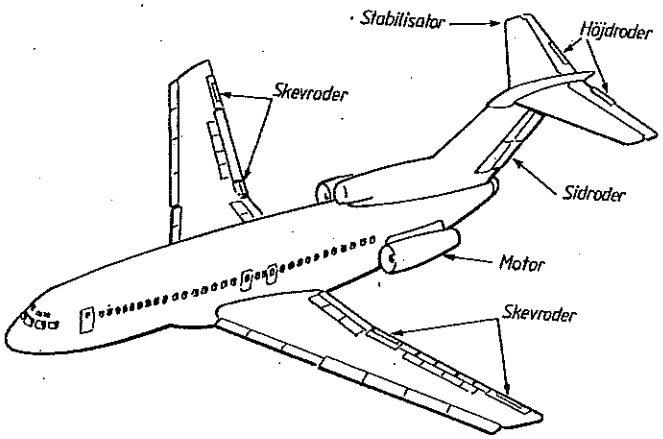
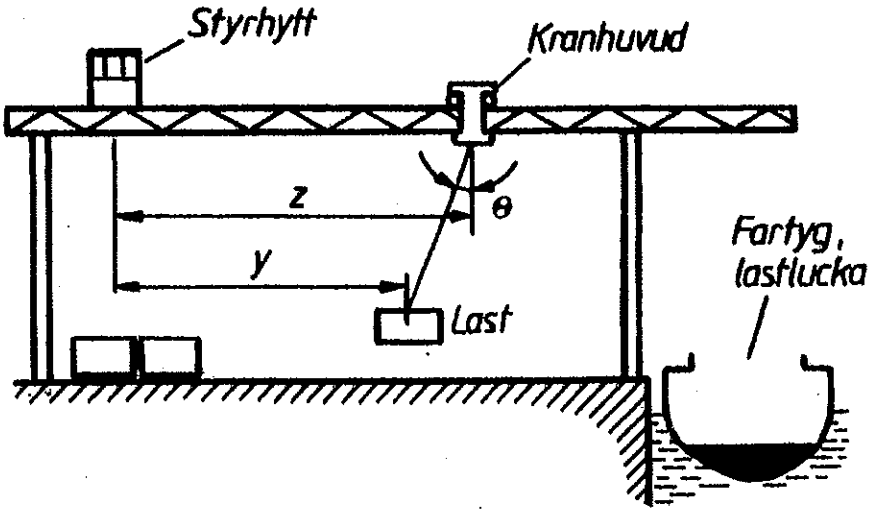
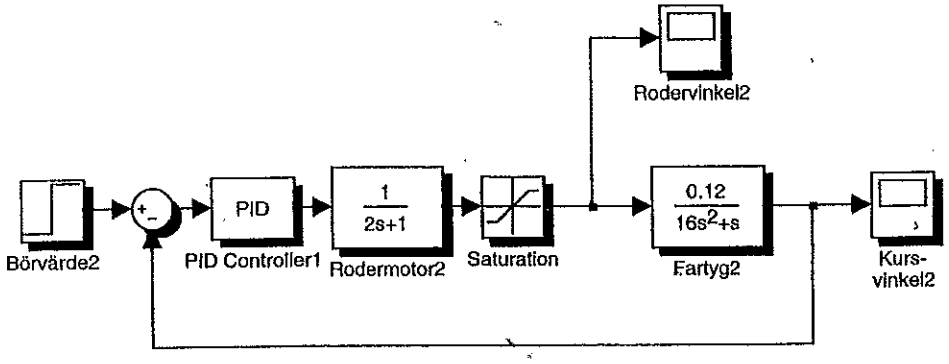
MATLAB & SIMULINK

Inlämningsuppgift 1-3
Version 2







INLÄMNINGSUPPGIFT 1

Inlämningsuppgift 1 behandlar grundläggande kommandon för att komma igång med att lösa enkla matematiska problem i Matlab, grafik mm. Innan du sätter dig vid datorn måste Du ha läst kapitel 24 i Modern reglerteknik samt de extrablad om Matlab som ligger efter själva uppgifterna i denna inlämningsuppgift. **Du måste lösa alla uppgifter. Som kontroll räcker det dock att Du lämnar in givna kommandon och svar på följande uppgifter:**

1.1, 1.5d, 1.5g, 1.6, 1.7, 1.8, 1.9, 1.15, 1.18, 1.19, 1.23.

OBS. Använd hjälp-kommandot "help" om Du behöver hjälp med något specifikt kommando.

Grundläggande kommandon i Matlab.

1.1) Börja med att genomföra följande beräkningar med hjälp av MATLAB:

$\sin(0{,}2 \cdot \pi)$

$e^2 + \sin(0{,}3)$

$(e^3 + \cos(0{,}4)) / (1 + 3.4^2)$

$(3 + 2i) \cdot (5 + 7i)$

$(2 + 4i)^3$

1.2) Mata därefter in följande radvektor i MATLAB:

$$x = \begin{bmatrix} 1.3 & 2.2 & 3.5 & 5.1 & 7.0 & 3.4 & 2.4 & 6.1 & 4.2 & 1.4 \end{bmatrix}$$

1.3) Beräkna med hjälp av olika MATLAB-kommandon följande:

a) summan av elementen i vektorn x
b) medelvärdet av elementen i x.
c) medianvärdet av elementen i x (använd kommandot median)
d) rita ett stapeldiagram för värdena i vektorn x (med kommandot bar(x))
e) tag fram en vektor y som innehåller elementen i x sorterade i storleksordning (med kommandot sort(x))

1.4) Mata nu in följande matriser i MATLAB:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \\ 0 & 2 & 4 \end{bmatrix}$$

$$B = \begin{bmatrix} 2 & 3 & 1 \\ 0 & 3 & 6 \\ 1 & 1 & 1 \end{bmatrix}$$

1.5) Använd olika MATLAB-kommandon för att beräkna följande:

a) Beräkna produkten A*B
b) Beräkna produkten B*A
c) Bestäm transponatet till B, dvs $C=B^T$
d) Bestäm inversen till A-matrisen
e) Bestäm inversen till B-matrisen
f) Bestäm B^3
g) Bestäm determinanten till B.
h) Bestäm egenvärdena till A-matrisen.

