för att kunna styra missilen tillfredsställande uppstår ytterligare ett problem. När $h \to 0$ kan Φ och Γ enligt ekv (4.3) - (4.5) skilja sig med flera tiopotenser vilket kan försämra de numeriska beräkningarna ytterligare.

För det första problemet finns i rapporten följande 3 metoder, nämligen

- styrbar och modal kanonisk form
- balansering
- balansering och modelreduktion

Emellertid blir den styrbara kanoniska sämre ju högre ordning modellen har för att sedan bryta ihop vid ordning n=5 eller högre. Ett annat sätt är balansering av modellen. Sektion 4.6 går närmare in på hur det går till att balansera en diskret modell för deltaoperatorn (δ).

För det andra problemet kan δ-operatorn användas istället för framåtskift-operatorn q (eller z) som tidigare i kapitel 4. Sektion 4.5 ersätter alltså sektion 4.2 - 4.4 där δ-operatorn används istället för q-operatorn.

δ-operatorn definierar en s k generaliserad derivata

$$(4.28) \qquad \frac{x(t+\Delta)-x(t)}{\Delta} = \left[\frac{q-1}{\Delta}\right]\cdot x(t) = \delta x(t)$$

där Δ motsvarar samplingstidslängden h.
I fortsättningen av denna kapitel kommer h att ersättas med Δ för att markera skillnaden mellan diskreta system på q-form och diskreta system på δ-form. δ-operatorn definieras alltså

$$(4.29) \qquad \delta \triangleq \frac{q-1}{\Delta}$$

Det kontinuerliga systemet fås då som ett specialfall då $\Delta \to 0$ så att

$$(4.30) \qquad \delta x(t) \to \frac{d}{dt}x(t) \qquad \text{då } \Delta \to 0$$

Tillståndsmodellen enligt ekv (4.3) omformuleras till

$$(4.31) \qquad \begin{aligned} x_{t+\Delta} &= Ix_t + (\Phi - I)x_t + \Gamma u_t + Gw_t \\ y_t &= Cx_t + Du_t + e_t \end{aligned}$$

Tillståndsmodellen för det diskreta systemet på δ-form liknar det kontinuerliga systemet enligt ekv (4.2) varför ekv (4.31) skrives enligt nedan

$$(4.32) \qquad \begin{aligned} \delta x_t &= A_\delta x_t + B_\delta u_t + G_\delta w_t \\ y_t &= Cx_t + Du_t + e_t \end{aligned}$$

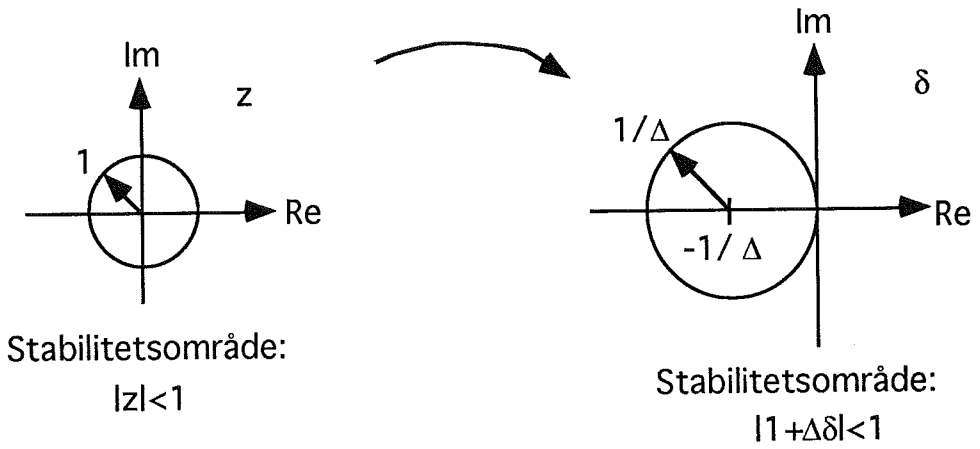
Genom att använda (4.28) och (4.31) kan därför tillståndsmatriserna i ekv (4.32) beräknas enligt följande

$$(4.33) \qquad \begin{cases} A_\delta = \dfrac{\Phi - I}{\Delta} \\ B_\delta = \dfrac{\Gamma}{\Delta} \end{cases} \quad \text{och} \quad G_\delta = \frac{1}{\Delta}G$$

Den digitala datorn arbetar alltså med tillståndsmodellen på δ-form enligt ekv (4.32). Alla nya tillståndsvärden för nästa tidpunkt t+Δ kan enkelt beräknas m h a (4.28), d v s

(4.34) $x(t+\Delta) = x(t) + \Delta\delta x(t)$

Detta motsvarar alltså en linjärisering av modellen och en påföljande addition av det gamla värdet. Detta bör leda till att de numeriska egenskaperna förbättras.
Sektion 4.6 visar hur modellen kan balanseras på δ-form och reduceras till en lägre modelordning . Denna reducerade modell kan sedan överföras på kanonisk form.
Resten av denna sektion beskriver hur observererarna och styrlagarna kan överföras på δ-form. Till sist vill jag kommentera att Matlab inte har något programstöd alls för reglering av system på δ-form utan bara för kontinuerliga system och för diskreta system på q-form.
δ-operatorn motsvarar alltså följande transformation av det stabila området enligt figuren nedan



Polerna på δ-form transformeras som

(4.35) $p_\delta = \frac{p_q - 1}{\Delta} = \frac{e^{p_s \Delta} - 1}{\Delta}$

där p_q är de diskreta polerna (i z-området) och p_s är de kontinuerliga polerna.
Emellertid medför transformationen från q-operatorn till δ-operatorn att de numeriska felen från den diskreta modellen enligt ekv (4.3) förstoras. Om det kontinuerliga systemet från ekv (4.2) är välkänt bör istället detta system transformeras till δ-formen enligt följande

(4.36) $\begin{cases} A_\delta = \Omega A \\ B_\delta = \Omega B \end{cases}$

där Ω väljes som tidsmedelvärdet av $e^{A\Delta}$, d v s

(4.37) $\Omega = \frac{1}{\Delta}\int_0^\Delta e^{A\tau} d\tau$

Detta uttryck kan beräknas enligt ekv (4.5) där B byts ut mot enhetsmatrisen I.
Användes observerare eller styrlagar som tillämpar polplaceringsprincipen är det enkelt att beräkna K_δ och L_δ genom att ersätta matriser och poler på δ-form enligt ekv (4.33) och (4.35). Däremot är det besvärligare att beräkna de optimala styrlagarna och Kalmanfiltret trots att samma transformationer används.

Kalmanfilter: Vi vill alltså ha ett kalmanfilter på δ-form för estimeringen av tillstånden x. Genom att ersätta kovariansmatriserna R_{1d} och R_{2d} med följande definition av spektraltätheterna

$$(4.38)\qquad \begin{cases} \Omega_\delta \triangleq \Delta R_{1\delta} = \frac{1}{\Delta} R_{1d} \\ \Gamma_\delta \triangleq \Delta R_{2d} = \Delta R_{2d} \end{cases}$$

och att utnyttja transformationerna vilka ges av ekv (4.32) och (4.33) fås då det diskreta kalmanfiltret på δ-form. Genom att sätta in dessa matriser i uppdateringsformlerna för det diskreta kalmanfiltret i sektion 4.2 erhålles formlerna nedan.

$$(4.39)\qquad \delta\hat{x}(t) = A_\delta \hat{x}(t) + K_\delta(t)\left[y(t) - C_\delta \hat{x}(t)\right]$$

$$(4.40)\qquad K_\delta(t) = \frac{1}{\Delta} K(t) = (I + \Delta A_\delta) P_\delta(t) C_\delta^T \cdot \left[\Gamma_\delta + \Delta C_\delta P_\delta(t) C_\delta^T\right]^{-1}$$

$$(4.41)\qquad \delta P_\delta(t) = \Omega_\delta + P_\delta(t) A_\delta^T + A_\delta P_\delta(t) + \Delta A_\delta P_\delta(t) - K_\delta(t)\left[\Gamma_\delta + \Delta C_\delta P_\delta(t) C_\delta^T\right] K_\delta(t)$$

Linjär-kvadratiska styrlagar (LQ-reglering): Den generella styrlagen på δ-form lyder

$$(4.42)\qquad u(t) = -L_\delta(t) x(t)$$

Samma transformation från q-form till δ-form via ekv (4.23) ger alltså

$$(4.43)\qquad L_\delta(t) = \left[R_\delta + \Delta B_\delta^T S_\delta(t) B_\delta\right]^{-1} \cdot B_\delta^T S_\delta(t) \cdot \left[I + \Delta A_\delta\right]$$

där straffmatriserna kan enkelt fås genom följande definitioner

$$(4.44)\qquad \begin{cases} Q_\delta \triangleq \frac{Q}{\Delta} \\ R_\delta \triangleq \frac{R}{\Delta} \end{cases}$$

där Q och R är straffmatriserna för det diskreta systemet på q-form. Riccati-ekvationen för δ-operatorn fås i följande rekursiv form bakåt i tiden (se sektion 4.3):

$$(4.45)\quad \bar{\delta}S_\delta(t) \triangleq Q_\delta + A_\delta^T S_\delta(t) + S_\delta(t)A_\delta + \Delta A_\delta^T S_\delta(t)A_\delta - L_\delta(t)\left[R + \Delta B_\delta^T S_\delta(t)B_\delta\right]L_\delta(t)$$

där $\bar{\delta}S_\delta(t) \triangleq -\left(\frac{S_\delta(t) - S_\delta(t-\Delta)}{\Delta}\right)$ är den omvända generaliserade derivatan.

Den stationära lösningen till Riccatiekvationen erhålles då

$$(4.46)\quad 0 = Q_\delta + A_\delta^T S_\delta + S_\delta A_\delta + \Delta A_\delta^T S_\delta A_\delta - \left(I + \Delta A_\delta^T\right)S_\delta B_\delta\left(R_\delta + \Delta B_\delta^T S_\delta B\right)^{-1} B_\delta^T S_\delta\left(I + \Delta A_\delta\right)$$

Att lösa matrisekvationen ovan är besvärligt men det finns ett sätt där man beräknar egenvärdena till den generaliserade Hamilton-matrisen vilket defineras

$$(4.47)\quad \begin{bmatrix} -S_\delta & I \end{bmatrix} \cdot M \cdot \begin{bmatrix} I \\ S_\delta \end{bmatrix} = 0 \qquad \text{där } M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}$$

Efter en mödosam räkning kan ekv (4.46) omskrivas till Hamiltonformen enligt ovan där delmatriserna för M fås som

$$(4.48)\quad \begin{cases} M_{11} = A_\delta + \Delta B_\delta R_\delta^{-1} B_\delta^T \left(I + \Delta A_\delta^T\right)^{-1} Q_\delta \\ M_{12} = B_\delta R_\delta^{-1} B_\delta^T - \Delta B_\delta R_\delta^{-1} B_\delta^T \left(I + \Delta A_\delta^T\right)^{-1} A_\delta^T \\ M_{21} = \left(I + \Delta A_\delta^T\right)^{-1} Q_\delta \\ M_{22} = -\left(I + \Delta A_\delta^T\right)^{-1} A_\delta^T \end{cases}$$

De generaliserade egenvärdena till matrisen M beräknas enligt följande formel

$$(4.49)\quad M \cdot \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \cdot \Lambda$$

där Λ är diagonalmatrisen för egenvärdena (se Matlabprogrammet *eig*). Lösningarna till ekvationen för diagonaliseringen är X_1 och X_2. Då ger följande algoritm lösningen till den stationära Riccati-ekvationen:

Om systemet (A_δ, B_δ) är styrbart så finns det en stabil lösning. I så fall kan matrisen M med dim=[2n,n] kan grupperas i två diskjunkta mängder Γ_1 och Γ_2. Då kan vi välja Γ_1 så att denna mängd innehåller enbart de för δ-tillståndet n st stabila poler och Γ_2 innehåller endast n st instabila poler.

Elementen i X_1 och X_2 omgrupperas genom att sortera egenvärdena i Λ så att $|1+\Delta\lambda_1| > |1+\Delta\lambda_2| > ... > |1+\Delta\lambda_{2n}|$. Därefter testar man om dessa mängder Γ_1 och Γ_2 uppfyller kriteriet ovan. I så fall finns en entydig lösning till den stationära Riccati-ekvationen vilket ges

$$S_\delta = \tilde{X}_2\tilde{X}_1^{-1} \tag{4.50}$$

där $\tilde{X}_1$ och $\tilde{X}_2$ är de omgrupperade matriserna enligt ovan.

4.6 Balansering och modelreduktion på deltaform

Denna sektion går ut på att omgruppera elementen i A_δ,B_δ och C_δ så att modellen blir minimumfas, d v s har den lägsta möjliga energiinnehållet och därför optimalt stabilt. Om systemet är styrbart så kan vi välja någon insignal u till systemet med en given energi för att nå tillståndet x_r vid tiden t=0 betingat x_0=0. Då är energin

$$J_r = \int_0^{t_f} u(t)^T u(t)dt = x_0^T P(t_f)x_0 \tag{4.51}$$

där $P(t_f)$ är den styrbara gramianen. Det gäller att $u(t)=BE(A_\delta,t)x_0$ vilket ger

$$P(t_f) = \int_0^{t_f} E(A_\delta,t)^T BB^T E(A_\delta,t)dt \tag{4.52}$$

$E(A_\delta,t)$ är den generaliserade matrisexponentialen vilket härledes iterativa beräkningar av tillståndsmodellen. Vi nöjer oss med den explicita formen:

$$E(A_\delta,t) = (I + A_\delta\Delta)^{t/\Delta} \tag{4.53}$$

För fallet $\Delta \to 0$ blir $E(A,t)=e^{At}$ och för z-transformen $E(\Phi,t)=\Phi^N$, där $N=t/\Delta$. Genom en differentiering av den styrbara gramianen fås alltså

$$\delta P(t) = A_\delta P(t) + P(t)A_\delta^T + \Delta A_\delta P(t)A_\delta^T + B_\delta B_\delta^T \tag{4.54}$$

Om alla egenvärden till A_δ ligger inom det stabila området så existerar en stationär lösning vilket ges av den generaliserade Lyaponovekvationen.

$$\alpha P + P\beta = -\gamma \tag{4.55}$$

Genom att sätta högerledet i (4.54) lika med noll och överföring till standardformen (4.55) fås koefficienterna α, β och γ som

$$(4.56)\quad \begin{cases} \alpha = A_\delta^{-1} \\ \beta = \left(I + \Delta A_\delta^T\right) A_\delta^{-T} \\ \gamma = -A_\delta^{-1} B_\delta B_\delta^T A_\delta^{-T} \end{cases}$$

Nu önskar vi veta vilken tillståndsenergi som är nödvändig då det inte finns någon insignal till systemet för att få en viss utsignalenergi inom tiden $(0,t_f)$. I så fall blir energin

$$(4.57)\quad J_r = \int_0^{t_f} y(t)^T y(t)dt = x_0^T \mathcal{Q}(t_f) x_0$$

Det gäller att $y(t)=CE(A_\delta,t)x_0$ vilket den oberverbara gramianen $Q(t_f)$ blir

$$(4.58)\quad \mathcal{Q}(t_f) = \int_0^{t_f} E(A_\delta,t)^T C_\delta^T C_\delta E(A_\delta,t)dt$$

Diffentiering av (4.58) ger då den generaliserade differentialekvationen

$$(4.59)\quad \delta\mathcal{Q}(t) = A_\delta^T \mathcal{Q}(t) + \mathcal{Q}(t)A_\delta^T + \Delta A_\delta^T \mathcal{Q}(t) A_\delta + C_\delta^T C_\delta$$

Härmed fås den stationära lösningen av (4.59) på standardformeln för Lyaponov-ekvationen (4.55)

$$(4.60)\quad \begin{cases} \alpha = A_\delta^{-T} \\ \beta = \left(1 + \Delta A_\delta\right) A_\delta^{-1} \\ \gamma = A_\delta^{-T} C_\delta^T C_\delta A_\delta^{-1} \end{cases}$$

En balanserad modell är alltid minimumfas, d v s både observerbar och styrbar. Om tillståndet x transformeras genom att sätta $z=Tx$ så fås det nya systemet

$$(4.61)\quad \begin{cases} z = T^{-1} A_\delta T x + T^{-1} B_\delta u \\ y = C_\delta T z \end{cases}$$

med de nya gramianerna

$$(4.62)\quad \begin{cases} P_{\delta z} = T P_\delta T^T \\ \mathcal{Q}_{\delta z} = T^{-T} \mathcal{Q}_\delta T^{-1} \end{cases}$$

Produkten ska alltså vara tidsinvariant så att energin bevaras vilket är kriteriet för ett minimumfas system.

$$(4.63)\quad P_{\delta z}\mathcal{Q}_{\delta z} = T(P_\delta \mathcal{Q}_\delta)T^{-1}$$

Kvadratrötterna av egenvärdena till produkten $P_\delta Q_\delta$ definieras som $\sigma_1, ...,\sigma_n$ vilka formuleras genom följande samband

$$(4.64) \quad \sigma_i = \sqrt{\lambda_i(P_\delta Q_\delta)}$$

På så sätt fås de transformerade gramianerna för det nya z-tillståndet

$$(4.65) \quad P_{\delta z} = Q_{\delta z} = \Sigma = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{bmatrix}$$

Här anges explicit algoritmen som finner denna transformationsmatrix T

$$(4.66) \quad \begin{cases} Q_\delta = R^T R \quad \text{(choleskyfaktoriserng)} \\ S \triangleq R P_\delta R^T = U \Sigma U^T \quad \text{(Hankel sin gular faktorisering)} \\ \text{med bivillkoret } U^T U = I \\ \Sigma = \text{diag}\{\sigma_1^2,...,\sigma_n^2\} \\ T = \Sigma^{-1/4} U^T R \end{cases}$$

De nya transformerade matriserna $A_{\delta z}$, $B_{\delta z}$ och $C_{\delta z}$ bör ha den egenskapen att de två gramianerna $P_{\delta z}=Q_{\delta z}=\text{diag}(\sigma_1,...,\sigma_n)$ har egenvärdena sorterade i avtagande ordning $\sigma_1>\sigma_2>...>\sigma_n$.

Det är också önskvärt att eliminera de tröga tillstånden, alltså de som har små värden på den styrbara eller observerbara gramianen. Denna eliminering sker genom att stryka det tillstånd z_i med raden i och kolonnen i som har den minsta gramianen σ_i i tillståndsmodellen. Vi kan ställa upp tillståndsmodellen enligt följande tillståndsmodell

$$(4.67) \quad \begin{cases} \begin{bmatrix} \delta z_1 \\ \delta z_2 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} u \\ y = \begin{bmatrix} C_1 & C_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + Du \end{cases}$$

Dynamiken i z_2 försummas varför $\delta z_2 \approx 0$. Eliminering av z_2 ger alltså det nya reducerade systemet

$$(4.68) \quad \begin{cases} \delta z_1 = \left[A_{11} - A_{12}A_{22}^{-1}A_{21}\right] z_1 + \left[B_1 - A_{12}A_{22}B_2\right] u \\ y = \left[C_1 - C_2 A_{22}^{-1} A_{21}\right] Z_1 + \left[D - C_2 A_{22}^{-1} B_2\right] u \end{cases}$$

4.7 Filtrering

Tanken med δ-operatorn i sektion 4.5-4.6 var att eliminera oscillationerna som verkade bero på numeriska fel. Dock fanns oscillationerna kvar ändå.
Dessa oscillationer av styrsignalen bestod av en enda frekvens. Jämförelser med Bode-diagrammen av det kontinuerliga öppna systemet för missilen visade att denna oscillationfrekvens sammanföll med resonansfrekvensen ω_p. Resonanstoppen var ganska kraftig vilken var 80 dB jämfört med lågfrekvensförstärkningen 60 dB. Beroende på farten och höjden hos missilen varierade ω_p mellan 0.8 och 3 Hz.
Denna resonanstopp kan elimineras med ett digitalt bandstoppfilter (notchfilter) som försvagar förstärkingen vid resonansfrekvensen. Detta filter inkopplas på reglerfelet δ eller styrsignalen u. Problemet är att finna ett sådant bandstoppfilter som klarade av variationerna i resonansfrekvensen utan att bandbredden blir för stor. Emellertid är sådana filter ganska svåra att hitta eller är besvärliga att dimensionera korrekt.

Ett sådant filter är Chebysevs bandstoppfilter typ II. I Matlab finns ett standardpaket för signalbehandling inklusive digitala filter. Här skrives bara de nödvändiga specifikationerna.

Nyquistfrekvensen ω_N definieras som

$$\omega_N = \frac{1}{2\Delta} \tag{4.69}$$

där Δ är samplingstidslängden. Cutoff-frekvensen definieras som

$$\omega_c = \frac{\omega_p}{\omega_N} \tag{4.70}$$

där $\omega_c \in [0,1]$

Om $\omega_p \approx 2$ Hz så blir $\omega_c \approx 0.5$. Vidare bör filtret vara av ordning 3. Simuleringar har visat att bandbredden bör vara 0.01 med 0.01 dB överslängar (ripple) i bandstoppet.
Detta filter fick bort alla oscillationer fullständigt för linjära system (se sektion 6.1). Även för olinjära system reducerades oscillationerna avsevärt (se sektion 6.2).

5. Identifiering och uppbyggnad av modeller

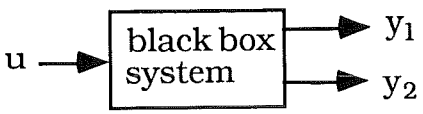
Hittills har vi förutsatt linjära modeller där Φ, Γ och C tillförts utifrån. Detta kan ske på olika sätt, t ex genom hämtning från modellbibliotek $\{M_i\}_{i=1}^{m}$ där modellerna har segmenterats på något lämpligt sätt. Beslutet om bästa modell M_i fattas härvid på basis av givna värden på farten v och höjden h. Sedan kan man försöka utvidga systemet genom estimering av modellerna on-line med hjälp av gamla indata som $\hat{\theta}$ och P för ett bestämt (v,h)-modellområde som hämtats från ovanämda modellbibliotek.
Oavsett vilken metod som används måste modellbiblioteket segmenteras och byggas upp med tillräckligt bra initialvärden före simuleringen. Jag rekommenderar att man har en modell bestämd, nämligen det (v,h)-område när missilen träffar målobjektet och expanderar den i modell-biblioteket för att sedan successivt byta ut gamla värden mot nya bättre dito på $\hat{\theta}$, P och eventuellt R_{1d} och R_{2d}.
Detta kapitel utreder också de praktiska möjligheterna att bygga upp modellbiblioteket på basis av en given segmentation.