1.6) Bestäm med MATLAB lösningen till följande ekvationssystem:

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \\ 0 & 2 & 4 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 5 & 1 & 2 \\ 0 & -3 & 2 & 3 \\ 2 & 1 & 0 & 0 \\ 3 & 3 & 3.2 & -3 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 10 \\ 3 \end{bmatrix}$$

1.7) Bestäm med MATLAB lösningen till följande överbestämda ekvationssystem:

$$\begin{bmatrix} 1 & 4 & 3 \\ 3 & 0 & 3 \\ -2 & 1 & -2 \\ 4 & 3 & 7 \\ 0 & 3 & 3 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 9.2 \\ 5.8 \\ -4.3 \\ 14.5 \\ 5.7 \end{bmatrix}$$

1.8) Bestäm med MATLAB lösningen (rötterna) till följande ekvationer (Använd kommandot roots(x)):

$$x^3 + 2x^2 + 3x + 1 = 0$$

$$5x^5 + 3x^4 + 1{,}5x^3 + 2x + 4 = 0$$

$$3x^8 + 1 = 0$$

1.9) Bestäm hur det polynom ser ut som har följande rötter (Använd kommandot poly(x) för att lista ut svaret)

x=2, x=3, x=4, x=-1

1.10) Testa följande kommandon och lär dig vad de används till:

a) length(x)
b) clear A
c) floor(3.8)
d) ceil(3.8)
e) round(5.3)
f) D=rand(3)

1.11) Hur gör man för att på ett enkelt sätt skapa en radvektor X som innehåller 50 element och där elementen är siffrorna från 1 till 50 i stigande ordning ?

1.12) Hur gör man för att på samma sätt skapa en radvektor Y med 50 element, där elementen utgörs av siffrorna 5, 10, 15, 20 ... upp till 250 ?

1.13) Hur gör man för att slå ihop de två radvektorerna X och Y till en matris A där första raden är vektorn X och andra raden vektorn Y ?

1.14) Hur gör man för att plocka ut kolumn 10 och 11 i matrisen A ?

Grafik, programslingor mm

1.15) Skriv en programslinga (med hjälp av FOR-kommandot) för att beräkna summan av alla tal mellan 0 och 500, resp mellan 0 och 2000. Vad blir svaren ?

1.16) Mata in följande vektorer i MATLAB

$$x = [1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9 \quad 10]$$

$$y = [3 \quad 2 \quad 6 \quad 7 \quad 8 \quad 6 \quad 2 \quad 5 \quad 0 \quad 3]$$

1.17) Testa därefter följande kommandon:

plot(x)
plot(y)
plot(x,y)
plot(y,x)
plot(x,y,'.')

1.18) Plotta funktionen $f(x) = sin(x) + cos(0.2x)$ på intervallet x=0-50, med avståndet 0,2 mellan punkterna, dvs med 251 punkter totalt på kurvan.

1.19) Använd kommandot *subplot* för att rita följande fyra kurvor i fyra olika delfönster på ett och samma grafikfönster:

sin(x)
cos(x)
sin(0.5x)
cos(0.5x)

1.20) Rita på nytt funktionen $f(x) = sin(x) + cos(0.2x)$ på intervallet x=0-50, med avståndet 0,2 mellan punkterna, dvs med 251 punkter totalt på kurvan. Namnge därefter axlarna med kommandona *xlabel* resp *ylabel*.

1.21) Prova kommandot *gtext*, för att placera en textsträng på godtycklig plats inuti det aktuella plotfönstret. Prova därefter kommandot ginput för att avläsa koordinaterna på olika punkter längs kurvan i plotfönstret.

1.22) Mata in en matris A med 5 rader och 5 kolumner. Vad blir resultatet om du därefter ger kommandot *plot(A)* ? Hur tolkas kurvorna som fås ? Läs mer om plotfunktionen genom att anropa kommandot *help plot*. Titta också på den beskrivning av andra kommandon som ges av kommandot *help*.

1.23) Skriv en M-fil i MATLAB (ett litet program) som utför följande uppgift: Indata till programmet är en matris A (av valfri storlek). Programmet skall under körning fråga efter ett reellt tal x. Efter inmatning av detta tal skall programmet dividera alla element i A-matrisen med talet x och skriva ut resultatet på skärmen. Programmet skall anropas med namnet *delning*. Programmet skall ge någon form av felutskrift om man försöker dividera med talet noll.

1.24) Hur skriver man kommentarer i en kommandofil ?

1.25) Hur fungerar pause-kommandot ?

1.26) Skriv en funktionsfil som beräknar medelvärdet och standardavvikelsen för elementen i en godtycklig vektor x.

Mer om Matlab

Här följer lite ytterligare information om vanliga kommandon i Matlab utöver de som finns beskrivna i boken Modern reglerteknik. Ett par av sidorna är kopierade ur skriften "Användarhandledning för Matlab – version 5.3" (av Lennart Edsberg, KTH).

Funktionsfiler och M-filer

Även om MATLAB har mängder av specialkommandon för matematik och vetenskap har man ibland behov av att skriva och spara egna små program för olika ändamål. Det finns två sätt att göra detta i MATLAB, antingen i form av M-filer eller i form av funktionsfiler:

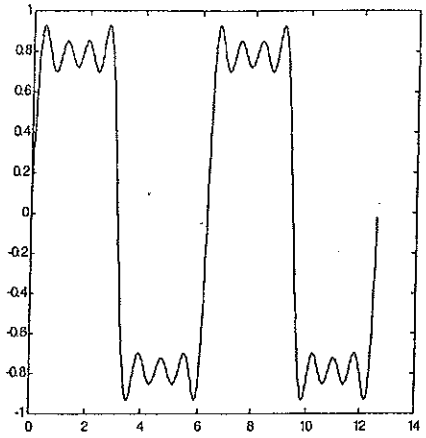
M-filer (även kallade script-filer)	En M-fil är helt enkelt en följd av MATLAB-kommandon som sparats under ett lämpligt namn. Vid anrop exekveras alla de i filen ingående kommandona ett efter ett. M-filen kan använda de variabler som definierats i Matlabs kommandofönster under den aktuella sessionen.
Funktionsfiler	En funktionsfil fungerar som en M-fil, frånsett att den kan anropas med en eller flera invariabler och att den lämnar ifrån sig en eller flera utvariabler. Funktionsfilen måste börja med nyckelordet function följt av namnet på funktionen. Lokala variabler som används i en funktionsfil är inte tillgängliga utanför funktionen.

Hur skapar man en m-fil eller en funktionsfil ? Det enklaste sättet är att använda den editor som finns inbyggd i MATLAB. Editorn öppnas genom menyvalet *File/New/M-file* under MATLAB:s *Command Window*. I denna editor skriver du sedan in de Matlabkommandon som ska ingå i ditt program, varefter du sparar det hela under lämpligt namn med hjälp av menykommandot *File/Save as..* Namnet på alla m-filer och funktionsfiler ska avslutas med extensionen .m. Under förutsättning att filerna sparats i den speciella mappen för m-filer (vilket är default) kan de sedan exekveras genom att filnamnet ges från den vanliga Matlab-prompten. För en funktionsfil ges filnamnet tillsammans med värdet på aktuella invariabler.

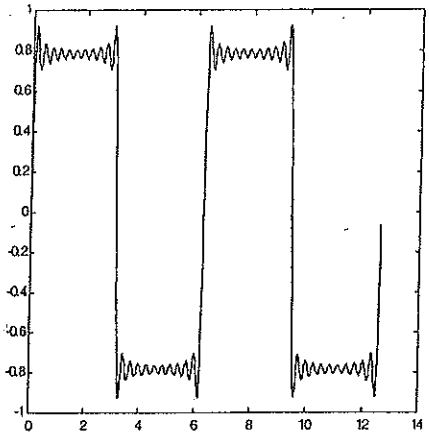
Här nedan visas exempel på en funktionsfil med namnet fourier(n). Filen beräknar och plottar en fyrkantvåg approximerad med dess fourierserie på intervallet $0 \le x \le 4 \cdot \pi$ Invariabeln n anger hur många termer som ska tas med i fourierserien. Ju större värde på n, desto bättre blir approximationen.

```
function fourier(n)
x=[0:0.02:4*pi];          % skapande av vektor med x-värden.
y=0*x;                    % skapande av startvektor för y.
for k=0:n                 % loop för beräkning av fouriersumman.
   p=2*k+1;
   y=y+(1/p)*sin(p*x);
end
plot(x,y)                 % plottning.
```

Ovanstående funktionsfil anropas från Matlab-prompten med kommandot fourier(n). Då kommandot fourier(3) resp fourier(10) ges erhålls t ex följande kurvor.



fourier(3)



fourier(10)

Som framgår av ovanstående exempel går det bra att använda en for-loop då man vill att en grupp av kommandon ska bli exekverad ett önskat antal gånger. For-loopar kan användas, både i m-filer, funktionsfiler och vid vanlig interaktiv körning. Kommandot *while* exekverar en grupp av kommandon så länge ett logiskt uttryck är sant. Gruppen av kommandon avslutas med ett end. Matlab använder följande generella syntax för for-loopar resp while-loopar, se nedanstående exempel:

for *variabel=uttryck*
    grupp av kommandon
end

while *logiskt uttryck*
    grupp av kommandon
end

y=0;
for k=1:100
  y=y+k;
end
	For-loop som summerar alla talen från ett till hundra.

y=0;
for k=0:2:100
  y=y+k;
end
	For-loop som summerar alla jämna tal från noll till hundra. Kolonnotationen 0:2:100 definierar ett startvärde (k=0), ett tillväxtvärde (2) och ett slutvärde (k=100).

» y=0; for k=0:1:100; y=y+k; end; y y = 5050	For-loop med flera kommandon på samma rad, givna i Matlab:s vanliga kommandofönster. Summerar alla talen från ett till hundra.

k=0;y=0;
while y<1000
  k=k+1; y=y+k;
end; k
 	
k= 45
	While-loop som beräknar hur många tal 1+2+3... som måste summeras för att komma upp i summan 1000.

3.6 ATT UTVIDGA REDAN DEFINIERADE MATRISER

MATLAB håller reda på alla matrisdimensioner. Man kan därför lätt ändra storleken på matriser om man vill lägga till fler element, rader eller kolumner. På detta vis är det enkelt att bygga nya matriser genom att använda delar av gamla matriser. Man använder samma regler för att slå samman matriser som att tilldela matriser, dvs enskilda element skiljs åt med blanktecken eller kommatecken och rader skiljs åt med semikolon eller Enter.