5.1 Processidentifiering

Om de tidsvariabla parametrarna v(t) och h(t) antas vara konstanta för ett visst tidsintervall fås alltså en linjär modell enligt ekv (4.2). Givet A, B och C kan sedan denna kontinuerliga linjära modell diskretiseras så att ekv (4.3) erhålles.
För vissa verkliga processer är det svårt, för att inte säga vanskligt, att beräkna en kontinuerlig modell som beskriver systemet tillräckligt noggrant. Robotmodellen enligt ekv (3.1) är endast approximativ.
Istället kan ekv (4.3) skrivas på en standardform som innehåller ett antal obekanta parametrar, och därefter kan verkligt uppmätta in-utdata användas för att beräkna värdena hos dessa parametrar så att den sålunda ansatta modellens utsignal följer det verkliga systemets så bra som möjligt för den givna insignalen.
För detta finns det färdiga standardrutiner, t ex RLS-algoritmen som minimerar variansen för parameterfelen. Denna algoritm kan identifiera en s k ARX-modell för SISO-system, d v s

$$(5.1) \quad A(q^{-1})y_t = B(q^{-1})u_t + e_t$$

Emellertid kompliceras processidentifieringen i vårt fall av att systemet har två utsignaler y_1 och y_2.



I denna sektion beskrives processidentifieringen för vårt SIMO-system.

Tillståndsekvationen $x(t+1)=\Phi x(t)+\Gamma u(t)$ kan skrivas som $x(t)=(qI-\Phi)^{-1}\Gamma u(t)$ varur insignal-utsignal-förhållanderna erhålles för de två utsignalerna:

$$(5.2)\qquad \begin{cases} y_1(t-1) = C_1x(t-1) = C_1(qI-\Phi)^{-1}\Gamma u(t-1) \\ y_2(t-1) = C_2x(t-1) = C_2(qI-\Phi)^{-1}\Gamma u(t-1) \end{cases}$$

där C_1 och C_2 är första respektive andra raden i C-matrisen. Detta kan även skrivas

$$(5.3)\qquad \begin{cases} A\left(q^{-1}\right)y_1 = B_1\left(q^{-1}\right)u \\ A\left(q^{-1}\right)y_2 = B_2\left(q^{-1}\right)u \end{cases}$$

där polynomen $A(q^{-1})$, $B_1(q^{-1})$ och $B_2(q^{-1})$ kan skrivas

$$(5.4)\qquad A\left(q^{-1}\right) = \sum_{i=0}^{n} a_i q^{-i} \qquad \text{med } a_0 = 1$$

$$(5.5)\qquad B_j\left(q^{-1}\right) = \sum_{i=1}^{m} b_{ji} q^{-i} \qquad j = 1,2$$

Detta är två separata in-utsignalbeskrivningar med gemensamt nämnar-polynom $A(q^{-1})$. De två in-utsignalrelationerna kan nu formuleras som

$$(5.6)\qquad \begin{cases} y_1(t) = -\sum_{i=1}^{n} a_i y_1(t-i) + \sum_{i=1}^{m} b_{1i} u(t-i) \\ y_2(t) = -\sum_{i=1}^{n} a_i y_1(t-i) + \sum_{i=1}^{m} b_{2i} u(t-i) \end{cases}$$

där $n > m$. Inför nu följande definitioner:

$$(5.7)\qquad \alpha \triangleq [a_1, a_2, \ldots, a_n]^T,\ \beta_1 \triangleq [b_{11}, b_{12}, \ldots, b_{1m}]^T,\ \beta_2 \triangleq [b_{21}, b_{22}, \ldots, b_{2m}]^T$$

$$(5.8)\qquad \begin{cases} \varphi_{1y}(t) \triangleq [-y_1(t-1), -y_1(t-2), \ldots, -y_1(t-n)]^T \\ \varphi_{2y}(t) \triangleq [-y_2(t-1), -y_2(t-2), \ldots, -y_2(t-n)]^T \\ \varphi_u(t) \triangleq [u_{t-1}, u_{t-2}, \ldots, u_{t-m}]^T \end{cases}$$

Sätt

$$\theta \triangleq \left[\alpha^T \ \beta_1^T \ \beta_2^T\right]^T \tag{5.9}$$

$$\begin{cases} \varphi_1(t) \triangleq \left[\varphi_{1y}^T(t) \ \varphi_u^T(t) \ 0\right]^T \\ \varphi_2(t) \triangleq \left[\varphi_{2y}^T(t) \ 0 \ \varphi_u^T(t)\right]^T \end{cases} \tag{5.10}$$

så fås

$$\begin{cases} y_1(t) = \varphi_1^T(t)\cdot\theta \\ y_2(t) = \varphi_2^T(t)\cdot\theta \end{cases} \tag{5.11}$$

Vi definierar slutligen

$$y(t) \triangleq \left[y_1(t)\, y_2(t)\right]^T \tag{5.12}$$

och

$$\phi(t) \triangleq \left[\varphi_1(t)\ \varphi_2(t)\right] \quad \text{(matris med två kolonner)} \tag{5.13}$$

Den kvadratiska felfunktionen definieras på standardform

$$V_t(\theta) = \frac{1}{t}\sum_{k=1}^{t}\left(y(k)-\phi^T(k)\theta\right)^T\left(y(k)-\phi^T(k)\theta\right)$$

Denna felfunktion skall minimeras m a p parametervektorn θ. Om optimum kallas $\hat{\theta}_t$, så gäller alltså

$$\frac{\partial}{\partial\theta}V_t(\theta)\Big|_{\theta=\hat{\theta}_t} = 0$$

Genom derivering och användning av matrisinversionslemmat vilket inte görs här i rapporten (se referens [4] eller [5]) fås följande två vanliga uppdateringsformler på rekursiv form:

$$\hat{\theta}_{t+1} = \hat{\theta}_t + P_{t+1}\phi(t+1)\left[y(t+1)-\phi^T(t+1)\hat{\theta}_t\right] \tag{5.14}$$

$$P_{t+1} = P_t - P_t\phi(t+1)\cdot\left(I+\phi^T(t+1)P_t\phi(t+1)\right)^{-1}\phi^T(t+1)P_t \tag{5.15}$$

där en 2x2 matris inverteras.

Detta ger uppdatering av estimat-vektorn $\hat{\theta}_t$ och kovariansmatrisen P_t i ett steg, varvid de båda utsignalerna $y_1(t)$ och $y_2(t)$ behandlas samtidigt.

RLS-algoritmen (Recursive Least Square method) kan via ekv (5.14) och (5.15) omformuleras till

$$(5.16) \quad K_t = P_{t-1}\phi(t)\left(\lambda I + \phi^T(t)P_{t-1}\phi(t)\right)^{-1}$$

$$(5.17) \quad \hat{\theta}_t = \hat{\theta}_{t-1} + K_t\left[y(t) - \phi^T(t)\hat{\theta}_{t-1}\right]$$

$$(5.18) \quad P_t = P_{t-1} - K_t\phi^T(t)P_{t-1}$$

där estimatvektorn $\hat{\theta}_t$ definieras av ekv (5.9) och (5.17), och in-utsignal-sekvensen $\phi(t)$ av ekv (5.13).
I ekv (5.16) har den s k exponentiella glömskefaktorn $\lambda \in [0\ 1]$ tillfogats. Orsaken är att även långsamma tidsvarianta system ska kunna estimeras genom att gamla in-utsignal-värden inte tillordnas lika stor betydelse som nya värden, d v s gamla värden glöms bort med simuleringstiden. Normalt bör λ vara mellan 0,90 och 0,99. För tidsinvarianta system skall $\lambda=1$ så att alla värden har lika stor betydelse oavsett "ålder".

För att öka bruståligheten hos RLS-algoritmen föreslog Åström (se t ex referens [4]) att man byter ut enhetsmatrisen I mot mätbrusets kovariansmatris R_{2d} i ekv (5.16). Därigenom förbättras konvergensen avsevärt i de fall processen innehåller mätbrus.

RLS-algoritmen är tyvärr inte tillräckligt robust. Biasfelen kvarstår under lång tid och man är inte alltid garanterad ett korrekt estimat för alla slags tidsinvarianta linjära system. Ett sätt att komma förbi detta problem är att faktorisera P_t. För detta ändamål finns två metoder, nämligen Biermans UD-faktorisering och Potters faktorisering.
Biermans metod går ut på att faktorisera så att $P_t=UDU^T$. För SISO-system fungerar det bra men det är matematiskt omöjligt att omformulera UD-faktoriseringsalgoritmen för SIMO-system.
Däremot går det bra att omformulera Potters faktorisering till SIMO-system så att $P_t=SS^T$. Process- och mätbrus kan göra så att estimaten inte blir tillräckligt noggranna varför Potters algoritm modifieras enligt samma ide som Åström föreslog.

Algoritm för processidentifieringen

Bestäm Q_0 genom faktoriseringen $P_0 = \mathcal{Q}_0\mathcal{Q}_0^T$. Beräkna

1) $f_t = \mathcal{Q}^T{}_{t-1}\phi_t$

2) $\beta_t = \lambda R_{2d} + f_t^T f_t$

3) $\alpha_t = \left(\beta_t + \sqrt{\beta_t \cdot \lambda R_{2d}}\right)^{-1}$

4) $L_t = \mathcal{Q}_{t-1} f_t$

5) $\mathcal{Q}_t = \mathcal{Q}_{t-1} - L_t \alpha_t f^T{}_t / \sqrt{\lambda}$

6) Gå till punkt 1.

Då ges förstärkingsvektorn K_t av

$$K_t = L_t \cdot \beta_t^{-1}$$

vilket bör beräknas under punkt 4. Kvadratsrotsutdragningen görs elementvis, varför matrisen ur vilken roten skall dras måste ha alla elementen ≥0. Om detta krav inte uppfylls, fås imaginära värden, vilket gör att simuleringen spolieras.

5.2 Processidentifiering på deltaoperator

När δ–operatorn infördes istället för framåtskift-operatorn q i regulatorn (kapitel 4), så vore det onödigt att behöva transformera modellen till deltaform efter processidentifieringen. Därför införes i rapporten processidentifieringen på δ–form. Här anges endast summariskt de fungerande algoritmerna för processidentifieringen. För en djupare förståelse hänvisas läsaren till referens [3].

Den generella RLS-algoritmen sammanfattas enligt

(5.19) $K_t = \alpha P_t \phi \left[\Gamma + \Delta\phi^T(t) P_t \phi(t)\right]^{-1}$

(5.20) $\delta\hat{\theta}_t = K_t\left(y_t - \phi^T(t)\hat{\theta}_t\right)$

(5.21) $\delta P_t = -K_t \phi^T(t) P_t + \Omega_t$

där Ω_t och α kan beräknas enligt följande sex algoritmer:

1) Rekursiv RLS algoritm (tidsinvariant kalmanfilter)

$\Omega=0$, $\alpha=1$

2) Kalman filter modifiering

$\Omega>0$, $\alpha=1$

3) Gradient algoritm

$\Omega_t = L_t\phi^T(t)P_t$, α konstant

4) RLS med glömskefaktor

$$\Omega_t = \left(\frac{1}{1-\Delta\lambda}\right)\left(\lambda P_t - \frac{\Delta P_t\phi^T(t)\phi(t)P_t}{\Gamma+\Delta\phi^T(t)P_t\phi(t)}\right),\quad \alpha=1$$

5) Constant Trace algoritm

$$\Omega_t = \frac{\alpha}{C_1}L_t\phi^T(t)P_tP_t \qquad C_1 = \text{Trace}(P_0)$$

6) EFRA (Exponential Forgetting and Reseting Algorithm)

$$\Omega_t = \sigma P_t - \varepsilon\sigma P_t^2$$
$$\alpha = \sigma,\quad \Gamma = 1+\beta\phi^T(t)\phi(t)$$

Givetvis går det att kvadratrotuppdela P på samma sätt som i Potters algoritm i sektion 5.1. Det ska gälla att $0 \le \alpha \le 1$ och $\Omega_t = \Omega_t^T \ge 0$.

Algoritm 2 förhindrar att $P \to 0$ och tillåter därför algoritmen att följa tidsvariabla parametrar. Gradient-metoden fixerar kovariansmatrisen P till σI så att den kan följa tidsvariabla parametrar. Dock blir konvergensen långsam vid dåliga val av P. Minsta kvadratmetoden med glömskefaktor glömmer gradvis bort gamla data så att nya data lättare kan påverka algoritmen att följa de tidsvariabla parametrarna. EFRA glömmer också bort gamla data men i de fall som $P \to \infty$ p g a dålig excitation så nollställer den automatiskt P till I/ε.

Processidentifieraren estimerar ARX-modellen

$$(5.22)\quad A\left(\delta^{-1}\right)y_t = B\left(\delta^{-1}\right)u_t$$

där $A(\delta^{-1})$ och $B(\delta^{-1})$ har samma utseende som $A(q^{-1})$ och $B(q^{-1})$ i ekv (5.4) och (5.5). Kvar återstår problemet med regressionsvektorn $\phi(t)$. Mätvärdena är diskreta, men derivator av grad n-1 i ϕ ska approximeras enligt nedan

$$(5.23)\quad \phi^T(t) = \left[\delta^{n-1}y_f(t),\ldots,y_f(t),\delta^m u_f(t),\ldots,u_f(t)\right]$$

Detta problem kan delvis lösas genom att filtera insignalen u och utsignalerna y. Detta stabila filter benämnes $E(\delta^{-1})$ med ordning n, d v s

$$\begin{cases} y_f(t) = \dfrac{1}{E(\delta)}y(t) \\ u_f(t) = \dfrac{1}{E(\delta)}u(t) \end{cases}$$

Detta filter kan implementeras som en tillståndsmodell inne i den digitala regulatorn. Om parametervektorn θ definieras enligt följande

$$(5.24)\quad \theta^T(t) = [e_1 - a_1,\ldots,e_n - a_n,\ldots,b_{11},\ldots,b_{1m},b_{21},\ldots,b_{2m}]$$

då kan vi finna $y(t)=\phi^T(t)\theta$ om ϕ filtreras med tillståndsmodellen

$$(5.25)\quad \begin{cases} \delta\phi_{1y} = \overline{E}\phi_{1y} + \beta y_1 \\ \delta\phi_{2y} = \overline{E}\phi_{2y} + \beta y_2 \\ \delta\phi_u = \overline{E}\phi_u + \beta u \end{cases}$$

där $\overline{E}$ och β ges av

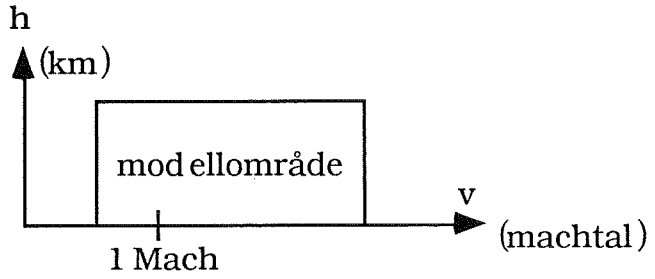
$$(5.26)\quad \overline{E} = \begin{bmatrix} -e_1 & -e_2 & \cdots & -e_n \\ 1 & 0 & \cdots & 0 \\ \cdots & \ddots & \ddots & \vdots \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \beta = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

5.3 Införande av processidentifiering i regulatorn

För att processidentifieringen skall fungera måste insignalen *u* vara tillräckligt exciterande och ha tillräckligt stor amplitud. Dessutom måste den vara styckvis konstant, d v s samplad. Diskretiserat vitt brus är den bästa insignalen. Då störs systemet ordentligt varefter den interna strukturen lättare kan beräknas (s k persistent excitation).
Processidentifieringen kan göras separat för varje modellfall eller försöksuppställning vilket kallas offline-identifiering. Sådana simuleringar kan enkelt utföras varefter ett modellbibliotek successivt kan byggas upp.
Tanken var från början att införa processidentifieringen direkt i regulatorn, vilket kallas online-identifiering (se figuren i sektion 4.1). En annan benämning för online-identifiering tillsammans med regulatorn är "indirect selftuning adaptive regulator".
Men om regulatorns insignal till systemet skiftar för kraftigt hinner den inte med. Då blir det oscillationer i utsignalen vilket inte är önskvärt. Om å andra sidan insignalen inte är tillräckligt exciterande kan inte identifieraren göra bra estimat. Ett annat krav är att initialvärdena på $\hat{\theta}_0$ inte får avvika för mycket från θ, och att kovariansmatrisen P är tillräckligt liten.
En kompromiss mellan offline och online identifiering är att för varje modellområde hämta initialvärden för parametervektorn och P-matrisen från modellbiblioteket och identifiera online. Detta kallas för autotuning. Ett annat sätt vore att göra några provskjutningar, lagra resultatet och sedan offline identifiera under hela banan och på så sätt förbättra modellbiblioteket.

5.4 Segmentation av modellområdet

Ett modellområde definieras så att det omsluter alla för systemet väsentliga modeller. I vårt fall avgränsas modellområdet enbart av hur högt upp över markytan som missilen kan arbeta, och hur hög och låg missilens hastighet normalt är.
Modellområdets utsträckning bestäms alltså enbart av höjden h och missilhastigheten v. Det kan vara allt från havsytan upp till 10-20 km höjd och från ett par hundra m/s till flera Mach. Det beror helt på missilens prestanda. Det enklaste modellområdet visas i figuren nedan

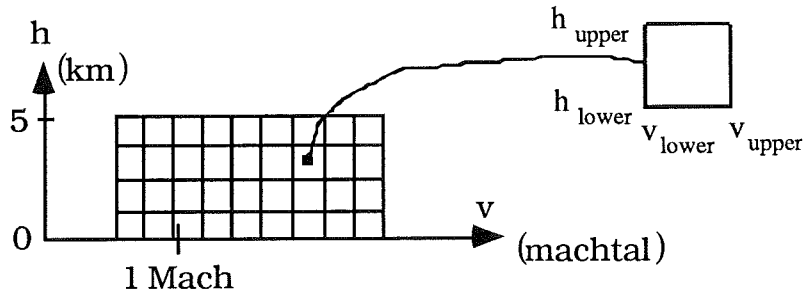


Med segmentation av modellområdet menas en bestämd uppdelning av modellområdet i mindre delar. Dessa delar kallas för modeller. Varje modell har ett nummer som benämnes modellindex. Modellområdet måste definieras först innan det kan segmenteras. Modellområdet kan vara en delmängd i tillståndsrummet eller (v,h)-rummet. I vårt fall tillhör modellområdet en delmängd av (v,h)-rummet.
På grund av att inparametrarna (v,h) till systemet är tidsvariabla och olinjära räcker inte en enda modell för hela modellområdet. Därför måste ett visst antal punkter som motsvarar (v,h) väljas. I dessa utvalda punkter beräknas exakta modeller med hjälp av processidentifiering. Den modell som beräknats i den utvalda punkten antages gälla i en viss närhet av ovannämnda punkt. Det blir givetvis större modellavvikelser ju mer man avlägsnar sig från den punkt för vilken modellen beräknats. I många fall kan inte denna utsträckning göras för hela modellområdet (= en enda modell) utan att missilens styregenskaper försämras avsevärt varför flera punkter måste väljas.
Frågan är nu hur dessa punkter i (v,h)-planet skall förläggas och hur många punkter (=modeller) som behövs. Det måste alltså finnas något slags kriterium för segmentationen av modellområdet. Det finns många kriterier. På detta område är forskningen ännu i sin linda. Emellertid berör de flesta forskarrapporter enbart segmentering av tillståndsrummet där processen genomgår plötsliga men sällsynta förändringar av sina systemparametrar.
I vårt fall sker emellertid förändringar av v och h, och därmed också av systemparametrarna, på ett mera kontinuerligt och långsamt sätt.
Föregående diskussion indikerar alltså att man är tvungen att hitta icke-optimala kriterier som kan verka "godtyckliga" men ändå är förnuftiga.
Det enkla kriterium som använts i detta arbete för att en segmentering för hela (v,h)-planet skall godkännas är att utsignalen a_z inte får avvika mer än 20 % från börvärdet a_{zref}, d v s

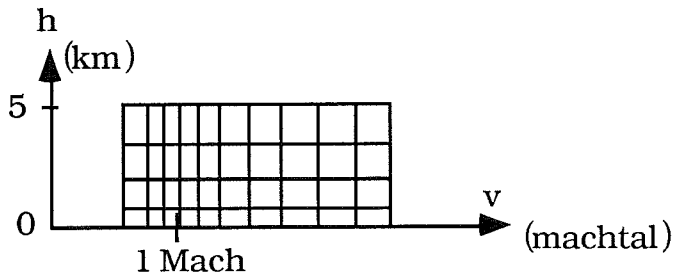
$$\left|\frac{a_z - a_{zref}}{a_{zref}}\right| \leq 0.20 \qquad (5.27)$$

under hela simuleringstiden (normalt 5-10 sekunder). Dessutom är det extra noga med att utsignalen följer börvärdet väl under slutfasen av simuleringen, eftersom det är viktigt att inte låta bomavståndet bli för stort (några meter).

Figuren nedan visar ett exempel på den enklaste tänkbara segmentationen av modellområdet.



där v_{lower}, v_{upper}, h_{lower} och h_{upper} utgör en viss modells undre och övre gränser för v och h. Med det använda kriteriet blir emellertid segmentationsintervallen variabla enligt figuren



Emellertid har hastighetens variation mycket större betydelse för modellindelningen än flyghöjdens variation. I detta arbete indelas därför endast v-axeln variabelt.
Eftersom Matlab endast kan hantera matriser måste modellbiblioteket implementeras som en matris. Lämpligen beskrives segmentationen av modellområdet i tabellform enligt figuren nedan

Modellindex i	v_{lower}	v_{upper}	h_{lower}	h_{upper}
⋮	⋮	⋮	⋮	⋮
9	1.1	1.2	1.5	2.0
10	1.1	1.2	2.0	2.5
11	1.2	1.3	0	0.5
12	1.2	1.3	0.5	1.0
⋮	⋮	⋮	⋮	⋮

Modellindexet är modellens nummer. Tabellen är ordnad enligt den systematiken att v-intervallen i första hand ska ordnas efter växande värden och sedan skall h-intervallen ordnas efter växande värden inom varje v-intervall.

5.5 Modellbeslutare

Hittills har vi antagit att missilfarten v och flyghöjden h kunnat mätas exakt. På så sätt har modellbeslutaren kunnat göras enkel och snabb. I verkligheten är alla mätningar behäftade med mätbrus. Det antages att det inte finns grova mätfel, och därför kan v och h delas upp i en deterministisk del och en stokastisk del enligt följande formel:

(5.28)
$$\begin{aligned} v &= v_{det} + \upsilon_v \\ h &= h_{det} + \upsilon_h \end{aligned}$$

där $\upsilon_v \in N(0,\sigma_v)$ och $\upsilon_h \in N(0,\sigma_h)$

Då är uppgiften att konstruera en modellbeslutare som inte tar fel modell när en störning inträffar. Dock har föregående resonemang i sektion 5.4 visat att det är mycket svårt att teoretiskt finna en sådan modellbeslutare för vår aktuella robotmodell. Detta faktum och tidsbristen har gjort att jag inte kunnat finna en sådan modellbeslutare utan bara genom simuleringar visat vad som händer, när mätningarna av hastigheten v och höjden h störs av kontinuerligt vitt brus.
Logiskt sett är det ingen ide att finna en sådan beslutare, eftersom om störningarna är tillräckligt små, så tar min enkla modellbeslutare bara den modell, som ligger närmast den korrekta, vilket inte gör så stor skillnad. Modellbiblioteket $\{M_i\}_{i=1}^{m}$ implementeras som en tabell vars systematiska uppbyggnad visas enligt figuren nedan

Modellindex i	$\hat{\theta}$	P	R_{ld}	...
⋮				
10				
11	...	...	...	...
12				
⋮				

Alla parametrar $\hat{\theta}$, P o s v ska lagras som radvektorer. Det är enkelt att lägga till nya parametrar i högra delen av tabellen. Radernas längd bestäms entydigt av antalet tillstånd n (= 4 i robotmodellen, och antalet in- och utsignaler (=3 i robotmodellen).