Ex. Antag att matriserna A och B, radvektorerna x och z samt kolumnvektorn y tilldelas värden:

```
>>A=[1 2;3 4]; B=[5 6;7 8]; x=[9 10]; y=[11;12]; z=[13 14];
```

För att utvidga x till en 1 x 4-vektor som är [9 10 11 12] kan vi göra på olika sätt:

```
1)  >>x(3)=11; x(4)=12;
2)  >>x=[x 11 12]
3)  >>xtra=[11 12]; x=[x xtra];
```

För att lägga till en ny rad till A, t.ex. den rad som motsvarar vektorn z, finns följande alternativ:

```
1)  >>A=[A; 13 14];
2)  >>A=[A;z];
```

Vill man lägga till flera rader följer man samma mönster, dvs: `>>A=[A;x;0 0;z];`
För att slå ihop två matriser gör man på motsvarande sätt: `>>A=[A;B];`

På samma sätt kan vi utvidga en matris med en eller flera kolumner:

```
>>A=[A y];
>>u=[15;16]; A=[A u];
>>A=[A B];
```

3.7 KOLONNOTATION, GENERERING AV VEKTORER.

I MATLAB använder man kolon (:) för att definiera en följd av värden.

i:k	definierar en radvektor av värden från i till k med steget 1, dvs i, i+1, i+2, ...
i:j:k	definierar en radvektor av värden från i till k med steget j, dvs i, i+j, i+2j, ...

```
Ex: >>a=2:4;                  ger vektorn a = (2 3 4)
    >>b=7:-1:3                ger vektorn b = (7 6 5 4 3)
    >>x=-0.2:0.1:0.3          ger vektorn x = (-0.2 -0.1 0 0.1 0.2 0.3)
    >>a=0;b=2*pi;n=10;x=(a:(b-a)/n:b)';y=sin(x);[x y]   ger en sinustabell
```

3.8 ATT SKAPA DELMATRISER.

Tecknet kolon (:) kan även användas för att skapa delmatriser av en befintlig matris A:

`A(i,j)`	delmatrisen bestående av det enda elementet A_{ij}
`A(:,j)`	kolumn nr j i matrisen A
`A(i,:)`	rad nr i i matrisen A
`A(:)`	kolumnvektor med kolumnerna i A staplade på varandra
`A(:,j:k)`	delmatrisen av A bestående av kolumnerna j t o m k
`A(i:k,:)`	delmatrisen av A bestående av raderna i t o m k
`A(i:k,j:l)`	delmatrisen med elementen i rad i t o m k och kolumn j t o m l

4.3 REPETITIONSKOMMANDOT I MATLAB

MATLAB har två kommandon för repetition, nämligen for-sats och while-sats (EJ repeat-sats!).

4.3.1 FOR-KOMMANDOT

For-kommandot tillåter en satsgrupp att bli exekverad ett fixt, förutbestämt antal gånger. Formen för en sådan sats är

```
for  variabel = uttryck
     satsgrupp
end
```

och kan även skrivas på en rad (obs! kommatecken eller semikolon före end) :

```
for variabel = uttryck   satsgrupp,  end
```

Här är uttryck sådant att det tilldelar variabeln ett startvärde, definierar ett tillväxtvärde och definierar ett slutvärde. I de flesta fall använder man kolon-notation (se 3.7), t.ex. i:j:k eller i:k. MATLAB-satsen

```
for v=i:j:k   uppför sig som   for v:=i to k step j
for v=i:k     uppför sig som   for v:=i to k step 1
```

Ex.
```
>>s=0; tecken=1;
>>for i=1:20  s=s+tecken/(i*i);tecken=-tecken;end
```

Man kan även räkna upp de variabelvärden för vilka en slinga ska utföras:

```
for v=[i1,i2,.....,in]
```

Ex.
```
>>s=0;
>>for i=[1,2,5,10]  s=s+2^i; end
```

Det går också att skriva nästlade slingor:

```
for  variabel1=uttryck1
     satsgrupp1
     for  variabel2=uttryck2
          satsgrupp2
     end
     satsgrupp3
end
```

4.3.2 WHILE-KOMMANDOT

While-kommandot exekverar en satsgrupp så länge ett logiskt uttryck är sant. Formen för en while-sats är

```
while logiskt uttryck
      satsgrupp
end
```

eller, på en och samma rad: `while` logiskt uttryck satsgrupp, end

Ex. Beräkning av sqrt(2) med Newton-Raphsons metod

```
>>c=2;x0=c;differens=1;
>>while abs(differens)>1E-13 x1=(x0+c/x0)/2;differens=x1-x0;
  x0=x1,end
```

Ex. Beräkning av maskinnoggrannheten för en dator, dvs det minsta tal ε för vilket $1+\varepsilon \neq 1$.
För alla tal $u<\varepsilon$ gäller alltså $1+u=1$ på datorn.

```
>>ep=1;while 1+ep>1  ep=ep/2;end;ep=2*ep
```

5 GRAFIK

MATLAB har kommandon som medger två- och tredimensionell ritning. Kommandona är flexibla och enkla att använda. Skalning och markering av axlar görs automatiskt. Vid tredimensionell grafik får man se ett perspektiv av en rymdyta där skymda linjer är borttagna. Nivåkurvor kan också genereras enkelt. När grafisk utmatning ska göras, öppnas ett *grafikfönster* på skärmen. I de flesta system är *kommandofönstret* och *grafikfönstret* öppna samtidigt på skärmen.

5.1 TVÅDIMENSIONELLA PUNKTMÄNGDER OCH KURVOR.

5.1.1 KOMMANDOT PLOT

MATLAB kan rita ut en mängd av ordnade koordinatpar. Det enklaste kommandot som kan användas är kommandot `plot`, som tar emot olika antal parametrar beroende på hur många kurvor man vill rita och vilken markering varje kurva skall ges. När plot-kommandot utförs växlar skärmen automatiskt från kommandofönster till grafikfönster. För att återvända till kommandofönstret igen så trycker man bara på en godtycklig tangent eller klickar med musen i kommandofönstret.

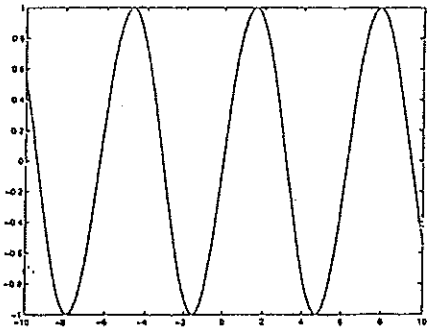
I den enklaste användningen av plot-kommandot ger man bara en vektor som indata. Antag att denna vektor innehåller n komponenter.

`plot(x)` ritar x-komponenternas värden mot index, dvs (i,x_i), $i=1,2,..,n$ ritas

Om x-koordinaterna ligger lagrade i vektorn x och y-koordinaterna i vektorn y fås en xy-plot genom kommandot

`plot(x,y)` ritar y som funktion av x

I bägge fallen fås en heldragen kurva, där punkterna sammanbinds med räta linjestycken.



Ex.
```
>>x=-10:0.2:10;
>>y=sin(x);
>>plot(x,y)
```

Om man inte vill ha en heldragen kurva finns olika typer av punkter, linjer och färgsättningar:

punkttyper	.	+	*	o	x	square	diamond
linjetyper	-	--	:	-.			
färger	t.ex. r (rött) b(blått) g(grönt) w(vitt)						

Några exempel på plotkommandon ges nedan. Låt t,x och y vara vektorer av samma längd, A och B matriser av samma storlek:

`plot(x)`	talparen (j,x_j) ritas
`plot(x,y)`	y plottas mot x, dvs talparen (x_j,y_j) ritas
`plot(x,y,'--')`	ritar kurvan streckad

`plot(x,y,'o')`	markerar punkterna med en ring
`plot(t,x,t,y)`	x ritas mot t och y plottas mot t, dvs två kurvor ritas
`plot(A)`	plottar kolumnerna i A mot radindex. Om A är en m x n-matris ritas n st kurvor med m st talpar i varje kurrva
`plot(x,A)`	plottar kolumnerna i A mot vektorn x. Antalet element i x måste vara lika med antalet rader i A
`plot(A,y)`	plottar vektorn y mot kolumnerna i A
`plot(A,B)`	plottar kolumnerna i B mot kolumnerna i A
`comet(x,y)`	ritar en animerad graf av y som funktion av x
`hold on`	ger ny bild ritad ovanpå gammal bild (axlar behålls)
`hold off`	ger ny bild i fräscht grafikfönster nästa gång plot anropas
`clf`	suddar grafikfönstret (även om hold gjorts)
`grid`	lägger till ett nät på befintlig plotbild
`errorbar(x,y,e)`	adderar felstaplar i en bild som ritats med `plot(x,y)`
`fill(x,y,c)`	ritar den polygon vars hörn definieras av vektorerna x och y; polygonen fylls med färgen som anges av c, t ex 'g' för grön
`drawnow`	framtvingar ritning; t ex kommer MATLAB vänta med att rita upp bilder från ritkommandon i en slinga tills slingan är klar om inte drawnow ges efter varje ritkommando
`area(x,y)`	fungerar som `plot`, men fyller utrymmet nedanför linjen med färg
`pie(x)`	ritar ett tårtdiagram av vektorn x.

När ingen punkttyp anges ritas heldragna linjer mellan punkterna. Om man vill ha en annan punkttyp, linjetyp eller färg än den som MATLAB ger anger man den i ett tredje argument, t.ex.

```
plot(x,y,'+')          plot(t,x,'r',t,y,'b')
```

Ex.
```
>>x=-6:0.2:6;
>>y=sin(x);plot(x,y);hold on
>>z=cos(x);plot(x,z);hold off
```
ger en graf av sinusfunktionen
ger en graf av cosinus ovanpå föregående graf

Ex.
```
>>plot(x,y,x,z)
```
ger samma graf som ovanstående exempel

Ex.
```
>>x=-6:0.2:6;
>>A=[sin(x);cos(x)];
>>plot(x,A)
```
ger samma graf som i exemplet ovan

Obs! Om `plot`-kommandot ligger inne i en slinga, kommer endast sista bilden att synas i grafikfönstret. Vill man se alla bilder en efter en måste man antingen lägga in `pause` efter `plot` (men före `end`) eller framtvinga ritning med `drawnow` (se ovan). Med `hold on` inne i slingan överlagras alla grafer i en bild. Glöm inte att göra `hold off` efter slingan om fler grafer skall ritas efteråt!

5.1.2. FLERA FÖNSTER PÅ SKÄRMEN

Med plot-kommandot får man alltså en bild av ett koordinatsystem med godtyckligt många kurvor plottade samtidigt. Vill man ha flera bilder samtidigt i grafikfönstret finns kommandot subplot:

`subplot(m,n,p)`	där m,n och p är tre heltal, delar upp grafikfönstret i ett m x n nät av små delfönster och väljer ut fönster nr p som aktuellt plotfönster. Delfönstren numreras från vänster till höger, uppifrån och ner. m<=9, n <=9.
`subplot`	återställer till det normala fallet, dvs ett fönster, nästa gång plot anropas

6 PROGRAMMERING I MATLAB

Normalt arbetar MATLAB som en interpretator, dvs MATLAB tolkar och utför ett kommando så snart kommandot är givet och man tryckt på Enter. Detta arbetssätt lämpar sig för mindre beräkningar, men blir klumpigt när antalet kommandon som behövs för att utföra en uppgift blir stort. Dessutom går de kommandon som man givit under en session förlorade när man lämnar MATLAB. I MATLAB finns dock möjligheten att spara de kommandon som behövs för att lösa en beräkningsuppgift på en fil och exekvera denna. Genom denna facilitet kan MATLAB även användas som ett högnivåprogramspråk, och de kommandofiler som byggs upp kan ses som MATLAB-program.