6. Simuleringsresultat

Resultaten delas upp i tre delar, nämligen testning av regulatorn, testning av kinematiken och testning av övriga system. Grunden för redovisningarna av simuleringen sammanfattas i sektion 1.2 (*målsättning, syfte*). Den integrationsmetod som används i hela simuleringen är rk45 med minsta steglängden = 0.2*h, där h är samplingstidslängden. Några andra integrationsmetoder duger inte eftersom hela systemet är starkt olinjärt. Empiriskt är den minsta steglängden 0.2*h den optimala för processidentifieraren och den går ganska bra på de övriga systemen. Toleransen bör sättas ungefär lika med 10^{-5} utom för den seperata testningen av regulatorn där toleransen istället bör sättas ungefär lika med 10^{-2}.
Om dessa riktlinjer följs till punkt och pricka undvikes de flesta fällor för simuleringen utom i vissa fall för processidentifieraren där en brist i programvaran SimuLink har förvållat högst märkvärdiga problem som fortfarande är oförklarliga för mig. Jag vill emellertid göra läsaren uppmärksam på att om en simulering går galet, så är det oftast själva simuleringen och dess parametrar som är felaktiga, ej programmen eller inmatade datavärden under förutsättning att dessa är rimliga.
Om inget annat anges så användes standardvärden på brusen vilka är $\sigma_{az}=10^{-3}$ m/s^2, $\sigma_q=5*10^{-5}$ s^{-1} och $\sigma_e=5$ m/s.

6.1 Testning av regulatorn

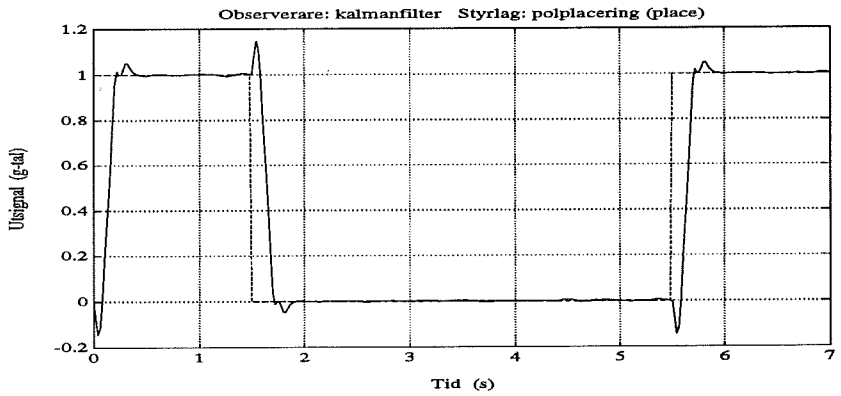
Regulatorn anses här innefatta både observerare och styrlag. Det enda väsentliga i denna sektion är att visa hur väl utsignalen a_z följer börvärdet a_{zref}.
Sju sekunder simuleras vilket medför en väntetid på c a 50 sekunder då samplingstidlängden h = 0.1 s. Dock blir beräkningarna inom den diskreta regulatorn tillräckligt snabba, d v s realtidsberäkningar.
Regulator och observerare baseras på en exakt linjär modell enligt ekv (4.3). Den "arbetspunkt", d v s missilhastighet och flyghöjd, som valts är 1.5 mach och 3 km. Denna motsvarar exakt en (v,h)-punkt för vilken en modell beräknats. Dessa värden ligger ungefär i mitten av missilbanan.

Steg 1: Bestämning av observerare och styrlagar

Det finns två slags observerare, nämligen
- polplacering enligt ekv (4.7)
- kalmanfilter vilket ges av ekv(4.8) och (4.9).

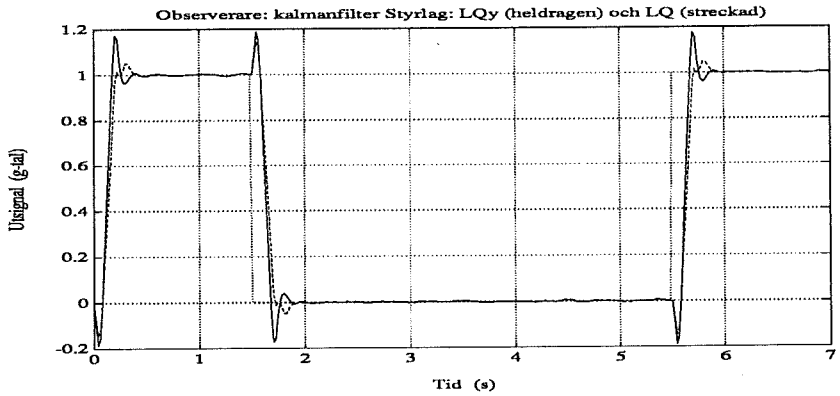
Fyra olika styrlagar kan väljas vilka är
- LQy-reglering enligt kriteriet vilket ges av ekv (4.21)
- LQ-reglering enligt kriteriet vilket ges av ekv (4.19)
- polplacering enligt Ackermanns formel ekv (4.13) och (4.14)
- en annan variant av polplacering som finns i Matlab (se *place*)

Plotten visar en typisk kurva för utsignalen (heldragen) och börvärdet (streckat) där kalmanfiltret och polplacering i styrlagen tillämpas.



Figur 1

Överslängarna beror på att robotsystemet inte är minimumfas. Tyvärr kan inte nollställena till det öppna systemet påverkas med enbart återkoppling.
De två observerarna (kalmanfilter och polplacering) ger identiska resultat. Emellertid kan olika val av styrlagar ge olika resultat. I figuren nedan jämföres LQy-styrlagen (heldragen) och LQ-styrlagen (streckad).



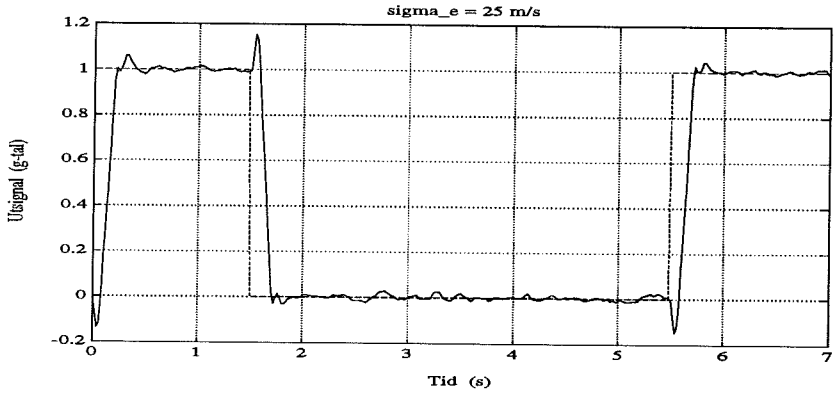
Figur 2

Av figurerna att döma är styrlagen enligt polplaceringsprincipen bättre än de linjärkvadratiska styrlagarna vilket bekräftas av nästan alla simuleringar i detta kapitel.
Det visar sig också att balansering och reducering inte förbättrar eller försämrar simuleringsresultaten. Likadant är det med δ-operatorn som vid en jämförelse med q-operatorn ger identiska resultat i alla avseenden utom då det förekommer en modellavvikelse.

Om inget annat anges så används alltså i fortsättningen ett kalmanfilter och en styrlag enligt polplaceringsprincipen. Emellertid är LQy-styrlagen mera robust mot stora process- och mätbrus om ingen hänsyn tas till överslängarna som orsakas av att systemet inte är minimumfas.

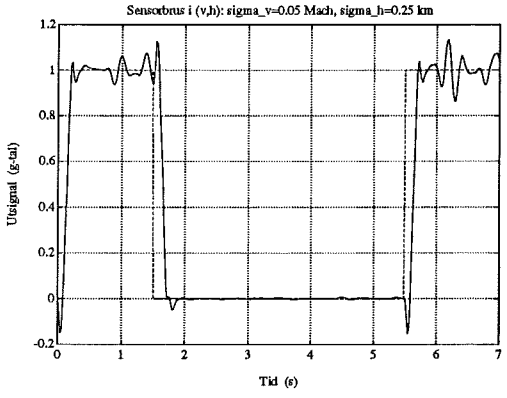
Steg 2: Brusens inverkan på simuleringsresultaten

Nästa figur visar resultaten då processbrusets styrka ökas till σ_e=25 m/s.

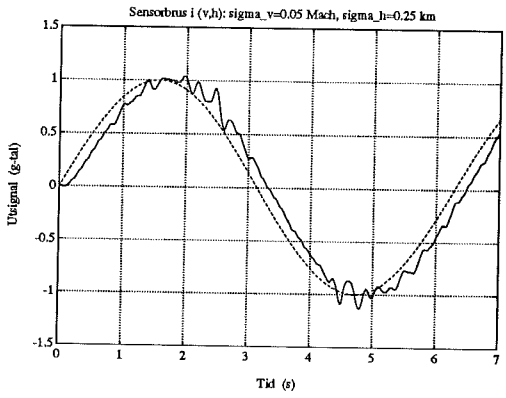


Figur 3

För små processbrus dämpas störningarna ut helt. För större processbrus pendlar utsignalen runt börvärdet, dock utan någon biasavvikelse. Liknande resultat fås då mätbrusets styrka varieras, dock med den skillnaden att regulatorn tål avsevärt kraftigare mätbrus vilket kan tiodubblas. Robustheten m a p brus i mätningen av "arbetspunkten" (v,h) är nöjaktig vilket följande figurer visar. Detta brus ökar risken att välja "fel" modell. Figurerna visar vad som händer då mätbrusens styrka är σ_v=0.05 mach och σ_h=0.25 km. Figuren till vänster har ett rektangelformat börvärde medan den till höger har börvärdet som en kontinuerlig sinuskurva.



Figur 4



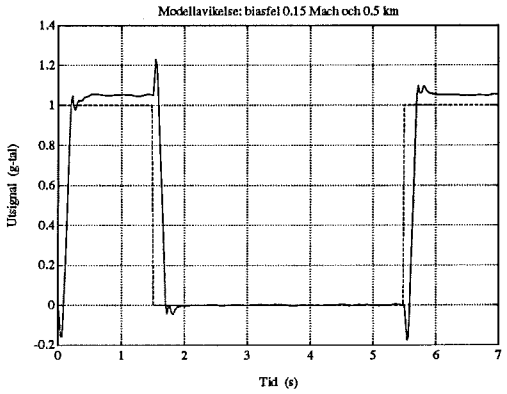
Figur 5

Ju större brusen är desto större blir överslängarna. Därför bör den övre gränsen för mätfelen av (v,h) vara σ_v=0.07 mach och σ_h=0.3-0.4 km.

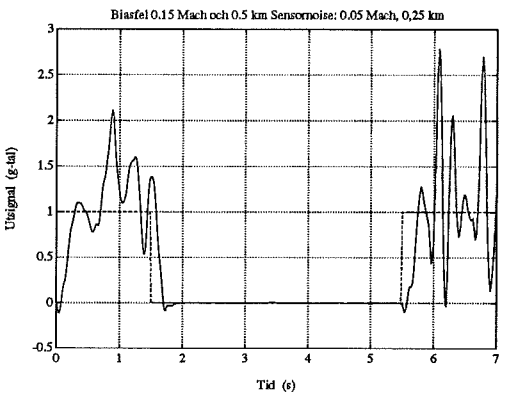
Med tanke på online-processidentifieringen (sektion 5.3) överlagrades diskretiserat vitt brus på insignalen a_{zref}, d v s börvärdet. Resultatet blev kraftiga oscillationer (ca 25 % av börvärdet) hos utsignalen, men de dämpas ut på 1-2 s vilket ändå ger en viss möjlighet att tillämpa online-identifiering.

Steg 3: Modellavvikelsens tillåtna storlek

Om det existerar en modellavvikelse på 0.15 Mach och 0.5 km börjar biasfel av utsignalen uppträda då börvärdet är skilt från noll, vilket nedanstående figur till vänster visar. Den till höger har också mätbrusen av "arbetspunkten" (v,h) där σ_v=0.05 mach och σ_h=0.25 km som förut.



Figur 6



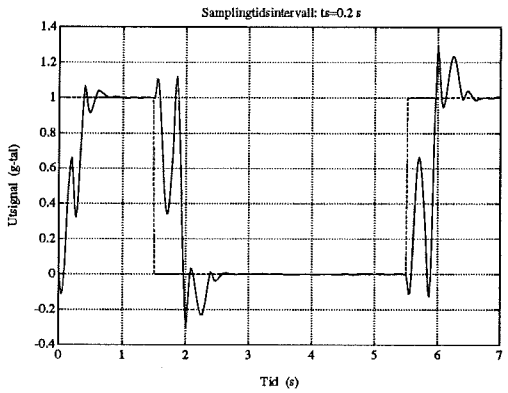
Figur 7

Ju fler modeller som finns i modellbiblioteket detso bättre blir regleringen, i synnerhet då det finns stora mätbrus i systemet.

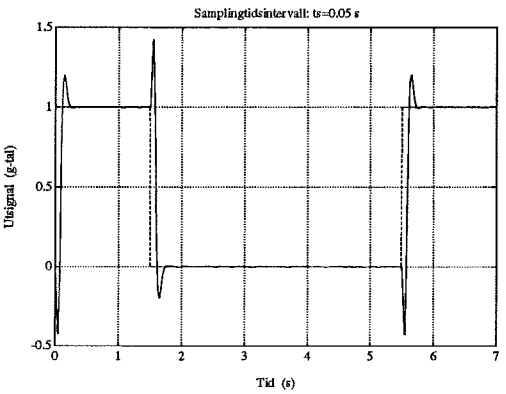
Emellertid blir biasfelet för δ-operatorn 20 % större än för q-operatorn vilket försämrar resultaten något jämfört med q-operatorn.

Steg 4: Samplingstidslängdens storlek

Angående samplingstidslängden h så får den inte vara större än maximalt 0.15 sekunder annars blir systemet instabilt. Försök har visat att avsevärt mindre h, t ex 0.05 s, ger större överslängar och dessutom blir simuleringstiderna ganska långa. Figuren till vänster har h=0.2 sekunder och den till höger h=0.05 sekunder.



Figur 8



Figur 9

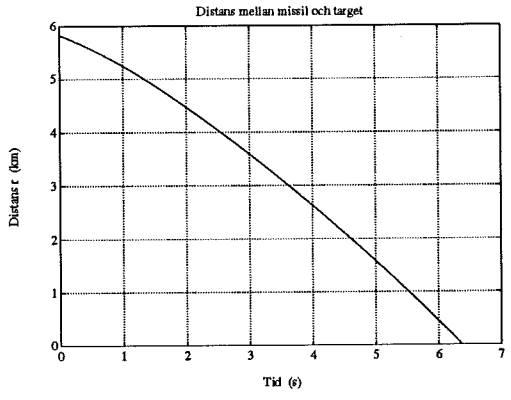
6.2 Testning av kinematiken

I denna sektion är vi intresserade av hur avståndet r minskar med tiden, variationerna av missilhastigheten v_M och anfallsvinkeln α, samt banorna för missilen och målobjektet. Det finns tre begränsningar, nämligen

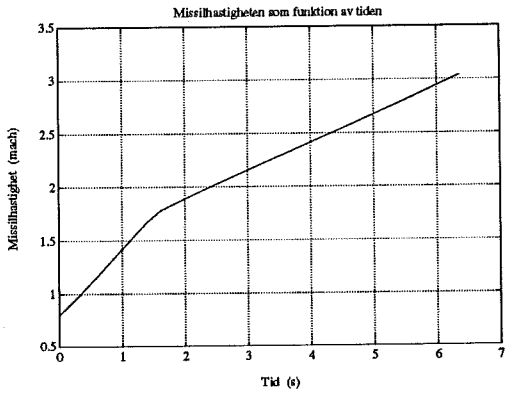
- anfallsvinkeln α bör inte vara större än 15 grader
- accelerationen a_z får inte vara större än 40 g (1 g = 9.8 m/s^2)
- bomavståndet, d v s det minsta värdet på r, får inte vara större än 5 meter

Denna sektion inledes med plottningar av ovannämnda variabler då målmodellen enligt sektion 3.2 används. Kalmanfilter och polplaceringsstyrlag tillämpas om inget annat anges. Ett modellbibliotek med 190 modeller och konstant segmentation med intervallen 0.2 mach och 0.5 km används. Detta modellbibliotek är inte optimalt men syftet är att underlätta jämförelserna genom att tvinga fram små oscillationer.
I denna sektion tillkommer det problemet att systemet är tidsvariabelt med olinjära matriser Φ och C, och att det alltid förekommer modellavikelser.

Figuren till vänster visar avståndet r vilket minskar olinjärt med tiden. Den till höger visar missilhastighetens variation med tiden. Om v_M ökar linjärt med tiden så länge dragkraften F är konstant, så betyder detta att regleringen är utmärkt, eftersom missilen då inte utsättes för onödiga omkastningar i luften. Därigenom minskas de onödiga energiförlusterna. Knycken i kurvan beror på att motorns dragkraft plötsligt minskat.



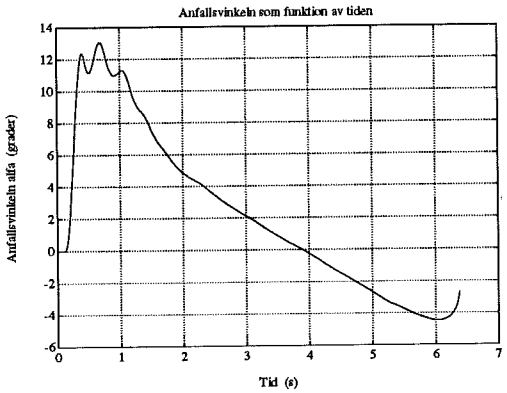
Figur 10



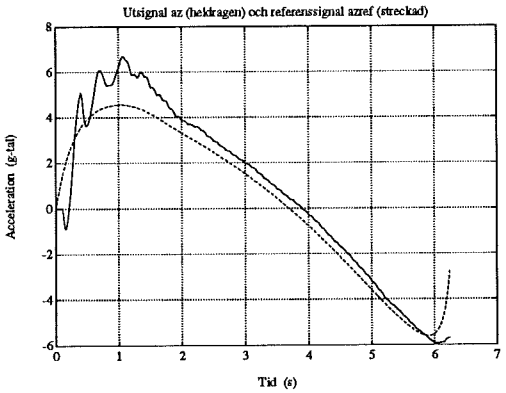
Figur 11

Figur 12 till vänster på nästa sida visar en typisk kurva för anfallsvinkeln α. Det stora utslaget i början beror på att det tar tid att ställa in missilen. Figur 13 till höger visar accelerationen a_z (heldragen) och börvärdet a_{zref} (streckat). Toppen i början av simuleringen beror på missilens tröghet.

Senare följer utsignalen a_z börvärdet a_{zref} ganska bra och inga oscillationer förekommer. Under de sista tiondels sekunderna stiger accelerationen plötsligt. Denna plötsliga ändring har inte så stor betydelse eftersom missilen förstörs i nästan samma ögonblick och att den tål minst 40 g.

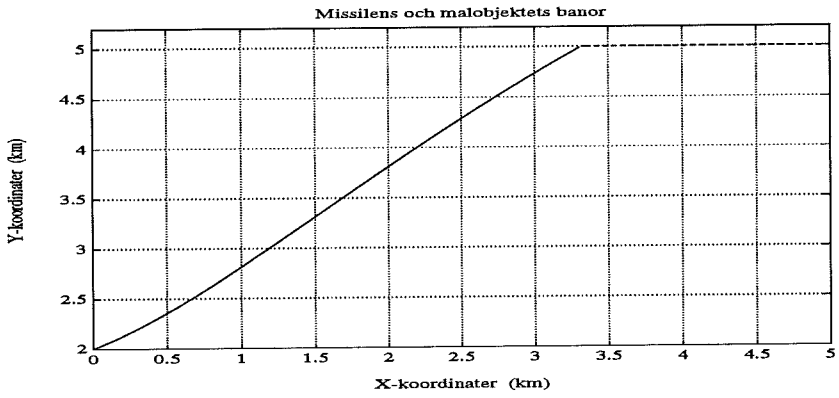


Figur 12



Figur 13

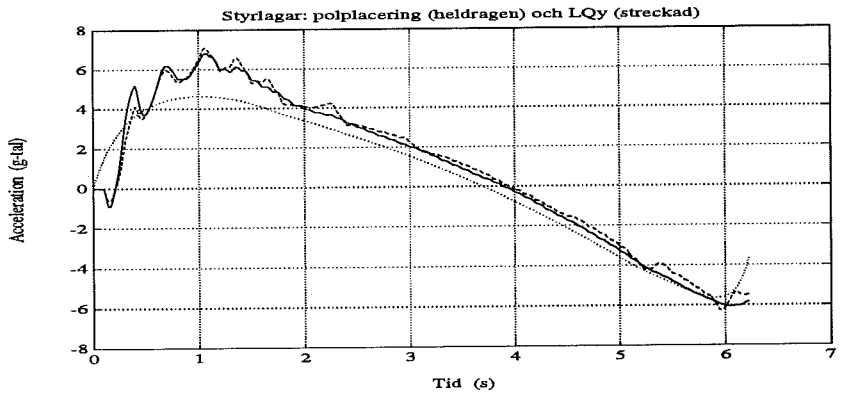
Plotten nedan visar missilens bana (heldragen) och målobjektets bana (streckad). Missilen visar en rät linje rakt mot målet vilket visar att regleringen gått ganska bra.



Figur 14

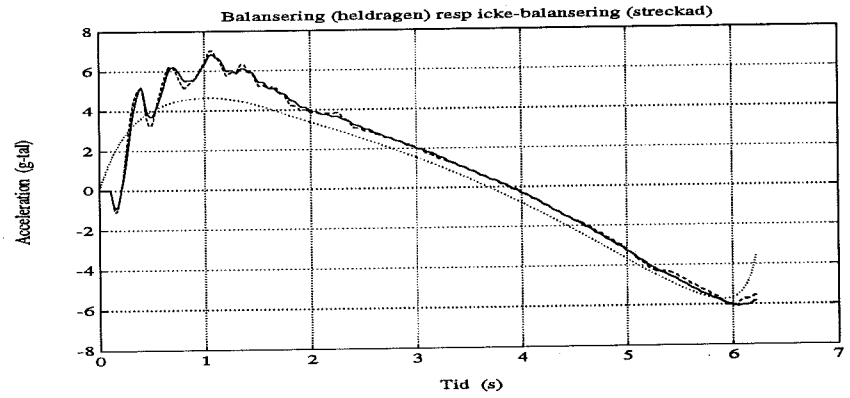
Steg 1: Regulator

I följande figur på nästa sida jämföres LQy-styrlagens (streckat) och polplaceringsstyrlagens (heldraget) styrprestanda genom att visa utsignalen a_z i de två fallen. Börvärdet a_{zref} är den prickade kurvan. Kalmanfilter tillämpas. Tydligen är polplaceringsstyrlagen bättre än LQy-styrlagen. Däremot är LQ-styrlagen nästan lika bra som polplaceringsstyrlagen. Polplaceringsobserveraren är sämre än kalmanfiltret, speciellt när brusen är kraftiga.