6.1 MATLABS KOMMANDOFILER.

Det man bör göra för lite större beräkningsuppgifter är att lagra kommandon på en *extern fil*, en s.k. *m-fil* eller *kommandofil* (även kallad *script-fil*), som kan läsas in och exekveras i MATLAB som ett MATLAB-program. En m-fil är uppbyggd av MATLAB-kommandon, men kan även referera till andra m-filer, t ex en m-fil som innehåller en funktion (se 6.3). Tillvägagångssättet vid användning av en m-fil är följande:

1) skriv in de MATLAB-kommandon som bygger upp m-filen *med någon editor* som finns tillgänglig på den dator som du kör på och spara den sedan på en fil med namnet `filnamn.m`, där `filnamn` är ett namn som du hittar på själv
2) återgå till MATLAB.
3) exekvera satserna i filen filnamn.m med kommandot `filnamn`. (EJ filnamn.m)

När man kör en m-fil exekveras genomlöps alla kommandon från början till slut. Exekveringen avbryts dock om m-filen innehåller input-, pause- eller keyboard-kommandon (se 6.3.4). Efter det att sista kommandot i m-filen lästs och utförts är MATLAB redo att ta emot nya kommandon från tangentbordet igen, dvs kontrollen återgår till tangentbordet.
Det är lämpligt att stoppa in kommentarer i sitt MATLAB-program precis som i andra programspråk. En kommentar skrivs i MATLAB genom att %-tecken skrivs framför den kommenterande texten.
Ett bra tips är att alltid börja ett MATLAB-program med satserna `clear, clf, hold off`

Ex. Antag att vi skrivit in kommandona i Ex. i 4.3.1 på en fil som vi kallat `serie.m`
Exekveringen av den startas genom att ge kommandot

```
>>serie
  s=                          utskriften på skärmen blir denna
     0.8213
>>                            MATLAB kan nu ta emot ett nytt kommando
```

Med `echo on` och `echo off` i en kommandofil kan man koppla på resp koppla från utskrift av filens kommandon under exekvering.

6.2 IN/UTMATNING TILL/FRÅN KOMMANDOFILER.

Exekveringen av en m-fil avbryts när ett input-, pause- eller keyboard-kommando påträffas bland kommandona i filen. Exekveringen återupptas då man matat in indata och/eller avslutat motsvarande kommando.

`input('textsträng')`	skriver ut texten textsträng och väntar på inmatning från tangentbordet. Exekveringen återupptas när man matat in indata och tryckt på Enter.
`disp('textsträng')`	skriver ut texten och fortsätter sedan exekveringen. Lämpligt att använda om man vill ha förklarande text insatt i den utmatning som görs av m-filen

`disp(a)`	skriver ut a utan texten a=
`pause`	medför att exekveringen stannar tills dess man tryckt ned en godtycklig tangent
`pause(2)`	åstadkommer en två sekunder lång paus i exekveringen
`keyboard`	anropar tangentbordet som om det vore en kommandofil. Exekveringen avbryts och kontrollen övergår till tangentbordet. Variabler kan skrivas ut och ändras, och ett godtyckligt MATLAB-kommando kan ges. Man lämnar tangentbordskontrollen genom att ge kommandot `return` (med bokstäver) varvid kontrollen övergår till m-filen.

Några exempel:

Kommando	Ger vid exekvering

`>>x=input('Ge talet x:  ')`	`Ge talet x:  2.0944`
	
	`x=`
	`  2.0944`

`>>A=input('Ge matrisen A radvis ')`	`Ge matrisen A radvis:[1 2;3 5]`
	`A=`
	`  1  2`
	`  3  5`

`>>disp('Nu startar iterationerna')`	`Nu startar iterationerna`

`>>disp('Ange matrisen A')`
`>>keyboard`
	`Ange matrisen A`
	nu kan MATLAB-kommandon ges, tex
	`>>A=[1 2;3 5]; return`

6.3 FUNKTIONSFILER.

6.3.1 ALLMÄNT OM FUNKTIONSFILER

Uppbyggnad och användning av funktioner i MATLAB görs analogt med deklaration och anrop av funktioner/procedurer (subrutiner) i andra programspråk. I MATLAB finns dock inte procedurbegreppet utan endast funktioner. I MATLAB lagras varje funktion på en egen m-fil, som sedan kan anropas med ett MATLAB-kommando. En funktion har ett antal indata-parametrar, som före anropet skall ha fått sina värden, och ett antal utdata-parametrar, som får sina värden när funktionen anropas. I MATLAB finns ett stort antal inbyggda funktioner, t.ex. en funktion för LR-faktorisering, vilken anropas som följande exempel visar:

Ex. `>>[L,R]=lu(A);` A är indata, L och R är utdata

För att se vilka funktioner som finns ger man `help`-kommandot och för att få mer information om en särskild funktion, t.ex. `lu`, så ger man kommandot `>>help lu`

Funktionsfiler har separat arbetsarea som innehåller de variabler som funktionen använder då den exekveras. Variablerna i en funktionsfil är lokala variabler och existerar endast under exekveringen av funktionen och går sedan förlorade. Variabler i en funktionsfil som heter precis samma sak som variabler i MATLABs arbetsarea är alltså *olika* variabler i olika minnesutrymmen.

INLÄMNINGSUPPGIFT 2

Syfte

1) Att lära sig använda Simulink för simulering, analys och dimensionering av återkopplade system.
2) Att ge en fördjupad förståelse för dynamiska och återkopplade system.

Förberedelser

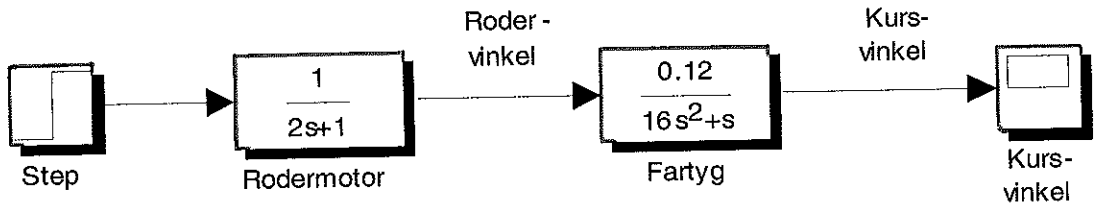
Läs kapitel 4 i läroboken, speciellt avsnitten om P, PI och PID-reglering.
Läs kapitel 24 i läroboken, speciellt avsnittet om Simulink.

Reglering av kursvinkeln på ett fartyg.

Uppgiften löses helt och hållet med Simulink.
Som redovisning av uppgifterna 2.1-2.6 nedan vill vi ha följande:

- Lämna in en lista med numeriskt svar på alla efterfrågade sifferuppgifter.
- Lämna in en figur på din egen simulinkmodell i uppgift 2.6, samt kurvor på motsvarande simulering (rodervinkel och kursvinkel).
- Skriv upp med egna ord upp vilka slutsatser och lärdomar du dragit av dessa uppgifter, (t ex fördelen med derivataverkan, inverkan av styrsignalbegränsningar mm).

I denna uppgift ska vi studera ett system för reglering av en kursvinkeln på ett fartyg. Fartyget inklusive styrdon (rodermotor) kan approximativt beskrivas som en process med integration och två tidskonstanter enligt nedanstående blockschema. Parametrarna nedan (tidskonstanter mm) är realistiska för ett medelstort lastfartyg. Uppgiftens syfte är att ge en första introduktion till användandet av programmet SIMULINK för dimensionering och simulering av reglersystem, samt att studera skillnaden mellan P-reglering och PD-reglering:



2.1 Fartyget utan regulator

Bygg upp ovanstående blockschema i Simulink. Studera därefter stegsvaret (d v s kursvinkeln som funktion av tiden) för själva processen (fartyget) utan något återkopplat reglersystem. Tidskonstanterna 2 och 16 mäts i sekunder. **Prova dig fram till lämplig simuleringstid.** Alla variabler mäts i grader. Kontrollera om stegsvaret verkar rimligt.

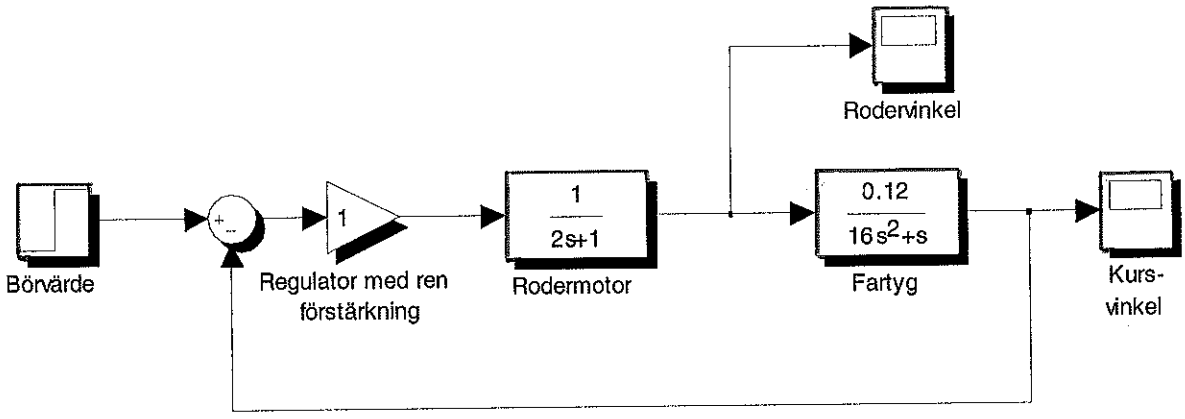
a) Hur många grader har fartyget vridit sig på en halv minut (ungefär) om man lägger på en rodervinkel på 30 grader?

b) Hur lång tid (ungefär) tar det för fartyget att vrida sig 90 grader i detta fall?

Svar: a) ..... b) .....

2.2 Fartyget med P-regulator

Vi ska studera insvängningsförloppet vid börvärdesändringar om vi reglerar kursvinkeln med en P-regulator som har förstärkningen K=1. Koppla därför upp följande blockschema:



Hur ser insvängningsförloppet ut för ovanstående reglersystem? Studera börvärdesändringar på 30 grader.

Hur stor blir översvängen (mäts i procent) ? .....

Snabbhet: Hur lång tid tar det för utsignalen att gå från 0 till 30 grader (till den första skärningspunkten)?.....

Hur stor blir det maximala roderutslaget under insvängningsförloppet? .....

2.3 Variation av förstärkningen

Översvängen blir, som synes, alldeles för stor om vi använder P-reglering med förstärkningen K=1. Vilken förstärkning ska vi välja om vi vill att översvängen ska vara mindre än 10%? (Prova dig fram för att approximativt lösa denna uppgift.)

Svar:.....

Hur lång tid tar det nu för utsignalen att gå från 0 till 30 grader? .....

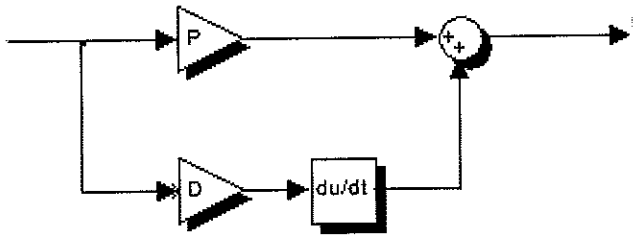
Hur stort blir nu det maximala roderutslaget under insvängningsförloppet? .....