Figur 15

Balansering av modellerna ger en något bättre reglering. I figuren nedtill jämföres balansering (heldragen) med icke-balansering (streckad).

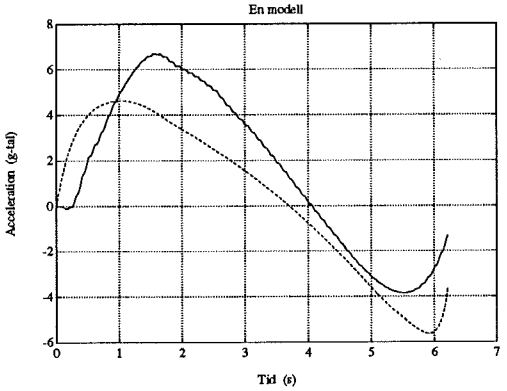


Figur 16

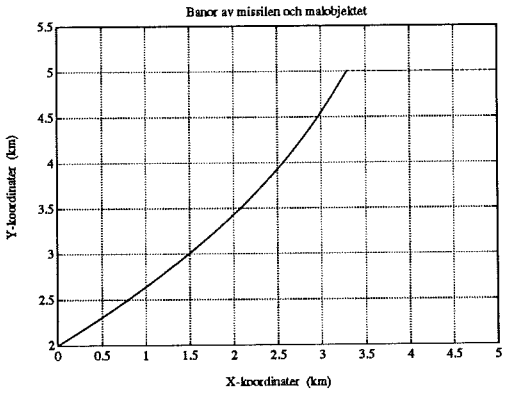
Tyvärr ger δ-operatorn en någon sämre reglering än q-operatorn i alla avseenden. Orsaken till denna försämring torde ligga i att δ-operatorn är känsligare för modellavvikelser än q-operatorn eftersom simuleringarna i sektion 6.1 gav cirka 20 % större biasfel för δ-operatorn än q-operatorn.

Steg 2: En eller flera modeller

Regulatorkonstruktionen underlättas väsentligt om en och samma robotmodell kan användas under hela robotbanan. Detta är möjligt om man väljer den modell som motsvarar (v,h) då missilen är nära målet. Regleringen blir emellertid sämre än med ett fullständigt modellbibliotek, vilket plottarna på nästa sida visar. Till vänster visas a_z (heldragen) och a_{zref} (streckad). Detta diagram bör jämföras med figur 13 och figur 20, där alltså modellbibliotek används. Till höger visas missil- och målbanor. Målbanan är här mera krökt än i figuren på föregående sida, vilket beror på sämre reglering.

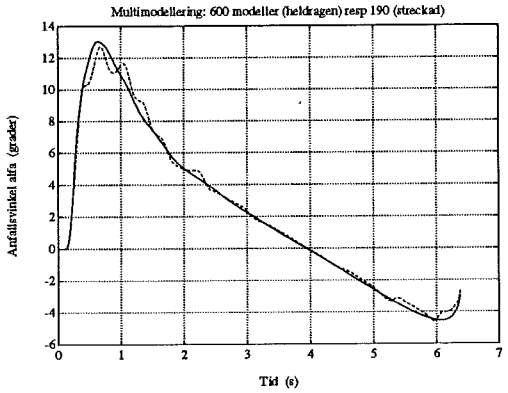


Figur 17

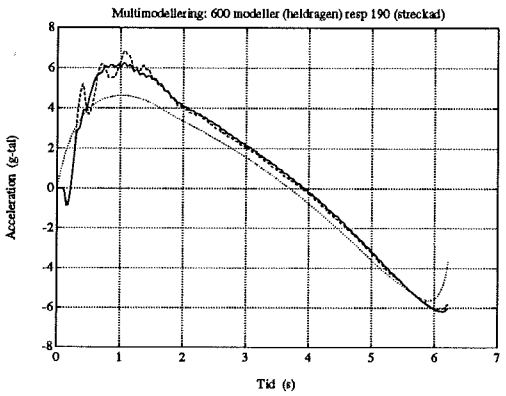


Figur 18

Övriga enmodell-val misslyckas helt. Följande figurer jämför ett modell-bibliotek med 600 modeller (heldragen) med ett mindre som har 190 modeller (streckat). Segmentationen är variabel. Figur 19 till vänster visar anfallsvinkeln α och den till höger utsignalerna a_z där börvärdet a_{zref} är den prickade kurvan.

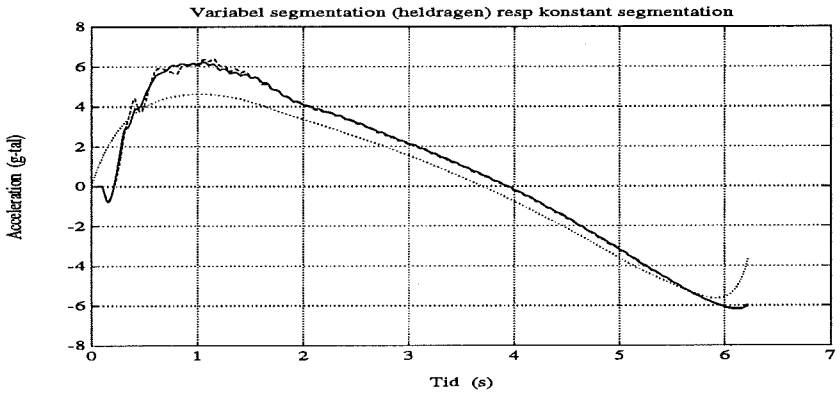


Figur 19



Figur 20

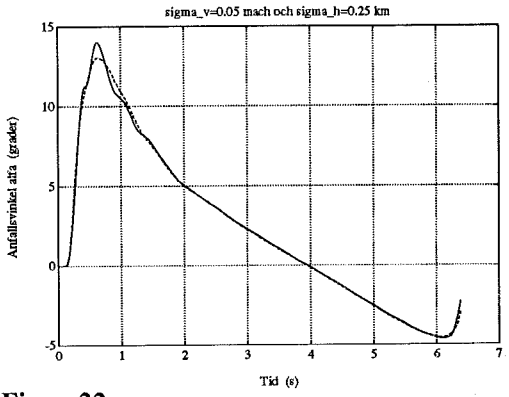
Variabel segmentation är något bättre än konstant segmentation vilket följande figur visar.



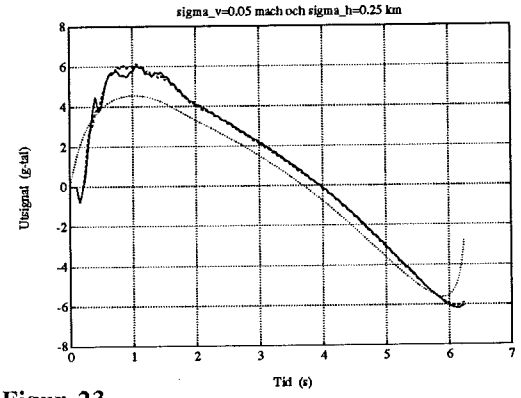
Figur 21

Steg 3: Brusens inverkan

Tack vare den andra styrlagen i kinematikloopen , d v s målsökarfunktionen, kan brusens styrka ökas avsevärt och ändå fås lyckade resultat (=uppfyller de tre begränsningarna på sid 52). I figur 22 visas för anfallsvinkeln α mätbrusens inverkan där σ_v=0.05 mach och σ_h=0.25 km (heldragen). Den streckade kurvan innehåller ingen mätbrus. I figur 23 till höger visas mätbrusens inverkan för utsignalen a_z.



Figur 22



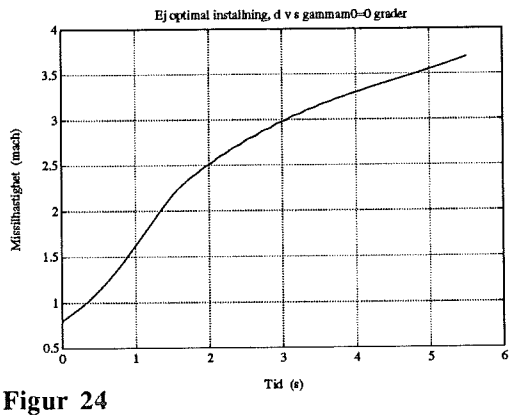
Figur 23

Steg 4: Samplingstidslängdens inverkan

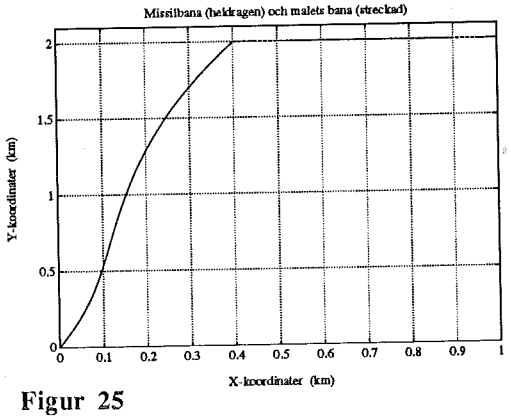
Olika h ger samma inverkan på styregenskaperna som i föregående sektion. Alltså bör h=0.1 s. Större h orsakar enorma oscillationer inne i regualtorn vilket gör att missilen blir instabil och tar fel kurs. Mindre h ger något bättre simuleringsresultat, dock tar det mycket längre tid.

Steg 5: Icke-optimala inställningar av missilen

Hittills har vi förutsatt att inställningen av missilen mot målobjektet varit optimal, d v s ε_0=0° och $\gamma_{M0}=\phi_0$. Simuleringar har emellertid visat att ε_0 måste vara mindre än 1° för att missilen skall träffa målet. Däremot går det bra att skjuta ut missilen längs flygplanets axelriktning, d v s γ_{M0}=0° vilket visas i figurerna nedan. Men då utsättes missilen för kraftiga oscillationer på cirka 4 g, d v s den skakar. Figur 24 till vänster visar v_M som funktion av tiden och figur 25 målbanorna för missil resp mål.



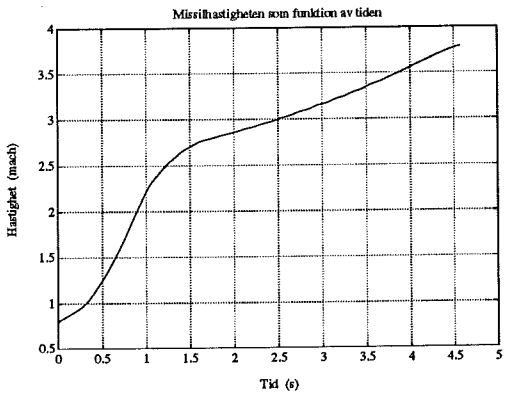
Figur 24



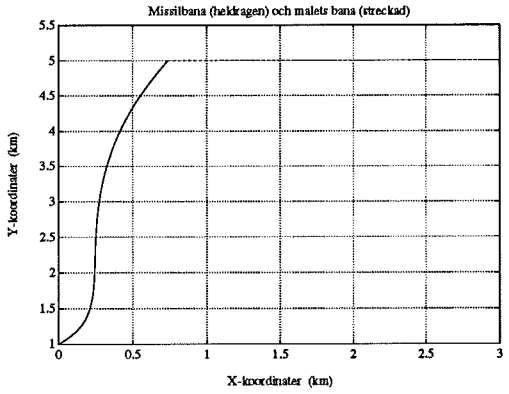
Figur 25

Steg 6: Andra målfall

Man kan ställa tuffare krav på missilen genom att öka målets hastighet till 1.5 mach och avstånd i höjdled till 4 km. Följande figur till vänster visar missilhastighetens (v_M) varation med tiden, och den till höger visar hur missilen stiger kraftigt för att sedan plana ut mot slutet.



Figur 26



Figur 27

Missilen klarar också av att jaga ifatt målet så länge målets fart inte är större än missilens fart. Även när motorn dött ut efter 6.5 s kan missilen fortfarande träffa målet efter ytterligare 5 s.

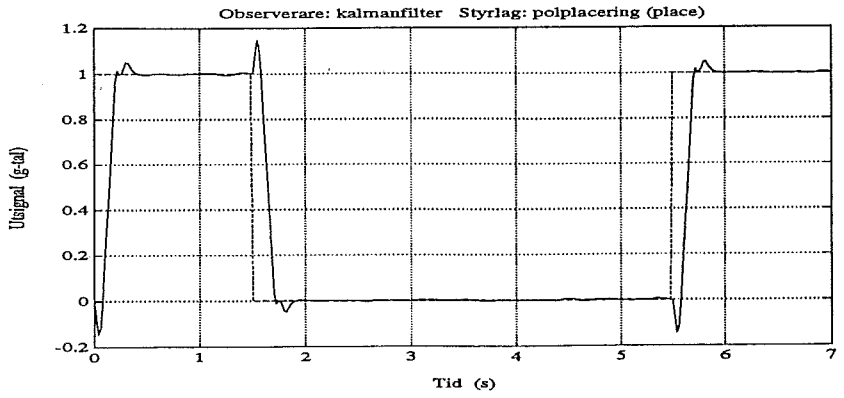
Sammanfattningsvis bör den optimala designen vara enligt följande lista:

* optimal förinställning mot målet
* stora modellbibliotek eller små modellavvikelser
* variabel segmentering
* samplingstidslängden h=0.1 s
* q-operator inklusive balansering av modellerna
* observeraren är ett kalmanfilter
* styrlagen är av polplaceringstyp med lämpliga poler

Vid konstruktion av styrdatorer måste man veta det verkliga beräknings-arbetet inne i den digitala regulatorn. Resultaten kan sammanfattas:

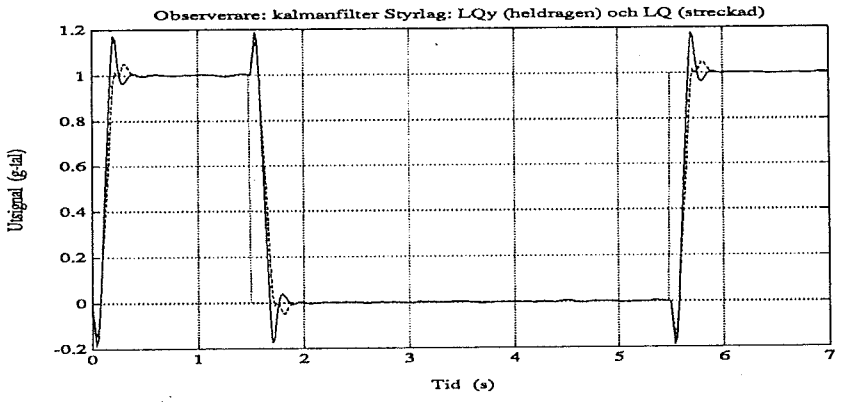
* ju fler modeller desto längre tid tar det. Antalet modeller i modell-biblioteket bör inte vara mer än 150-200 st. I simuleringen används bara 25-35 modeller utmed hela missilbanan.
* balansering av modellerna tar extra tid
* vid tidsbrist kan samplingstidlängden h väljas till 0.15 sekunder
* kalmanfiltret är snabbare än polplaceringsobserveraren
* de linjärkvadratiska styrlagarna är ännu snabbare än polplacerings-styrlagarna

Plotten visar en typisk kurva för utsignalen (heldragen) och börvärdet (streckat) där kalmanfiltret och polplacering i styrlagen tillämpas.



Figur 1

Överslängarna beror på att robotsystemet inte är minimumfas. Tyvärr kan inte nollställena till det öppna systemet påverkas med enbart återkoppling.
De två observerarna (kalmanfilter och polplacering) ger identiska resultat. Emellertid kan olika val av styrlagar ge olika resultat. I figuren nedan jämföres LQy-styrlagen (heldragen) och LQ-styrlagen (streckad).



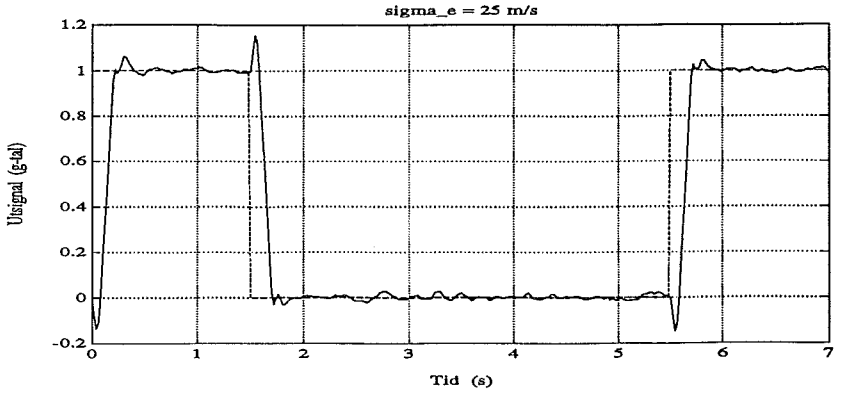
Figur 2

Av figurerna att döma är styrlagen enligt polplaceringsprincipen bättre än de linjärkvadratiska styrlagarna vilket bekräftas av nästan alla simuleringar i detta kapitel.
Det visar sig också att balansering och reducering inte förbättrar eller försämrar simuleringsresultaten. Likadant är det med δ-operatorn som vid en jämförelse med q-operatorn ger identiska resultat i alla avseenden utom då det förekommer en modellavvikelse.

Om inget annat anges så används alltså i fortsättningen ett kalmanfilter och en styrlag enligt polplaceringsprincipen. Emellertid är LQy-styrlagen mera robust mot stora process- och mätbrus om ingen hänsyn tas till överslängarna som orsakas av att systemet inte är minimumfas.

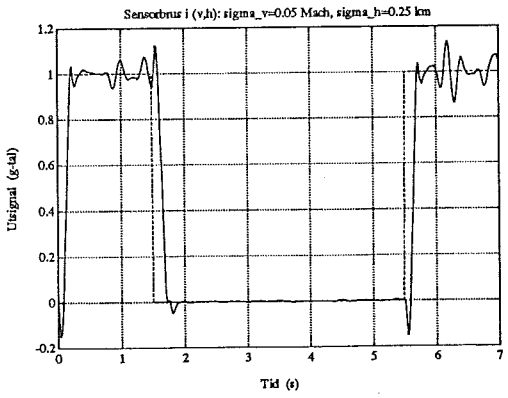
Steg 2: Brusens inverkan på simuleringsresultaten

Nästa figur visar resultaten då processbrusets styrka ökas till σ_e=25 m/s.

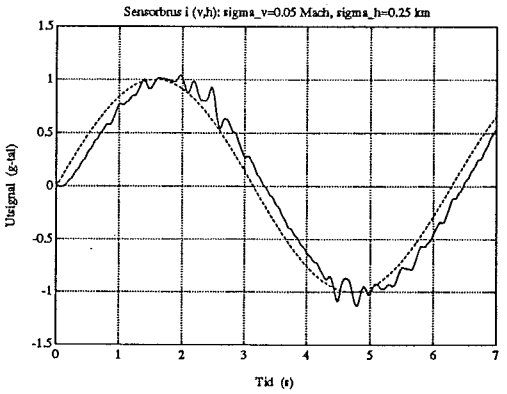


Figur 3

För små processbrus dämpas störningarna ut helt. För större processbrus pendlar utsignalen runt börvärdet, dock utan någon biasavvikelse. Liknande resultat fås då mätbrusets styrka varieras, dock med den skillnaden att regulatorn tål avsevärt kraftigare mätbrus vilket kan tiodubblas. Robustheten m a p brus i mätningen av "arbetspunkten" (v,h) är nöjaktig vilket följande figurer visar. Detta brus ökar risken att välja "fel" modell. Figurerna visar vad som händer då mätbrusens styrka är σ_v=0.05 mach och σ_h=0.25 km. Figuren till vänster har ett rektangelformat börvärde medan den till höger har börvärdet som en kontinuerlig sinuskurva.



Figur 4



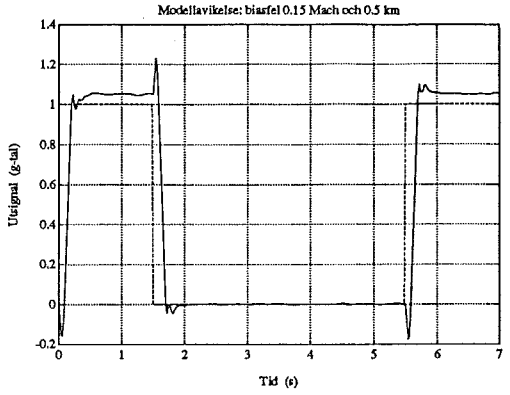
Figur 5

Ju större brusen är desto större blir överslängarna. Därför bör den övre gränsen för mätfelen av (v,h) vara σ_v=0.07 mach och σ_h=0.3-0.4 km.

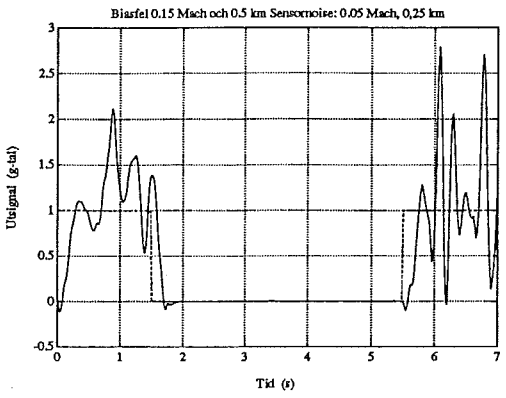
Med tanke på online-processidentifieringen (sektion 5.3) överlagrades diskretiserat vitt brus på insignalen a_{zref}, d v s börvärdet. Resultatet blev kraftiga oscillationer (ca 25 % av börvärdet) hos utsignalen, men de dämpas ut på 1-2 s vilket ändå ger en viss möjlighet att tillämpa online-identifiering.

Steg 3: Modellavvikelsens tillåtna storlek

Om det existerar en modellavvikelse på 0.15 Mach och 0.5 km börjar biasfel av utsignalen uppträda då börvärdet är skilt från noll, vilket nedanstående figur till vänster visar. Den till höger har också mätbrusen av "arbetspunkten" (v,h) där σ_v=0.05 mach och σ_h=0.25 km som förut.



Figur 6



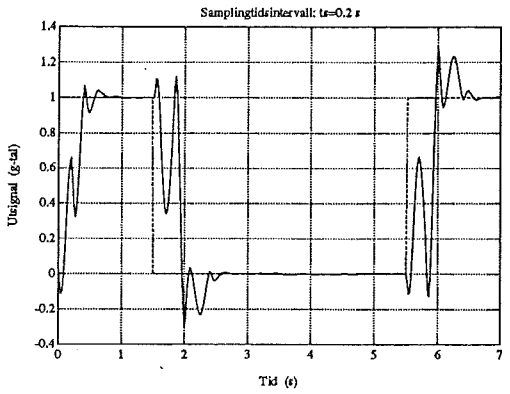
Figur 7

Ju fler modeller som finns i modellbiblioteket detso bättre blir regleringen, i synnerhet då det finns stora mätbrus i systemet.

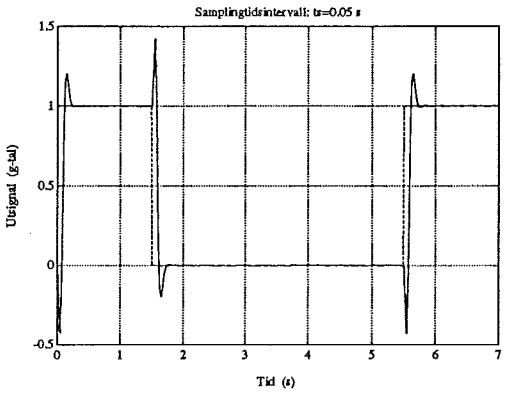
Emellertid blir biasfelet för δ-operatorn 20 % större än för q-operatorn vilket försämrar resultaten något jämfört med q-operatorn.

Steg 4: Samplingstidslängdens storlek

Angående samplingstidslängden h så får den inte vara större än maximalt 0.15 sekunder annars blir systemet instabilt. Försök har visat att avsevärt mindre h, t ex 0.05 s, ger större överslängar och dessutom blir simuleringstiderna ganska långa. Figuren till vänster har h=0.2 sekunder och den till höger h=0.05 sekunder.



Figur 8



Figur 9

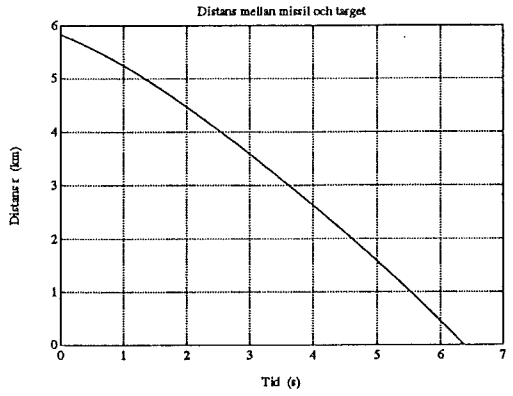
6.2 Testning av kinematiken

I denna sektion är vi intresserade av hur avståndet r minskar med tiden, variationerna av missilhastigheten v_M och anfallsvinkeln α, samt banorna för missilen och målobjektet. Det finns tre begränsningar, nämligen

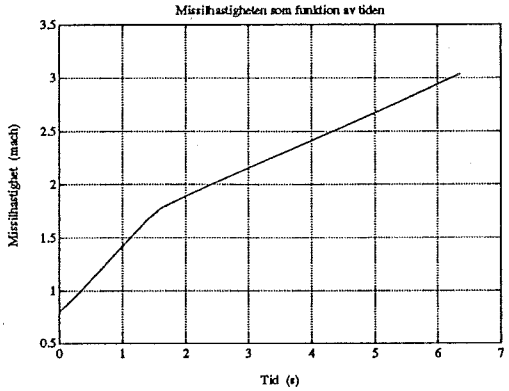
- anfallsvinkeln α bör inte vara större än 15 grader
- accelerationen a_z får inte vara större än 40 g (1 g = 9.8 m/s^2)
- bomavståndet, d v s det minsta värdet på r, får inte vara större än 5 meter

Denna sektion inledes med plottningar av ovannämnda variabler då målmodellen enligt sektion 3.2 används. Kalmanfilter och polplaceringsstyrlag tillämpas om inget annat anges. Ett modellbibliotek med 190 modeller och konstant segmentation med intervallen 0.2 mach och 0.5 km används. Detta modellbibliotek är inte optimalt men syftet är att underlätta jämförelserna genom att tvinga fram små oscillationer.
I denna sektion tillkommer det problemet att systemet är tidsvariabelt med olinjära matriser Φ och C, och att det alltid förekommer modellavikelser.

Figuren till vänster visar avståndet r vilket minskar olinjärt med tiden. Den till höger visar missilhastighetens variation med tiden. Om v_M ökar linjärt med tiden så länge dragkraften F är konstant, så betyder detta att regleringen är utmärkt, eftersom missilen då inte utsättes för onödiga omkastningar i luften. Därigenom minskas de onödiga energiförlusterna. Knycken i kurvan beror på att motorns dragkraft plötsligt minskat.



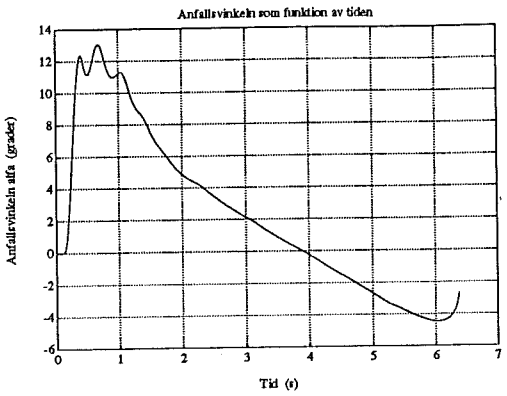
Figur 10



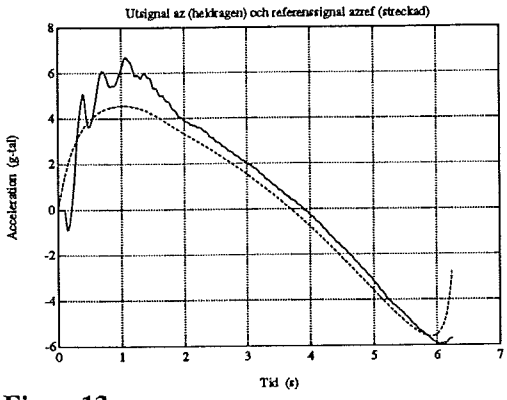
Figur 11

Figur 12 till vänster på nästa sida visar en typisk kurva för anfallsvinkeln α. Det stora utslaget i början beror på att det tar tid att ställa in missilen. Figur 13 till höger visar accelerationen a_z (heldragen) och börvärdet a_{zref} (streckat). Toppen i början av simuleringen beror på missilens tröghet.

Senare följer utsignalen a_z börvärdet a_{zref} ganska bra och inga oscillationer förekommer. Under de sista tiondels sekunderna stiger accelerationen plötsligt. Denna plötsliga ändring har inte så stor betydelse eftersom missilen förstörs i nästan samma ögonblick och att den tål minst 40 g.

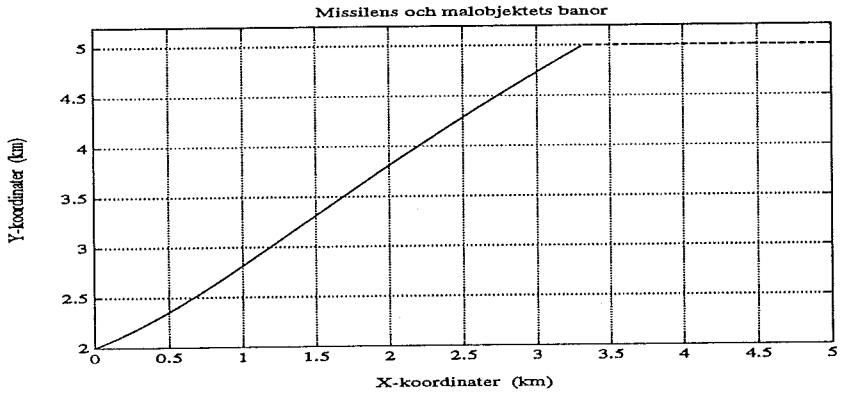


Figur 12



Figur 13

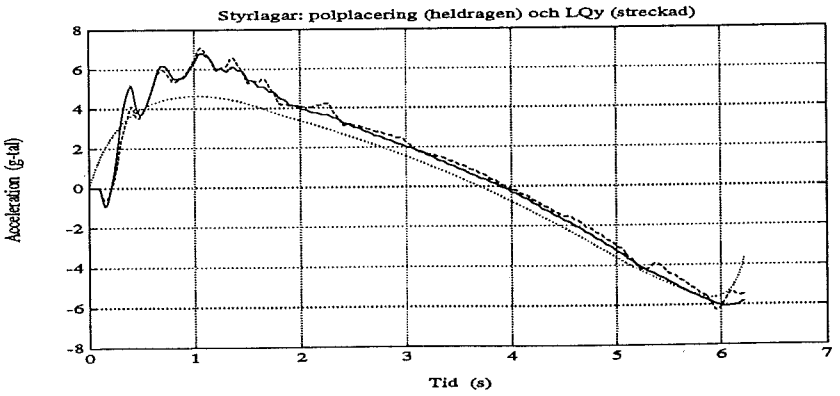
Plotten nedan visar missilens bana (heldragen) och målobjektets bana (streckad). Missilen visar en rät linje rakt mot målet vilket visar att regleringen gått ganska bra.



Figur 14

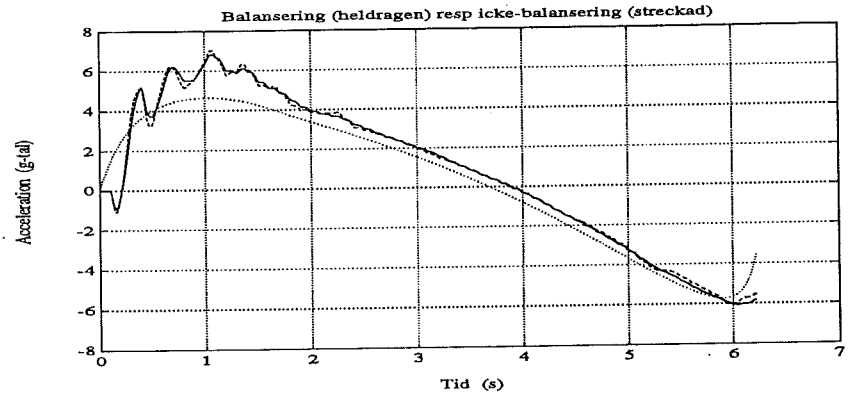
Steg 1: Regulator

I följande figur på nästa sida jämföres LQy-styrlagens (streckat) och polplaceringsstyrlagens (heldraget) styrprestanda genom att visa utsignalen a_z i de två fallen. Börvärdet a_{zref} är den prickade kurvan. Kalmanfilter tillämpas. Tydligen är polplaceringsstyrlagen bättre än LQy-styrlagen. Däremot är LQ-styrlagen nästan lika bra som polplaceringsstyrlagen. Polplaceringsobserveraren är sämre än kalmanfiltret, speciellt när brusen är kraftiga.



Figur 15

Balansering av modellerna ger en något bättre reglering. I figuren nedtill jämföres balansering (heldragen) med icke-balansering (streckad).

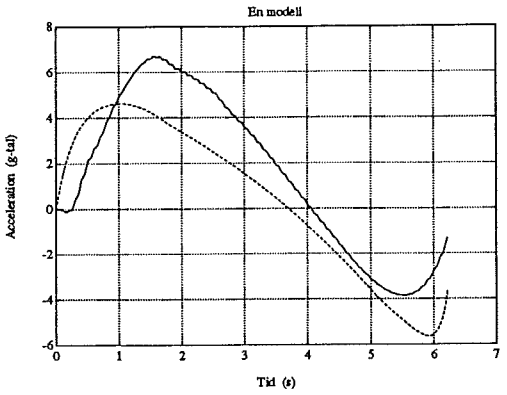


Figur 16

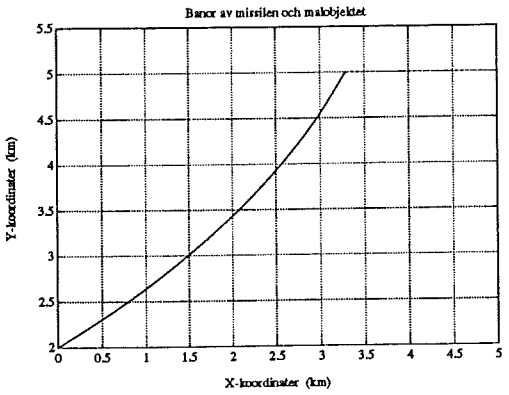
Tyvärr ger δ-operatorn en någon sämre reglering än q-operatorn i alla avseenden. Orsaken till denna försämring torde ligga i att δ-operatorn är känsligare för modellavvikelser än q-operatorn eftersom simuleringarna i sektion 6.1 gav cirka 20 % större biasfel för δ-operatorn än q-operatorn.

Steg 2: En eller flera modeller

Regulatorkonstruktionen underlättas väsentligt om en och samma robot-modell kan användas under hela robotbanan. Detta är möjligt om man väljer den modell som motsvarar (v,h) då missilen är nära målet. Regleringen blir emellertid sämre än med ett fullständigt modellbibliotek, vilket plottarna på nästa sida visar. Till vänster visas a_z (heldragen) och a_{zref} (streckad). Detta diagram bör jämföras med figur 13 och figur 20, där alltså modellbibliotek används. Till höger visas missil- och målbanor. Målbanan är här mera krökt än i figuren på föregående sida, vilket beror på sämre reglering.

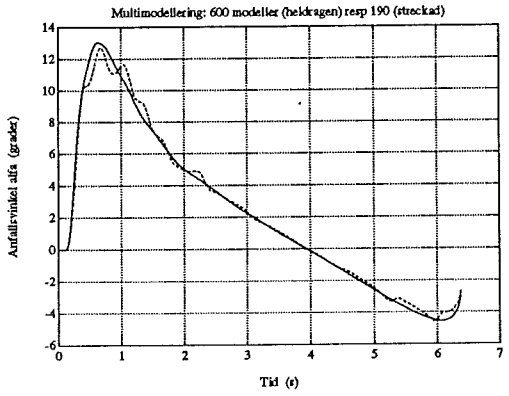


Figur 17

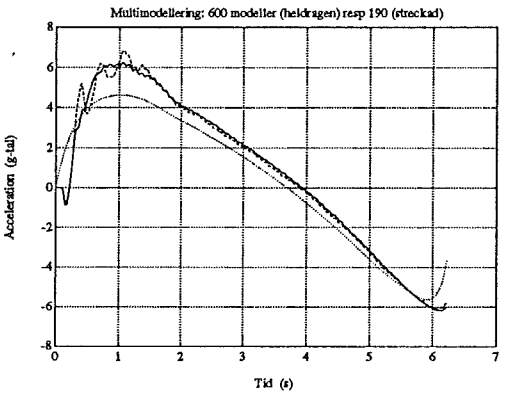


Figur 18

Övriga enmodell-val misslyckas helt. Följande figurer jämför ett modell-bibliotek med 600 modeller (heldragen) med ett mindre som har 190 modeller (streckat). Segmentationen är variabel. Figur 19 till vänster visar anfallsvinkeln α och den till höger utsignalerna a_z där börvärdet a_{zref} är den prickade kurvan.

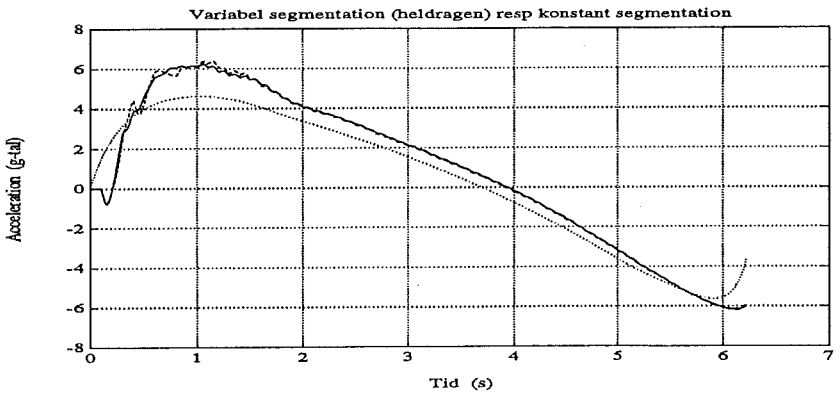


Figur 19



Figur 20

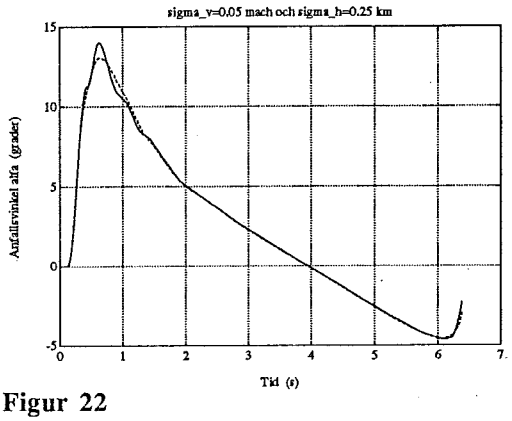
Variabel segmentation är något bättre än konstant segmentation vilket följande figur visar.



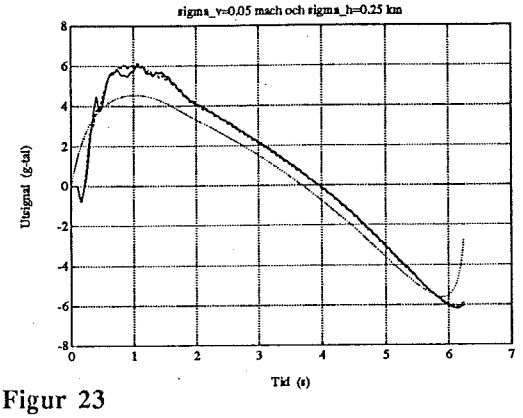
Figur 21

Steg 3: Brusens inverkan

Tack vare den andra styrlagen i kinematikloopen , d v s målsökarfunktionen, kan brusens styrka ökas avsevärt och ändå fås lyckade resultat (=uppfyller de tre begränsningarna på sid 52). I figur 22 visas för anfallsvinkeln α mätbrusens inverkan där σ_v=0.05 mach och σ_h=0.25 km (heldragen). Den streckade kurvan innehåller ingen mätbrus. I figur 23 till höger visas mätbrusens inverkan för utsignalen a_z.



Figur 22



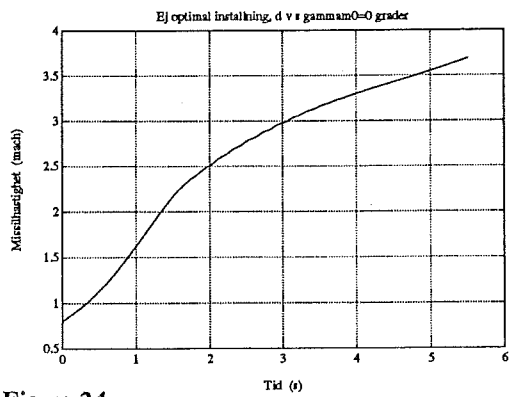
Figur 23

Steg 4: Samplingstidslängdens inverkan

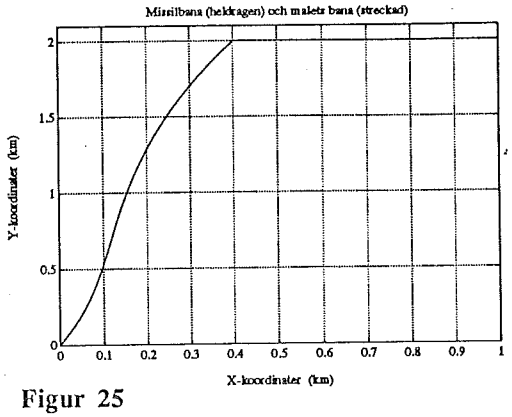
Olika h ger samma inverkan på styregenskaperna som i föregående sektion. Alltså bör h=0.1 s. Större h orsakar enorma oscillationer inne i regualtorn vilket gör att missilen blir instabil och tar fel kurs. Mindre h ger något bättre simuleringsresultat, dock tar det mycket längre tid.

Steg 5: Icke-optimala inställningar av missilen

Hittills har vi förutsatt att inställningen av missilen mot målobjektet varit optimal, d v s ε_0=0° och $\gamma_{M0}=\phi_0$. Simuleringar har emellertid visat att ε_0 måste vara mindre än 1° för att missilen skall träffa målet. Däremot går det bra att skjuta ut missilen längs flygplanets axelriktning, d v s γ_{M0}=0° vilket visas i figurerna nedan. Men då utsättes missilen för kraftiga oscillationer på cirka 4 g, d v s den skakar. Figur 24 till vänster visar v_M som funktion av tiden och figur 25 målbanorna för missil resp mål.



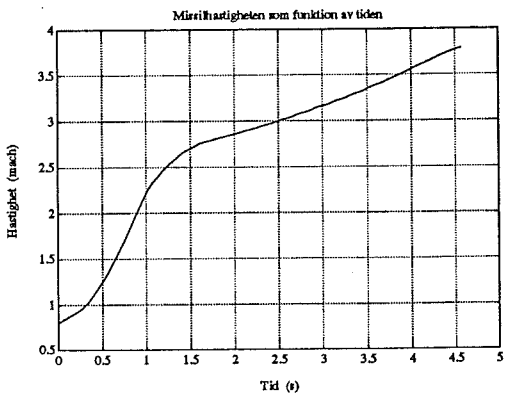
Figur 24



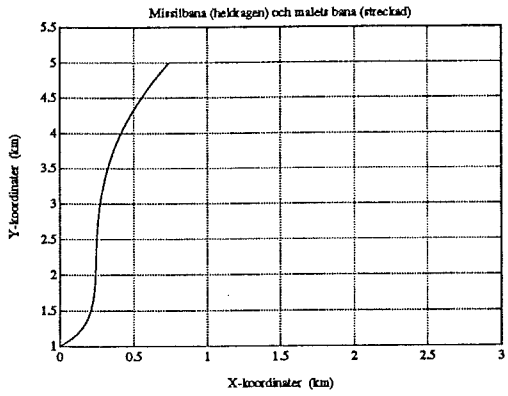
Figur 25

Steg 6: Andra målfall

Man kan ställa tuffare krav på missilen genom att öka målets hastighet till 1.5 mach och avstånd i höjdled till 4 km. Följande figur till vänster visar missilhastighetens (v_M) varation med tiden, och den till höger visar hur missilen stiger kraftigt för att sedan plana ut mot slutet.



Figur 26



Figur 27

Missilen klarar också av att jaga ifatt målet så länge målets fart inte är större än missilens fart. Även när motorn dött ut efter 6.5 s kan missilen fortfarande träffa målet efter ytterligare 5 s.

Sammanfattningsvis bör den optimala designen vara enligt följande lista:

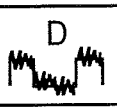
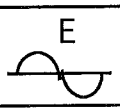
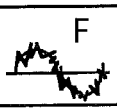
* optimal förinställning mot målet
* stora modellbibliotek eller små modellavvikelser
* variabel segmentering
* samplingstidslängden h=0.1 s
* q-operator inklusive balansering av modellerna
* observeraren är ett kalmanfilter
* styrlagen är av polplaceringstyp med lämpliga poler

Vid konstruktion av styrdatorer måste man veta det verkliga beräknings-arbetet inne i den digitala regulatorn. Resultaten kan sammanfattas:

* ju fler modeller desto längre tid tar det. Antalet modeller i modell-biblioteket bör inte vara mer än 150-200 st. I simuleringen används bara 25-35 modeller utmed hela missilbanan.
* balansering av modellerna tar extra tid
* vid tidsbrist kan samplingstidlängden h väljas till 0.15 sekunder
* kalmanfiltret är snabbare än polplaceringsobserveraren
* de linjärkvadratiska styrlagarna är ännu snabbare än polplacerings-styrlagarna

6.3 Övriga system

I början av denna sektion ska processidentifieraren jämföras för 12 olika fall. Syftet är att vilka algoritmer som är "bäst", hur viktigt R_{2d} är för estimatens noggrannhet, hur stor excitation insignalen måste ha och hur mät- och processbrusen påverkar resultaten.
"Arbetspunkten" väljes som förut, d v s v=1.5 mach och h=3 km utom för de två sista fallen i tabellen nedan. För denna arbetspunkt kan en exakt modellkännedom härledas m h a SimuLink-kommandot *dlinmod* eller räknas fram för hand så att processidentifierarna kan valideras.
Det första felkriteriet nedan i tabellen är det relativa felet för det <u>första</u> elementet i parametervektorn θ, d v s a_1. Det andra felkriteriet är "relativa" felet för <u>hela</u> parametervektorn vilket definieras $\|\theta - \hat{\theta}\| / \|\theta\|$.
Standardvärden används för mät- och processbrusen samt "arbetspunkten" (R_{2d} ingår normalt). In-signalen är vitt brus med σ =1.
Simuleringstiden är 2 s och samplingtidslängden är 0.1 s. Dessa standardvärden används genomgående i tabellen om inget annat anges.

	exkl R_{2d}	inkl R_{2d}	σ_e=1 m/s	σ_e=10 m/s	A	B
Potter	3.2% 0,36%	1.2% 0.14%	0.13% 0.022%	0.24% 0.030%	0.88% 0.096%	0.89% 0.098%
RLS	4.7% 0,55%	4.7% 0.55%	0.30% 0.032%	32% 3.7%	5.7% 0.65%	5.3% 0.62%
	C	D 	E 	F 	0.8 mach 2 km	3.2 mach 5 km
Potter	1% 0.7s	8.6% 0.98%	72% 7.9%	3.9% 0.43%	2.5% 0.79%	0.043% 0.20%
RLS	5-10% 1.6s	1.6% 0.19%	48% 24%	105% 920%	8.6% 26%	1.1% 0.23%

A och B står för de fall då både σ_{az} och σ_q fördubblas resp halveras.
C är ett mått på snabbheten, d v s hur lång tid det tar för att minska det relativa felet för a_1 till under 5%. Procentalen är de minsta realativa fel man kan få för a_1.
Fall D,E och F visar hur bra de två algoritmerna kan estimera då insignalen a_{zref} har de utseenden som visas i tabellen för resp fall. De överlagda brusen är diskretiserade med σ= 5% av amplituden för a_{zref} vilket antingen är rektangelformad eller sinusformad. Detta är viktigt för att veta om adaptivitet skall kunna tillämpas på det slutna systemet i regulatorn och kinematiken.

Som tabellen antyder är Potters algoritm bäst, speciellt för dåligt exciterande system. Faktum är att den kan användas för det slutna regulatorsystemet där insignalen av naturliga skäl inte får variera för mycket.

Om initialvärdena för $\hat{\theta}_0$ och P_0. är tillräckligt bra, d v s "arbetspunkten" avviker maximalt 0.15 mach och 0.5 km från den modell där $\hat{\theta}_0$ och P_0 hämtas ifrån så fås både bra reglering och estimatvärden på $\hat{\theta}$. Det borde gå att i vissa gynnsamma fall tillämpa adaptivitet så länge parametrarna (v,h) är konstanta så att systemet blir linjärt och tidinvariant.

Jag har försökt tillämpa processidentifiering online men det har inte gått så bra. Däremot går det ibland bra att processidentifiera tillsammans med det slutna systemet. Frågan är emellertid om systemet är tillräckligt exiterande. Dessutom försvårar de tidsvariabla parametrarna (v,h) det hela att man kan fråga sig om resultaten är tillräckligt pålitliga.

Det är inte heller bättre med processidentifieringen på δ-operator. Konvergensen av parameterestimeringen försämras avsevärt. Noggrannheten kan inte bli bättre än 5 % för något fall. Orsaken torde ligga i att regressionsvektorn ϕ har derivator av ordningen n-1 trots det stabila filtrets inverkan.
Ett sätt att komma förbi detta problem vore att filtera signalerna med ett första ordningens filteroch därefter omtransformera parametervektorn θ. Detta bör bli en lämplig fortsättning av examensarbetet. En viss vägledning ges av referens [5] i kapitel 12.

Övriga kommentarer: Efter dessa lyckade försök att identifiera godtyckliga ARX-modeller enligt ekv (5.1), försökte jag även identifiera godtyckliga ARMAX-modeller enligt ekv (6.1).

$$(6.1) \qquad A(q^{-1})y_t = q^{-d}B(q^{-1})u_t + C(q^{-1})e_t$$

Där identifieras också koefficienterna i $C(q^{-1})$-polynomen för brusen.

För detta ändamål användes en variant av RLS-metoden, nämligen ELS-metoden (Extended recursive Least Squares method) för ett SISO-system. Ordentliga försök har gjorts men trots teorin är korrekt och programmen väl kontrollerade fick förblev alla koefficienter i $C(q^{-1})$-polynomen mycket små. Orsaken är fortfarande ej klargjorda.
Däremot har ett program som tillämpar Biermans UD-faktorsering för SISO-system skrivits. Denna algoritm var nästan lika bra som Potters algoritm. Tyvärr kunde UD-faktoriserngen ej omformuleras för att också omfatta SIMO-system därför att detta är matematiskt omöjligt.

6.4 Förslag till framtida arbete

* förbättring av processidentifieraren i slutna system där excitationen är för dålig för att kunna få helt pålitliga modelldata

* estimering av $C(q^{-1})$-polynomen för att optimalt filtera utsignalerna mot brusens inverkan

* och om detta lyckas så modifiera regulatorn genom användning av polplacering baserat på en insignal-utsignal-sekvens istället för estimering av tillstånden m h a ett optimalt Kalmanfilter

* förbättring av processidentifieringen på δ-operator genom att istället införa första ordningens filter i ϕ och sedan omtransformera θ

* om möjligt, kvadratrotuppdela alla kalmanfilter d v s också ekv (4.8)-(4.9) och (4.40)-(4.41)

* använd Lyaponov-baserade adaptiva metoder istället för den adaptiva metod som beskrevs i sektion 5.3. Dessutom är det svårt att hitta andra adaptiva metoder som fungerar för SIMO-system

* använda olinjär kompensering (referens [3] och [6]), men problemet är veta om det stabila området är tillräckligt stort för missilen

Appendix A: Beteckningar

Följande beteckningar används i de flygmekaniska ekvationerna för missilen:

m	missilens massa
v	missilens hastighet (m/s eller machtal)
S	referensyta (robotkroppens genomskärningsyta)
d	referenslängd (kroppens genomskärningsdiameter)
J_0	tröghetsmoment m a p axeln tvärs planet
ζ_0	roderservons dämpning
ω_0	roderservons egenvinkelfrekvens
$\rho(h)$	luftens täthet som funktion av höjden över havsytan
l_t	avstånd mellan tyngdpunkt och tryckcentrum
l_o	avstånd mellan stjärtroder och tyngdpunkt
$C_{N\alpha}(v)$	normalkraftskoefficientens derivata m a p anfallsvinkeln α som funktion av machtalet
$C_{N\delta}(v)$	normalkraftskoefficientens derivata m a p roderutslaget δ som funktion av machtalet
$C_{m\alpha}(v)$	momentkoefficientens derivata m a p anfallsvinkeln α som funktion av machtalet
$C_{mq}(v)$	momentkoefficientes derivata m a p kroppsvinkelhastigheten q som funktion av machtalet
$C_{m\delta}(v)$	momentkoefficientens derivata m a p roderutslaget δ som funktion av machtalet

Alla enheter förutsättes vara SI-enheter utom i $C_{N\alpha}(v)$, $C_{N\alpha}(v)$ o s v där missilfarten v anges i machtal.

Appendix B: Process- och mätbrus

I detta appendix beskrives de störningar som roboten råkar ut för. Här härledes kovariansmatriserna för processbruset R_{1d} och mätbruset R_{2d}. I detta appendix antages att alla brus är vita och okorrelerade. I så fall kan störningarna i utsignalerna a_z och q modelleras som en deterministisk komponent och en stokastisk komponent

$$\begin{aligned} a_z &= a_{z,det} + \upsilon_{az} \qquad \text{där } \upsilon_{az} \in N(0,\sigma_{az}) \\ q &= q_{det} + \upsilon_q \qquad\qquad \upsilon_q \in N(0,\sigma_q) \end{aligned}$$

Oftast vet man hur stora standardavvikelserna σ_{az} och σ_q är. Då kan R_{2c}, d v s kovariansmatrisen för det kontinuerliga mätbruset, enkelt beräknas

$$\text{(B.1)} \qquad R_{2c} = E\left[e(t)e^T(t)\right] = \begin{bmatrix} \sigma_{az}^2 & 0 \\ 0 & \sigma_q^2 \end{bmatrix} \qquad \text{där } e(t) = \begin{bmatrix} \upsilon_{a_z} \\ \upsilon_q \end{bmatrix}$$

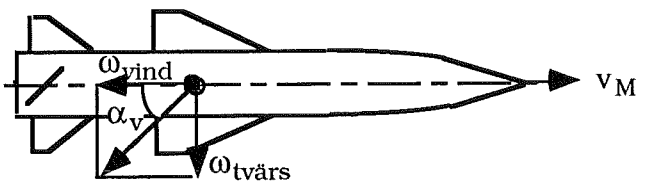
Enligt Matlab-manualen ges den diskreta varianten R_{2d} av

$$\text{(B.2)} \qquad R_{2d} = \frac{R_{2c}}{h}$$

där h är samplingstidslängden. I de fall som σ_{az} och σ_q inte är kända kan jag rekommendera att R_{2d} estimeras rekursivt tillsammans med processidentifieraren enligt formeln

$$\text{(B.3)} \qquad \begin{aligned} \hat{R}_{2d}(t) &= (1-\mu)\left(1-\varphi(t)^T P_i(t)\varphi(t)\right)\varepsilon^2(t) + \mu\hat{R}_2d(t-1) \\ \varepsilon(t) &= y(t) - \hat{\theta}^T(t)\varphi(t) \end{aligned}$$

där $\mu \in [0\ 1]$ är den s k glömskefaktorn och indexet i står för den aktuella modellen M_i. Tyvärr har jag inte hittat någon möjlighet att estimera processbruset R_{1c} rekursivt. I de flesta fall brukar R_{1d} (eller R_{1c}) modelleras som vI där v är ett reellt positivt tal. För att få klarhet i att R_{1d}=vI är lika bra som den exakta R_{1d} göres därför en teoretisk härledning av R_{1d}. I figuren nedan införes den approximationen att anfallsvinkeln α=0.



Här antages det att endast vinden stör roboten (inget brus i rodret). Vinden "stöter" alltså på roboten. Endast anfallsvinkeln α påverkas eftersom vinden har inte har någon påverkan på rotationen q kring robotens tyngdpunkt. Då ges

$$\alpha_v \approx \tan\alpha_v = \frac{\omega_{tvärs}}{\omega_{vind}} \approx \frac{\omega_{tvärs}}{v_M} \tag{B.4}$$

Vindstötarna vinkelrätt roboten modelleras som

$$\omega_{tvärs} = \frac{1}{1+sT} e_e \qquad \text{där } e_e \in N(0,\sigma_e) \text{ och } T = 10 \text{ s} \tag{B.5}$$

Det vita bruset σ_e har spektraltätheten $R_e(f)= \sigma_e^2$. Enligt läran om stationära stokastiska processer är spektraltätheten $R_y(f)=|H(i2\pi f)|^2 R_u(f)$, där $R_u(f)$ och $R_y(f)$ är spektraltätheter för insignal respektive utsignal till ett system med överföringsfunktionen H(f). Därigenom fås

$$R_{\omega_{tvärs}}(f) = \left|\frac{1}{1+i2\pi fT}\right|^2 \sigma_e^2 \tag{B.6}$$

och eftersom variansen för utsignalen y är $\sigma_y^2=R_y(0)$ så fås

$\sigma_e^2 = R_{\omega_{tvärs}}(0) = \sigma_{\omega_{tvärs}}^2$, där $\sigma_{\omega_{tvärs}}$ kan uppskattas mellan 5 och 10 m/s. Standardavvikelsen σ_α för störningen α_υ i anfallsvinkeln kan beräknas enligt ekv (B.4).

$$\sigma_\alpha = \frac{1}{v_m}\sigma_e \tag{B.7}$$

Sambandet $R_{1c} = E\left[Gw(t)(Gw(t))^T\right] = GE\left[w(t)w^T(t)\right]G^T$ kan bestämmas genom att i ekv (3.1) addera den stokastiska delen α_υ till α. Därigenom fås processbrustermen

$$Gw = \frac{1}{2}S\rho(h)v^2 \cdot \begin{bmatrix} \frac{1}{mv}C_{N\alpha}(v) \\ \frac{d}{Jo}C_{m\alpha}(v) \\ 0 \\ 0 \end{bmatrix} \cdot \alpha_\upsilon \tag{B.8}$$

Ekv (B.8) och ovannämnda samband ger alltså

$$R_{1c} = \begin{bmatrix} k_{11}^2\sigma_\alpha^2 & k_{12}^2\sigma_\alpha^2 & 0 & 0 \\ k_{21}^2\sigma_\alpha^2 & k_{22}^2\sigma_\alpha^2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{B.9}$$

där koefficienterna k_{11},k_{12},k_{21} och k_{22} ges av ekv (B.10)

$$\text{(B.10)} \qquad \begin{cases} k_{11} = \frac{1}{2}\frac{S\rho(h)v^2}{mv}\cdot C_{N\alpha}(v) \\ k_{12} = k_{21} = \frac{1}{2}S\rho(h)v^2\sqrt{\frac{1}{mvJ_0}C_{N\alpha}(v)\cdot C_{m\alpha}(v)} \\ k_{22} = \frac{1}{2}\frac{S\rho(h)v^2 d}{J_o}\cdot C_{m\alpha}(v) \end{cases}$$

Systemet skall nu diskretiseras. Betrakta ekv (4.3). Den första av dessa ekvationer kan genom integration av den första ekvationen i ekv (4.2) skrivas

$$x_{t+1} = e^{Ah}x_t + \int_0^h e^{A(h-\tau)}[Bu(t+\tau)+Gw(t+\tau)]d\tau$$

$$= e^{Ah}x_t + \int_0^h e^{A(h-\tau)}Bd\tau\cdot u_t + \int_0^h e^{A\tau}Gw(t+h-\tau)d\tau = \Phi x_t + \Gamma u_t + w_t$$

där $w_t = \int_0^h \cdots d\tau$ är den diskretiserade processbrustermen. Dennas kovariansmatris kan då skrivas

$$R_{1d} = E\left[w_t w_t^T\right] = E\left[\int_0^h e^{A\tau}Gw(t+h-\tau)d\tau\cdot\int_0^h w^T(t+h-s)G^T e^{A^T s}ds\right]$$

$$= \int_{\tau=0}^{h}\int_{s=0}^{h} e^{A\tau}GE\left[w(t+h-\tau)w^T(t+h-s)\right]G^T e^{A^T s}dsd\tau$$

$$= \int_{\tau=0}^{h}\int_{s=0}^{h} e^{A\tau}R_{1c}\delta(\tau-s)e^{A^T s}dsd\tau = \int_{\tau=0}^{h} e^{A\tau}R_{1c}e^{A^T\tau}d\tau$$

Tyvärr är ||Ah|| för stort så att en beräkning av matrisintegralen m h a serieutveckling inte kan göras. I stället måste matrisintegralen beräknas på något annat numeriskt sätt. Integralen kan lämpligen omformuleras till ekv (4.5) vilket göres i det egna Matlabprogrammet *matrisintegral2*.

Emellertid använder regulatorn sig av den styrbara kanoniska tillståndsrepresentationen. Givet transformationsmatrisen

$$\text{(B.11)} \qquad T = W_s^{-1}\tilde{W}_s$$

där W_s är styrbarhetsmatrisen för tillståndet x enligt ekv (4.3) och

$$(B.12) \qquad \tilde{W}_s = \begin{bmatrix} 1 & a_1 & a_2 & \cdots & a_{n-1} \\ 0 & 1 & a_1 & \cdots & a_{n-2} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & & 1 & a_1 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}^{-1}$$

motsvarande styrbarhetsmatris för den styrbara kanoniska tillståndet. Då kan ett nytt styrbart kanoniskt tillstånd z=Tx beräknas. På så sätt kan systemet med det nya tillståndet z skrivas

$$(B.13) \qquad \begin{aligned} \dot{z} &= TAT^{-1}z + TBu + TGw(t) \\ y &= CT^{-1}z + e(t) \end{aligned}$$

Kovariansmatrisen för processbruset TGw(t) är

$$(B.14) \qquad R_{1c_{can}} = E\left[TGw(t)w^T(t)G^TT^T\right] = TR_{1c}T^T$$

och R_{2c} påverkas inte av transformeringen. På så sätt finns två val att beskriva R_{1d}, nämligen

$$(B.15) \qquad R_{1d} = vI$$

$$(B.16) \qquad R_{1d} = \int_0^h e^{A\tau}TR_{1c}T^Te^{A^T\tau}d\tau$$

Emellertid, om endast den styrbara kanoniska tillståndsrepresentationen finns tillgänglig, kan därför transformationsmatrisen T vilken ges av ekv (B.11) inte beräknas. Detta betyder att den korrekta kovariansmatrisen inte kan beräknas om endast processidentifieraren används, varför endast ekv (B.15) kan användas.

Simuleringarna har dock visat att när en fullständig modellkännedom finns blir resultaten endast marginellt bättre när ekv (B.16) används istället för ekv (B.15). Därför användes endast ekv (B.15) för hela simuleringen.

Appendix C: Referenser

Grundläggande reglerteknik för kontinuerliga system

[1] K. J. ÅSTRÖM, *Reglerteori*, Almqvist & Wiksell, Stockholm, 1985

Grundläggande reglerteknik för diskreta system

[2] K. J. ÅSTRÖM OCH B. WITTENMARK, *Computer-controlled systems*, Prentice-Hall, 1990

[3] R. H. MIDDLETON OCH G. C. GOODWIN, *Digital Control and Estimation, A Unified Approach*, Prentice-Hall, 1990

Processidentifiering

[4] L. LJUNG OCH T. SÖDERSTRÖM, *Theory and Pratice of Recursive Identification*, The MIT Press, London, 1983

[5] R. JOHANSSON, *Processidentifiering*, Lunds Tekniska Högskola, Lund, Augusti 1991

Adaptivitet

[6] K. J. ÅSTRÖM OCH B. WITTENMARK, *Adaptive Control*, Addison-Wesley, 1989

Segmentering av olinjära system och multimodellering

[7] S.L. WIRKANDER, *Om multi-modellering av olinjära dynamiska system*, FOA Rapport C 20862-8.4 (2.1), Stockholm, December 1991

[8] A. SKEPPSTEDT, *Construction of Composite Models from large Data-Sets*, LiU-TEK-LIC-1988:22 ISBN 91-7870-404-9, Linköping, 1988

[9] F. GUSTAFSSON, *Optimal Segmentation of Linear Regression Parameters*, LiU-TEK-LIC-1990:46 ISBN-91-7870-712-9, Linköping, 1990

[10] F. GUSTAFSSON, *Estimation of discrete parameters in linear systems*, ISBN 91-7870-876-1, Linköping, 1992

Elementär teori om stokastiska stationära processer och tidsserieanalys

[11] L. OLBJER, *Tidsserieanalys*, Lunds universitet och Lunds Tekniska Högskola, 1985

Appendix D: Programlistning

Programlistan består dels av de grafiska SimuLink-blockscheman, dels av Matlab-program. SimuLink-programmen som bifogas i rapporten är ordnat som ett träd där *Missilblock* är det översta systemet som omfattar båder den snabba regulator-loopen som hanteras av *Robot and Regulator* och den långsamma kinematikloopen vilket finns i *Kinematikblock*. *Orbit* är målmodellen och *Engine* missilens motor som funktion av tiden. *Robot*-blocket representerar ekv (3.1), d v s tillståndsmodellen för robotens mekanik. *Alfa-sys* simulerar det första ledet i ekv (3.1) därav namnet. Kinematikblocket innehåller tre differentialekvationer, nämligen ekv (3.6)-(3.8) samt ekv (3.5) för beräkning av missilens korrdinater x_M och y_M. Dessa delsystem visas i de därpå följande två sidorna med titlarna *rprim,vm,CD* från ekv (3.9) vilket används i i blocket för beräkningen av v_M, o s v.
De två sista sidorna är seperata SimuLink-program för offline process-identifiering och blocket för beräkningen av kovariansen R_1.

Matlabprogrammen är av naturliga skäl ganska omfattande. Här följer en lista över vilka program som används och vilka ekvationer som dessa program beräknar.

loadkoef - hämtar datavärden $C_{N\alpha}$,$C_{N\delta}$, o s v i tabellform.
loadkin - hämtar data för målmodellen
initkin - beräknar initialvärdena av r, ϕ, och κ enligt (3.4)-(3.5)
datainit - hämtar datavärden m, S, o s v vilket bl a används i (3.1)
matrisinit - skapar matriser Q, R, o s v som beskrivs i sektion (4.2)-(4.3)

regulator - simulering av den digitala datorn och omfattar hela kapitel 4

binearsearch - söker modell på basis av (v,h) enligt sektion (5.4)
fetchmodel - hämtar modellen $A(q^{-1})$ och $B(q^{-1})$ ur modellbiblioteket
transformmodel - behandlar modellövergångar från q-operator till δ-operator enligt sektion 4.5, balansering och modell-reduktion för båda operatorerna (sektion 4.6) och kanonisk realisering

deltagram - beräknar gramianen enligt (4.55),(4.60) och (4.65)
deltabalreal - beräknar balanserad realisering enligt (4.66)
invWc - ger inversen av styrbarhetsmatrisen W_s enligt (B.12)
obervermat - observerare från sektion 4.2 och 4.5, bl a kalmanfilter vilka ges av (4.8)-(4.9) och (4.38)-(4.41)
styrlagmat - beräkning av styrlagar enligt sektion 4.3 och 4.5

deltalqry - LQy-reglering enligt definition (4.21) och (4.22) och anrop av deltalqr
deltalqr - LQ-styrlag med definitionen enligt (4.19) och (4.20) och beräknar ekv (4.46), (4.49) och (4.50)

rho - beräknar densiteten för luften
hit - stoppar programmet om avståndet r blir mindre än hittol
myswitch - relä som skiftar insignal om den understiger ett visst värde

id12q - offline processidentifiering för q-operatorn enligt sektion 5.1
id12delta - offline processidentifiering för δ-operatorn enligt sektion 5.2

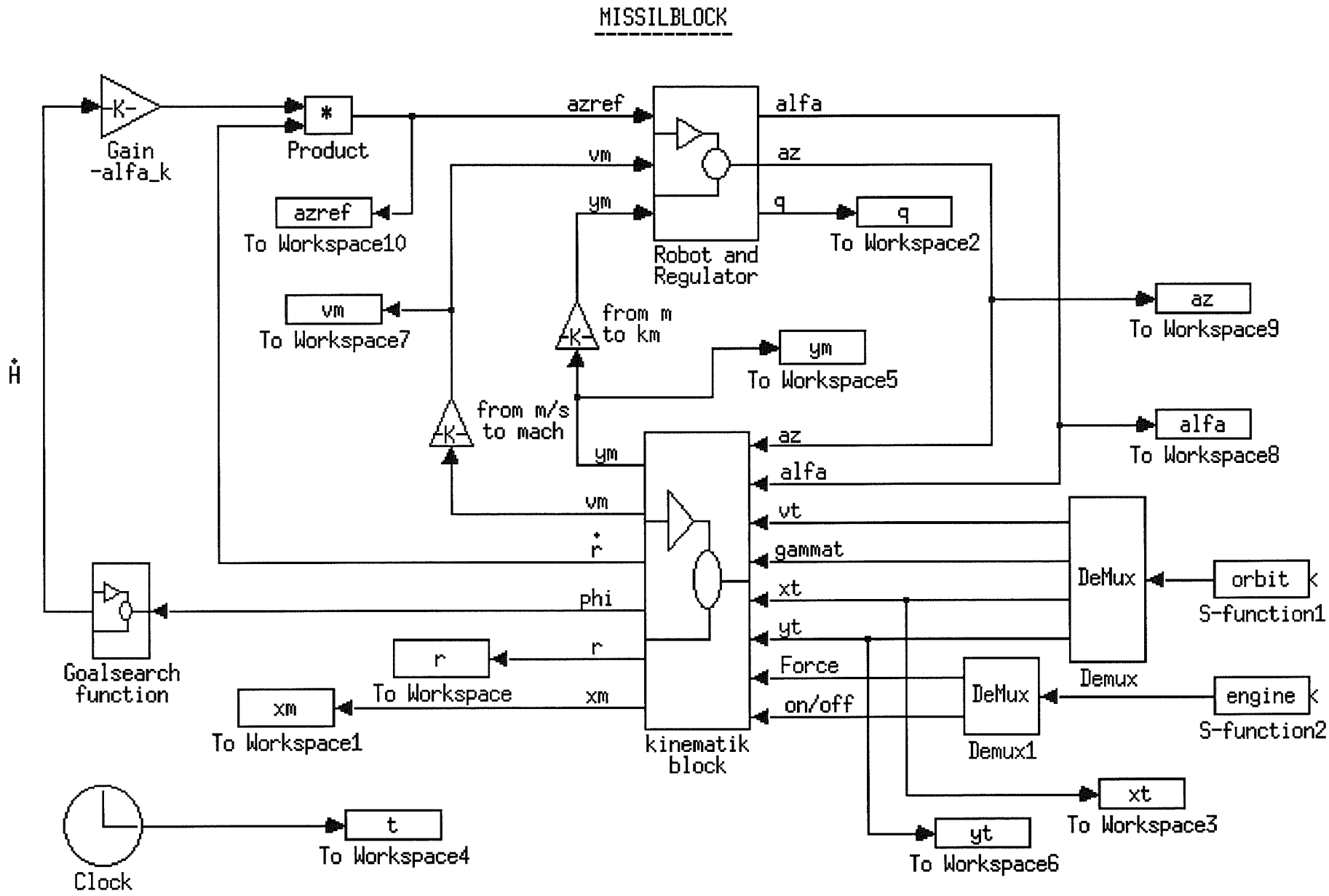
modeltablemod - ger en konstant segmentation enligt tabellen på sektion 5.3
modeltable - ger en variabel segmentation enligt kriteriet (5.27)

matrisintegral2 - beräknar (B.16)
calc_R1d - beräknar R_{1d} på basis av R_1

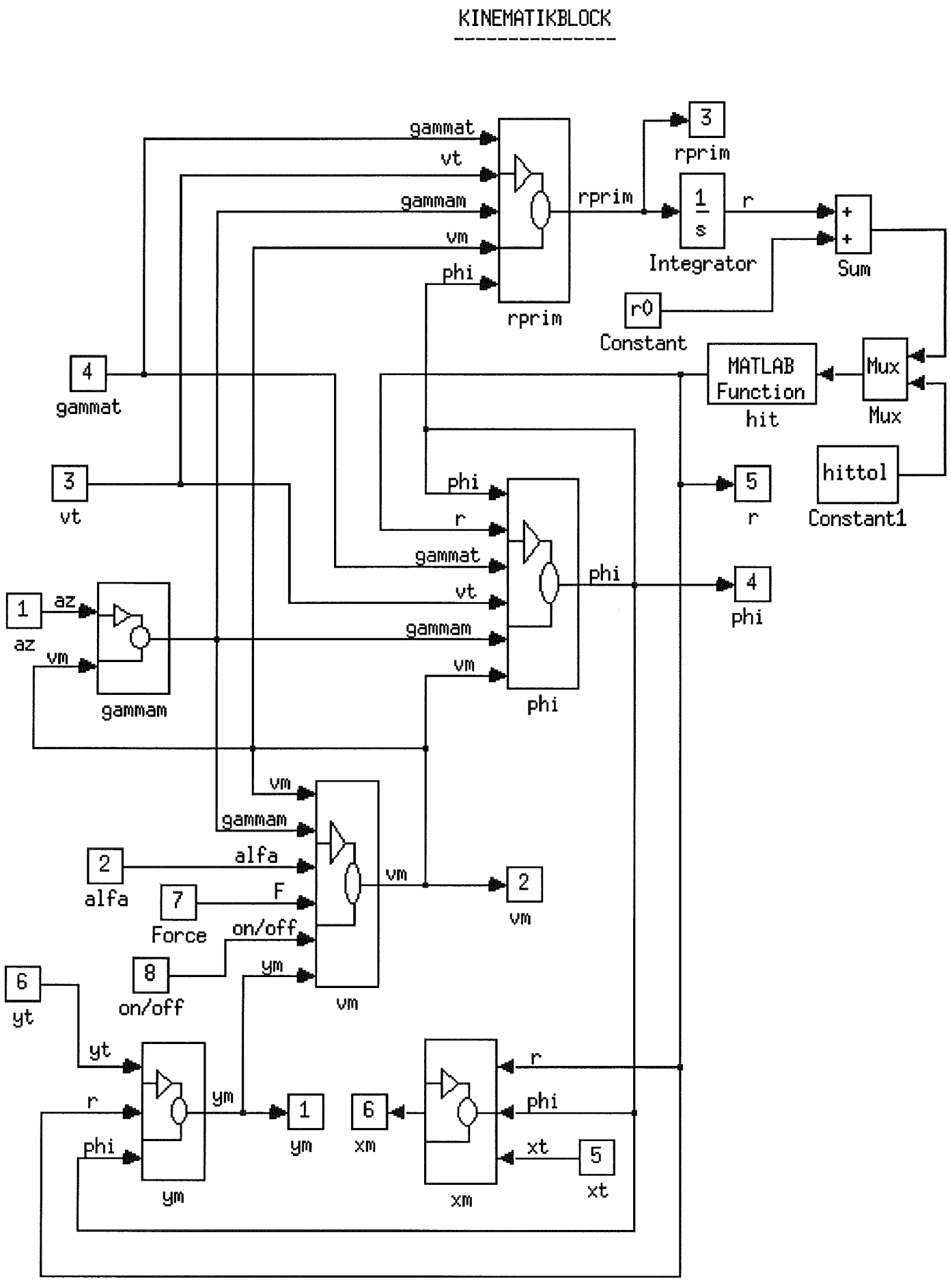
newlib - administrerar modelluppbyggnaderna
autosimlib - bygger upp ett modellbibliotek givet segmentationen av modellområdet
autosimlibold - äldre och långsammare version av autosimlib men mera pålitlig p g a märkvärdiga programproblem
modelbackup - tar kopia av modellen och sparar (v,h)
buildmodel - givet en modell så söker buildmodel den rätta modell-indexet och sätter in denna modell
insertmodel - givet modelindex så raderar den gamla modellen och sätter den nya modellen

För att underlätta förståelsen av S-funktioner och deras uppbyggnad som beskrives utförligt i sektion 2.4 så bifogas ett gammalt Matlab-program *id12PotterBrusQ* för processidentifieraren enligt Potters algoritm i sektion 6.1.

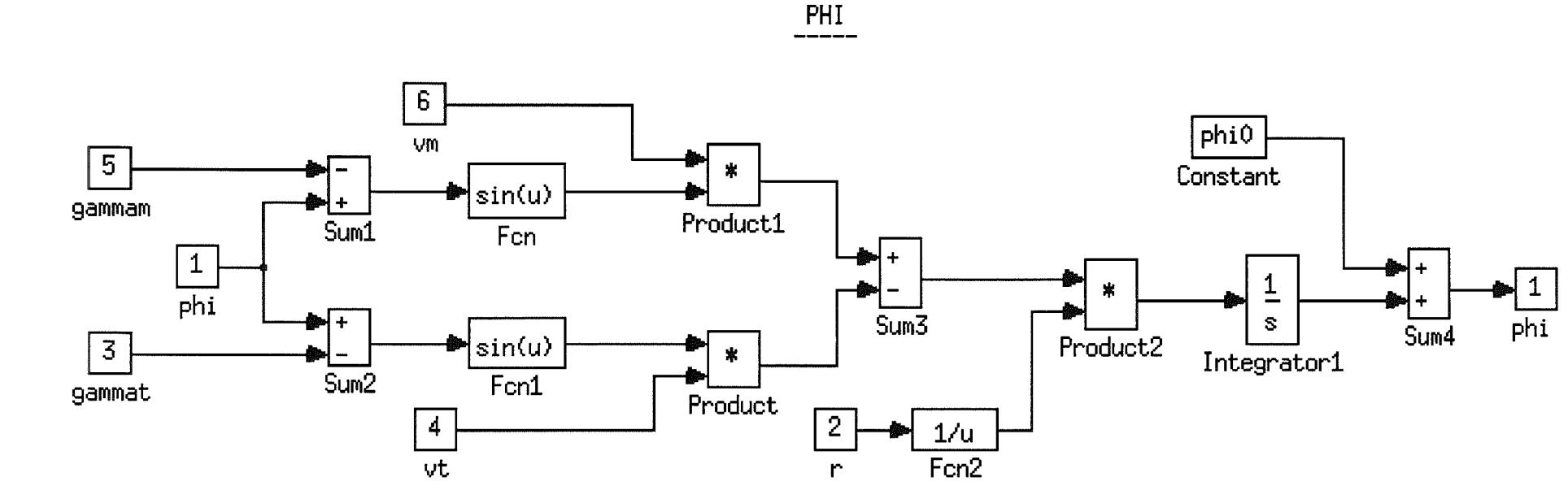
File Edit Options Simulation Style

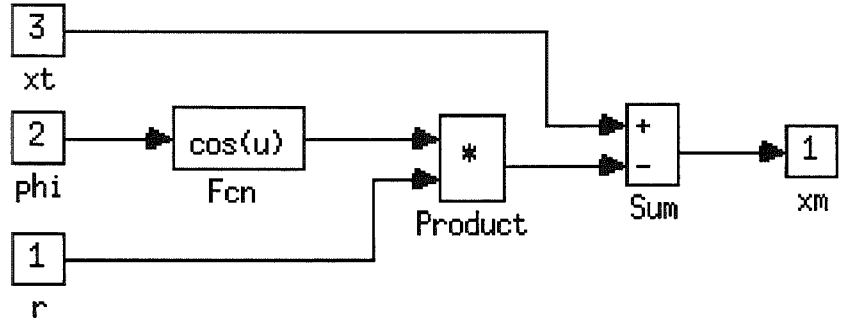


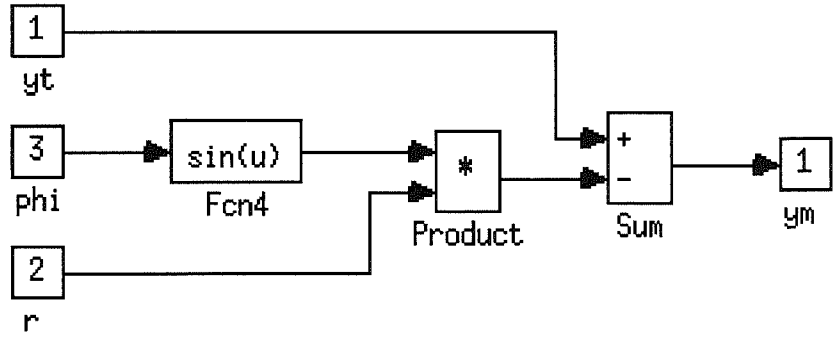
File Edit Options Simulation Style



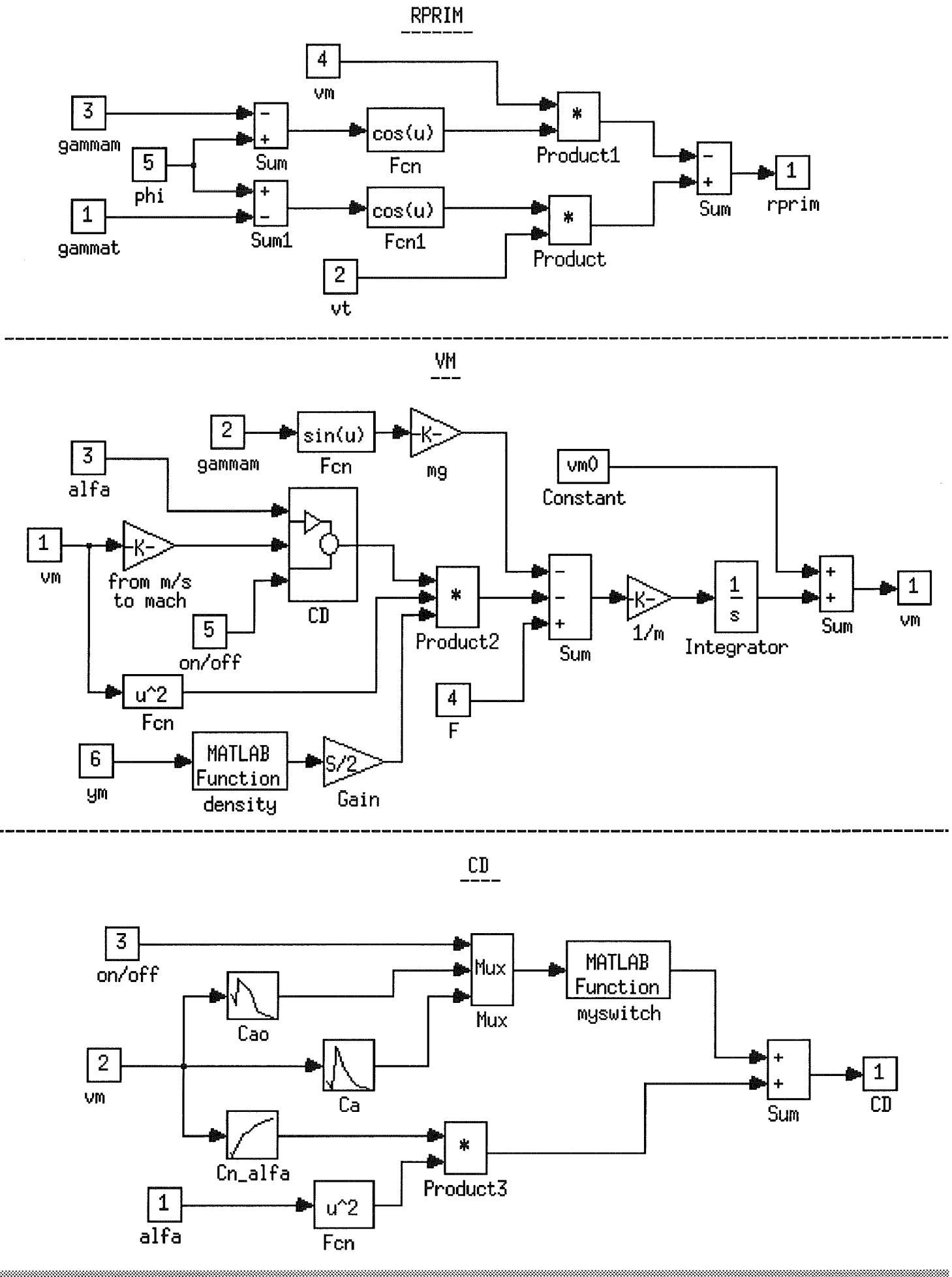
File Edit Options Simulation Style



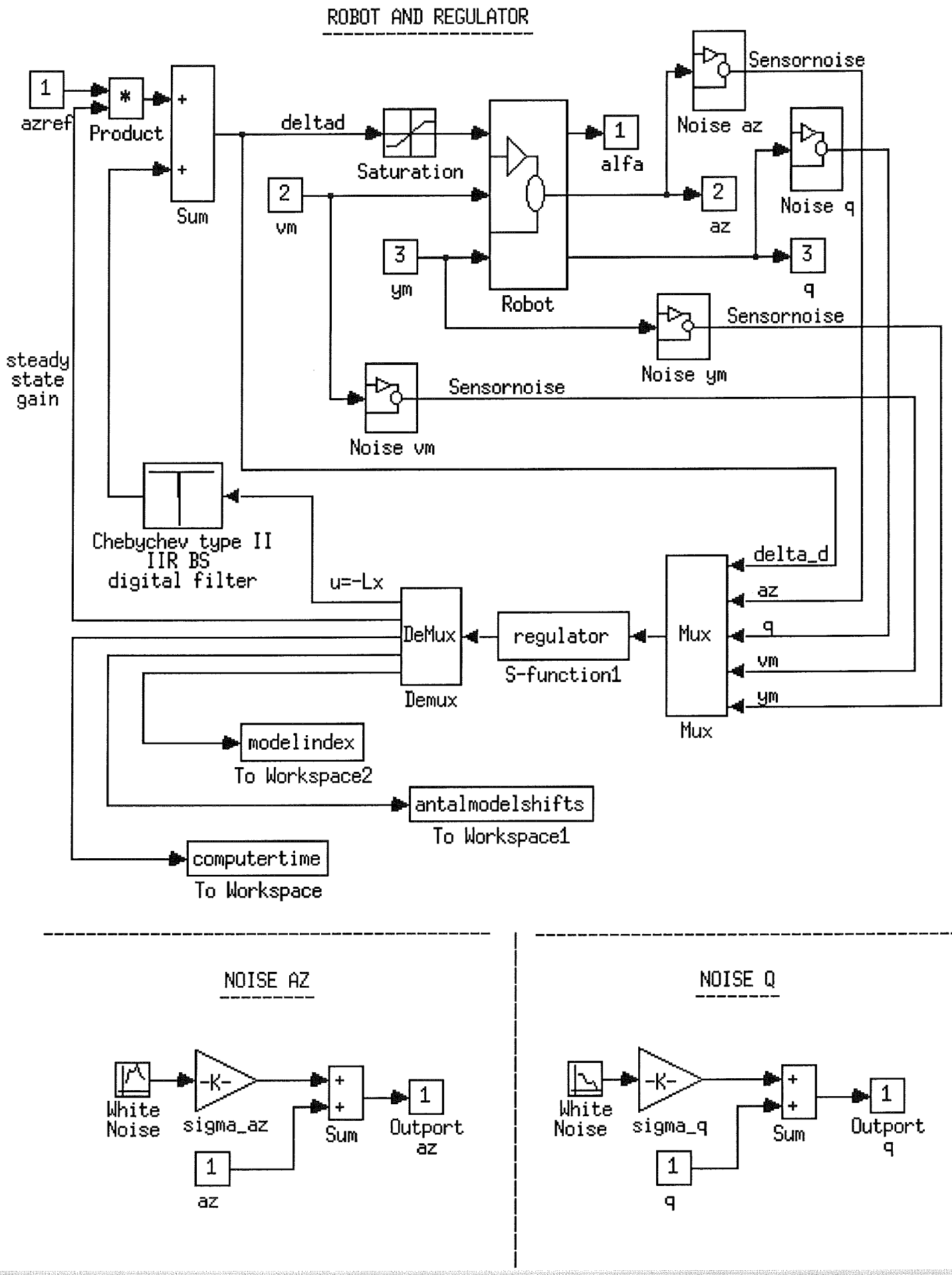




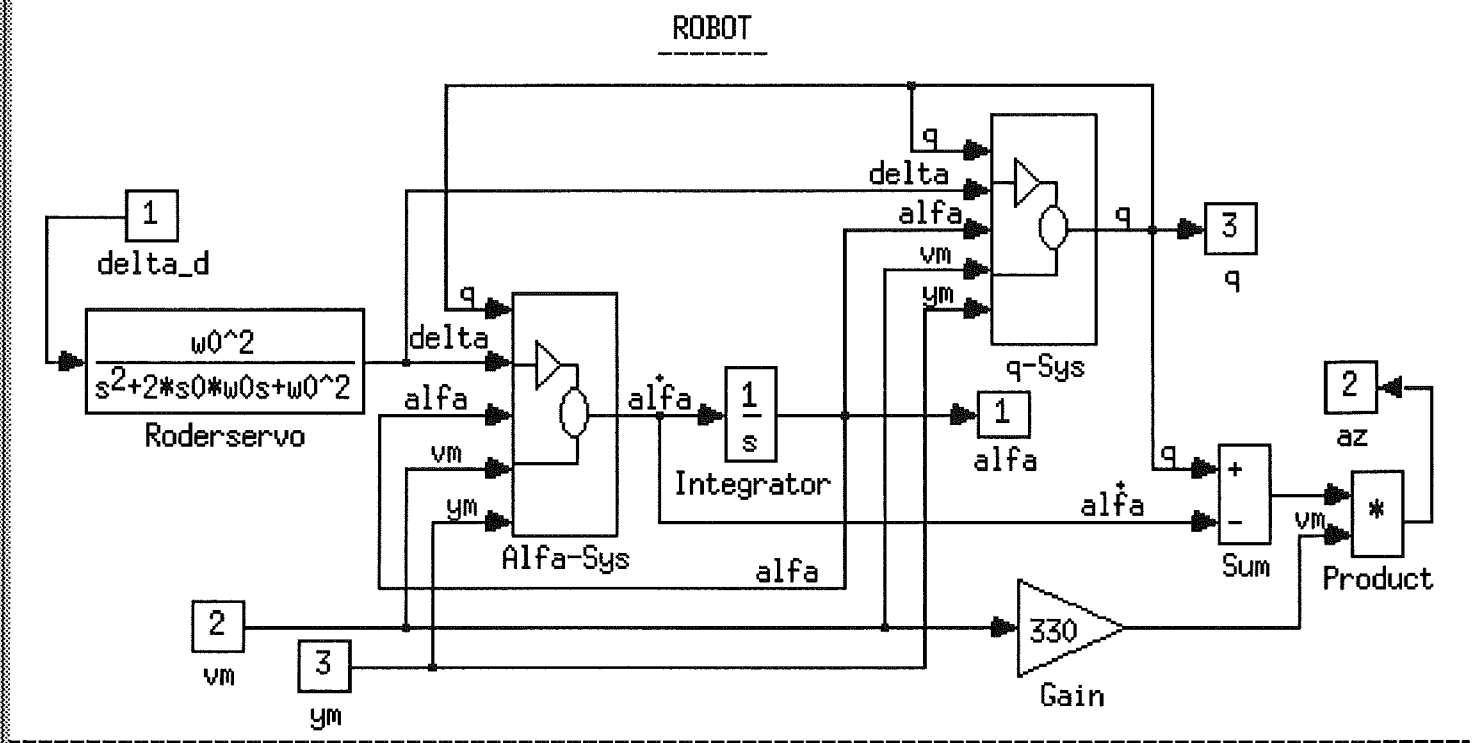
File Edit Options Simulation Style

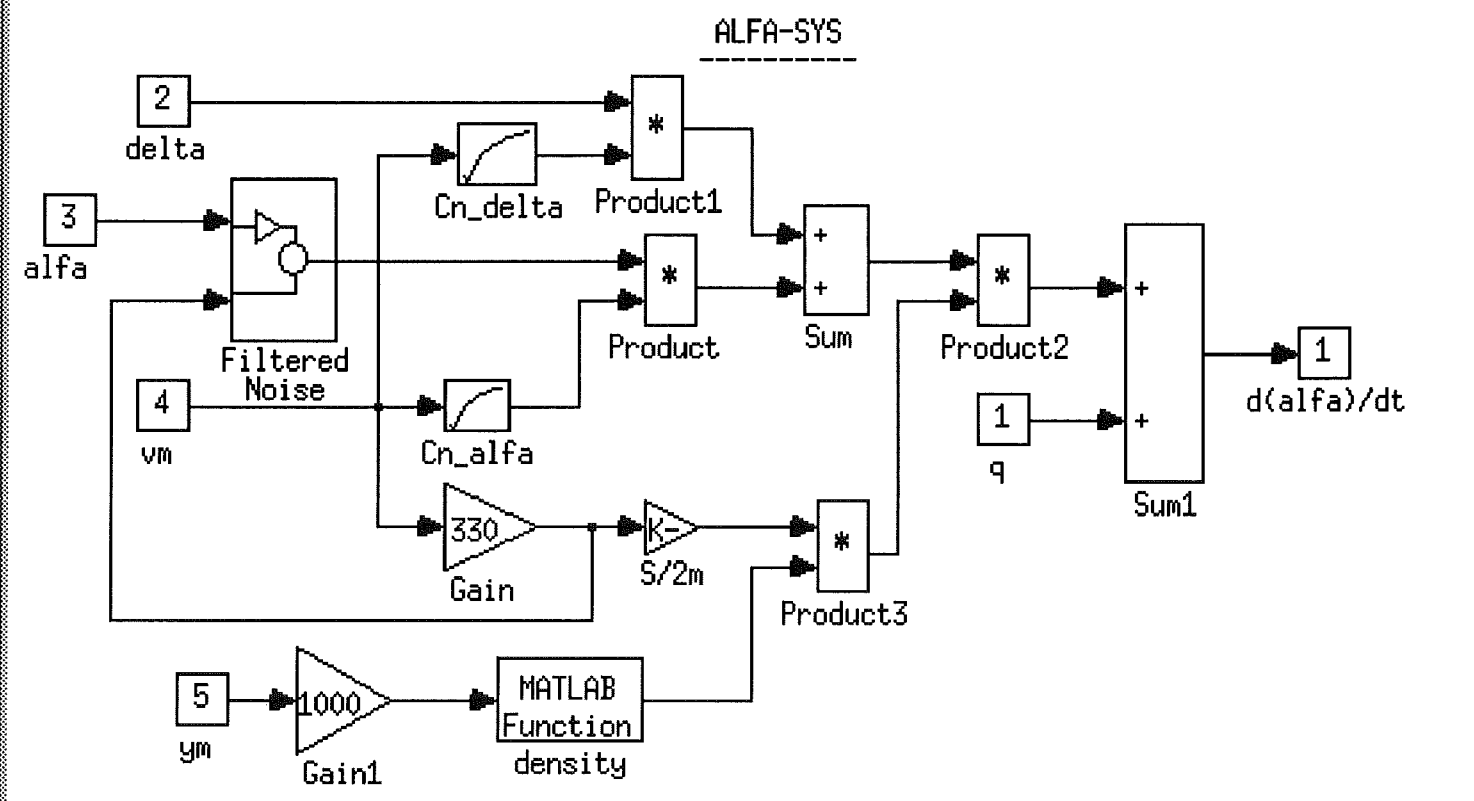


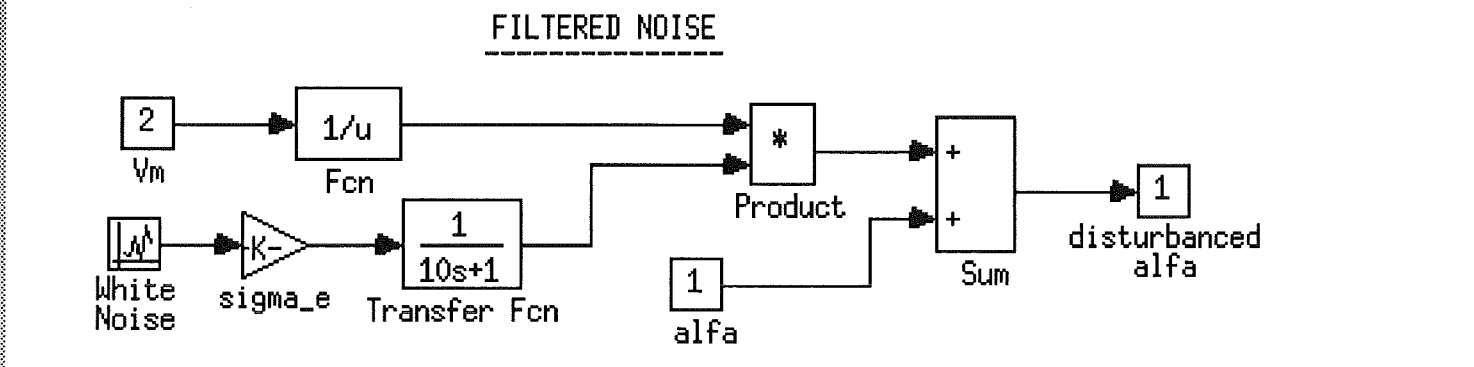
File Edit Options Simulation Style



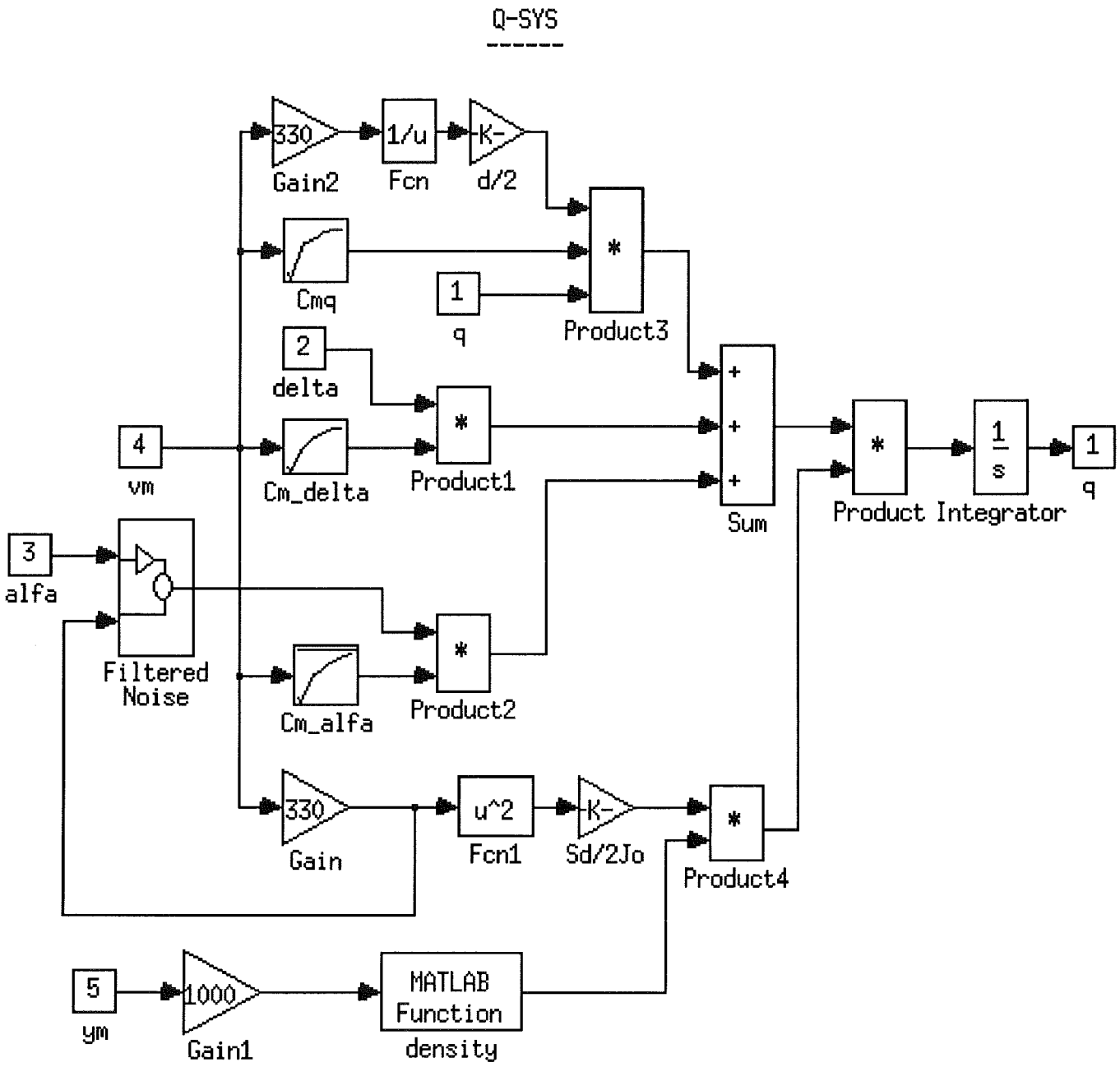
File Edit Options Simulation Style

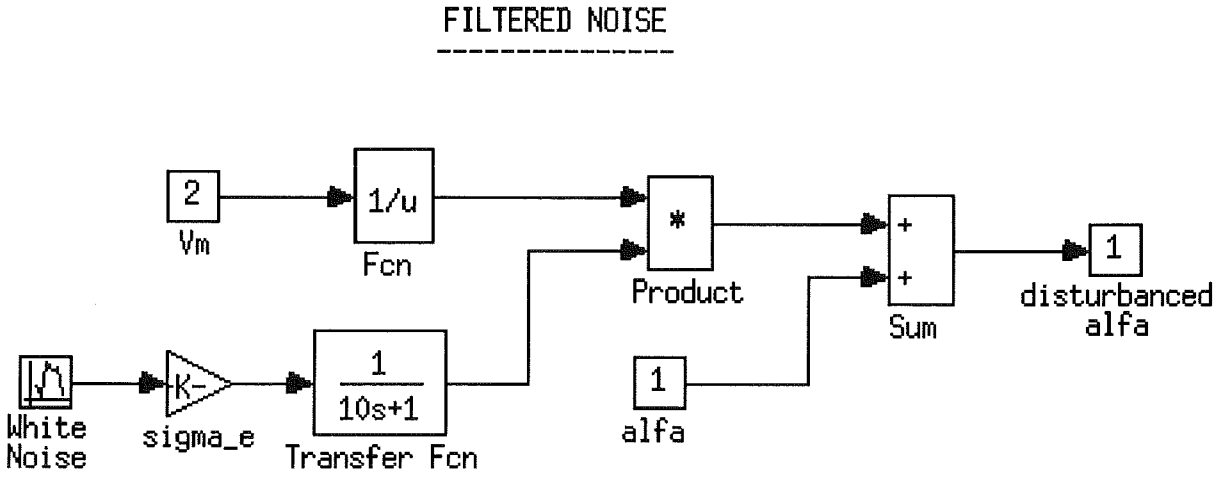






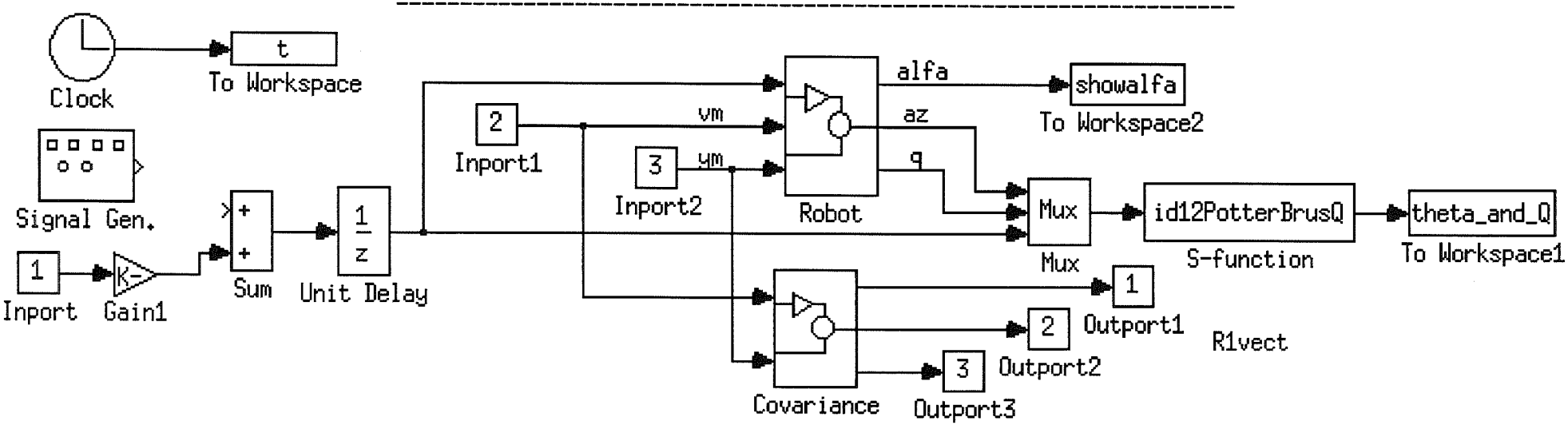
File Edit Options Simulation Style



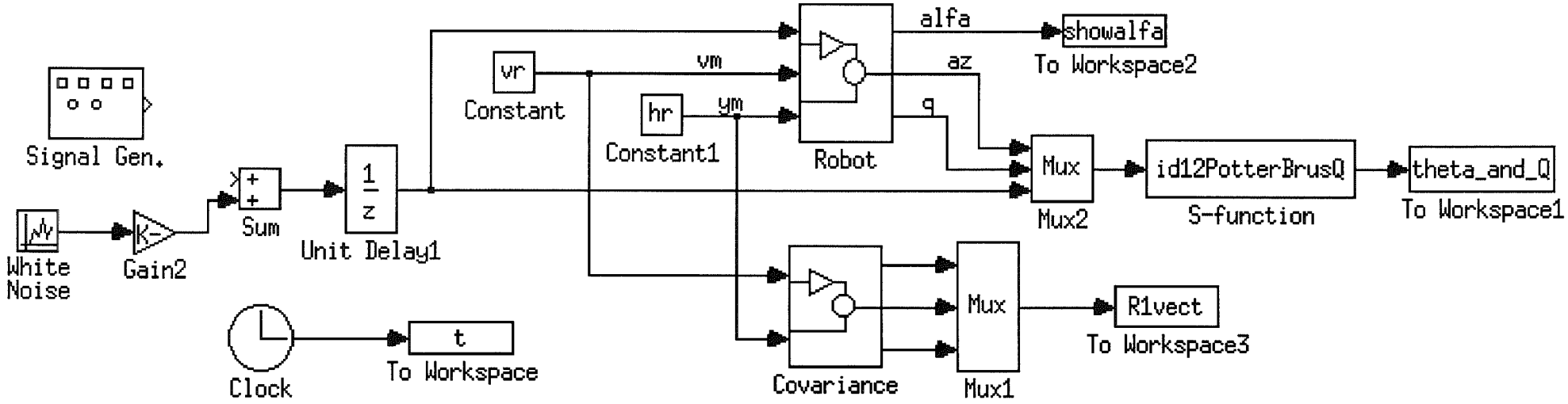


File Edit Options Simulation Style

SEPERAT PROCESSIDENTIFIERING, PARAMETRARNA (v,h) SOM INSIGNALER



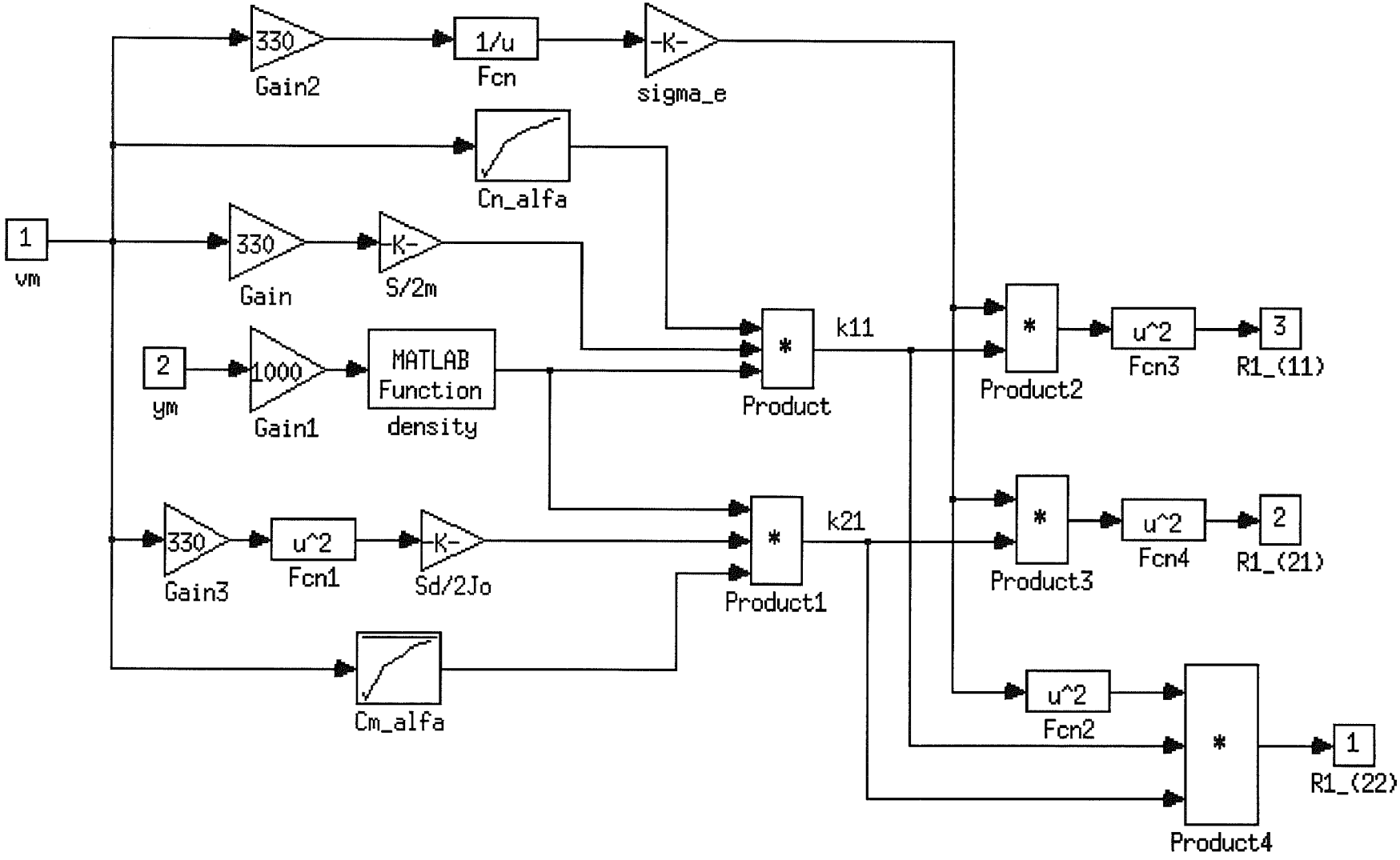
SEPERAT PROCESSIDENTIFIERING, PARAMETRARNA (v,h) SOM MATLAB-KONSTANTER



File Edit Options Simulation Style

COVARIANCE

This block calculates the three elements of the covariancematrix R_{1c} (processnoise)



```
function[sys,x0]=id12PotterBrusQ(t,x,u,flag,ts,na,nb,d,theta0,P01,R2)
%ID12POTTERBRUSQ [sys,x0]=id12PotterBrusQ(t,x,u,flag,ts,na,nb,d,theta0,P01,R2)]
%
%               This is a S-function. It processidentify a SIMO system
%               ( 1 input 2 outputs) by use of Potters algorithm.
%               ts is sampletime and na is the number of parameters of A(q).
%               nb is the number of parameters of B(q).
%               theta0 are initialvector and P01 initialmatrix (positive
%               definite).
%               R2 is sensor noise matrix (positive definite)
%
error(nargchk(11,11,nargin));
n=na+2*nb;        % Parametervector's dimension
nfi=2*na+nb;      % Regressionsvector's dimension
offset=0;
if flag == 0,     % Initialvalue for statespace-vector
  x0=theta0;  Q0=chol(P01);
  for j=1:n, x0=[x0;Q0(:,j)]; end
  x0=[x0;zeros(nfi+d-1,1)];
  sys=[0;(n+1)*n+nfi+d-1;n*n+n;3;0;0];   %Systeminfo

elseif abs(flag) == 2,   % Next discret statespace
  if ~(abs((round((t-offset)/ts)-(t-offset)/ts))<(ts*0.1))
    sys=x;
  else
    theta=x(1:n);  Q=[];   % Get theta and Q
    for j=1:n, Q=[Q x(j*n+1:j*n+n)]; end
    fi=x(n+n^2+1:n+n^2+nfi);
    fi1y=fi(1:na);    fi2y=fi(na+1:2*na);   fiu=fi(2*na+1:nfi);   % get fi
    fi1=[fi1y;fiu;zeros(nb,1)];   fi2=[fi2y;zeros(nb,1);fiu];
    FI=[fi1 fi2];
    uold=x(n+n^2+nfi+1:n+n^2+nfi+d-1);     % and insignal
    ypred=FI'*theta;                       % predicted y
    y=u(1:2);                              % get outsignals
    yerr=y-ypred;                          % estimate-error of y

    f=Q'*FI;                          % Potters algorithm starts now here
    beta=R2+f'*f;
    alfa=inv(beta+sqrt(beta*R2));
    Ls=Q*f;
    Q=(Q-Ls*alfa*f');                 % and algoritm finishs here

    theta=theta+Ls*inv(beta)*yerr;   % update estimatevector theta
    fia1=fi1y(1:na-1);            % erase the oldest values, i e shift vectors
    fia2=fi2y(1:na-1);
    fib=fiu(1:nb-1);
    if d>1,
      fi=[-y(1);fia1;-y(2);fia2;uold(d-1);fib];     % update fi-vector d steps
      uold=[u(3);uold(1:d-2)];                      % and shift and save uold
    else
      fi=[-y(1);fia1;-y(2);fia2;u(3);fib];       % update fi-vector one step
    end
    sys=theta;
    for j=1:n, sys=[sys;Q(:,j)]; end
    sys=[sys;fi;uold];                % feed out statespace-vector
  end

elseif flag == 3  % Output (=parameterestimate theta and Q)
  sys=x(1:n+n*n);
elseif flag == 4  %Next time interval for update
  ns=(t-offset)/ts;
  sys=offset+(1+floor(ns+(1e-13)*(1+ns)))*ts;
else
  sys=[];
end
```