2.4 Reglering med PD-regulator

Som synes måste vi välja en mycket låg förstärkning K om översvängen ska vara mindre än 10%. Detta ger dock en mycket långsam reglering. Enbart P-reglering ger alltså inga bra egenskaper. Vi ska därför prova med en PD-regulator med nedanstående överföringsfunktion.

$$G_{PD}(s) = P + Ds$$

Överföringsfunktionen ovan motsvarar en PD-regulator med förstärkningen K = P och derivatatiden T_D = D/P. PD-regulatorn kan i Simulink byggas enligt nedanstående blockschema.



Byt ut P-regulatorn i reglersystemet mot en sådan PD-regulator. Ett rimligt värde D är D=15. Antag att detta värde används på konstanten D. Prova dig i så fall fram till vilket P-värde du nu behöver för att få en översväng på ca 10%.

Svar P =.....

Hur lång tid tar det nu för utsignalen att gå från 0 till 30 grader?.....

Hur stort blir nu det maximala roderutslaget under insvängningsförloppet? .....

2.5 Ytterligare försök

Som synes fick vi ett ganska bra resultat med ovanstående PD-regulator med klart snabbare reglering än vid P-reglering. Vi ska dock se om vi kan få ännu bättre snabbhet genom att öka derivataverkan ytterligare. Prova därför med värdet D=40. Vilket P-värde behöver i så fall för att få en översväng på ca 10%?

Svar P =.....

Hur lång tid tar det nu för utsignalen att gå från 0 till 30 grader?.....

Hur stor blir nu det maximala roderutslaget under insvängningsförloppet? .....

2.6 Mättad styrsignal

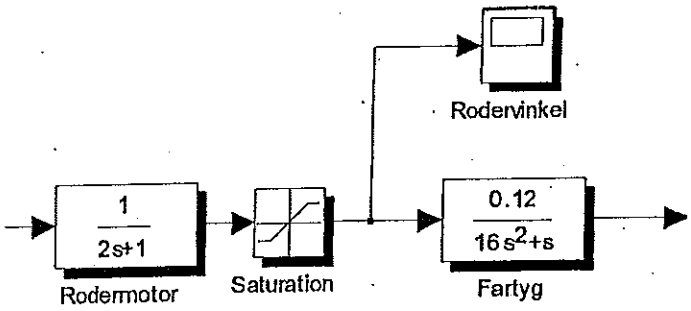
Det sista försöket ger ett mycket snabbt och bra insvängningsförlopp för kursvinkeln vilket är jättebra. Emellertid ser vi att det maximala roderutslaget under insvängningsförloppet blir större än vad rodret på ett fartyg kan klara. (På många fartyg kan man inte ställa ut rodervinklar över +-60 grader, ibland bara 45 grader). Eftersom rodervinkeln kommer att "slå i taket" är resultatet av simuleringen orealistiskt.

Gör följande för att få realistiska simuleringar: Komplettera blockschemat för rodermotorn med en styrsignalmättning (saturation) enligt nedanstående figur. Mättningen gör att rodervinkeln till fartyget under simuleringen aldrig blir mer än 60 grader även om regulatorn ibland vill ha ett större värde. Styrsignalmättningen (eng. saturation) är ett olineärt block som finns i underbiblioteket "Nonlinear" i Simulink. Ställ in gränserna +/-60 grader i blockets dialogruta.

Bibehåll tidigare värden på P och D. Studera därefter hur insvängningsförloppet nu ser ut för kursvinkel och rodervinkel vid börvärdesändringar.

Hur lång tid tar det nu att gå från 0 till 30 grader?.....

Hur stor blir översvängen i detta fall?.....



Som redovisning av uppgifterna 2.1-2.6 nedan vill vi ha följande:

- Lämna in en lista med numeriskt svar på alla efterfrågade sifferuppgifter.
- Lämna in en figur på din egen totala simulinkmodell i uppgift 2.6, samt kurvor på motsvarande simulering (rodervinkel och kursvinkel).
- Skriv upp med egna ord upp vilka slutsatser och lärdomar du dragit av dessa uppgifter, (t ex fördelen med derivataverkan, inverkan av styrsignalbegränsningar mm).

Studium av PI- och PID-reglering, processer med dödtid mm.

Uppgift 2 löses helt och hållet med Simulink.
Som redovisning av uppgift 2.7-2.13 vill vi ha följande:

- Lämna in numeriskt svar på alla efterfrågade sifferuppgifter.
- Lämna in en figur på din simulinkmodell i uppgift 2.13.
- Skriv upp med egna ord upp vilka slutsatser och lärdomar du dragit av denna uppgift, bl a inverkan av dödtid, fördel med integrerande verkan mm.

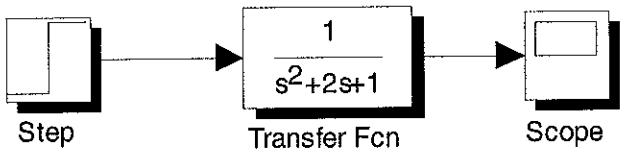
I denna uppgift ska vi gå vidare och studera fördelarna med integrerande verkan i en regulator, vi ska också studera hur insvängningsförloppet ser ut vid störningar samt studera vilken inverkan dödtid har ett system.

2.7 Process med två tidskonstanter

Vi ska börja med att studera en relativt "lättreglerad" process, nämligen en process med två reella tidskonstanter. Som du säkert vet har en sådan process ett monotont s-format stegsvar (se läroboken). Det finns många processer som kan beskrivas med denna typ av överföringsfunktion, t ex ugnsprocesser, kemiska processer, elmotorer mm. (Se exemplen i läroboken) För enkelhets skull har båda tidskonstanter storleken 1 sekund:

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

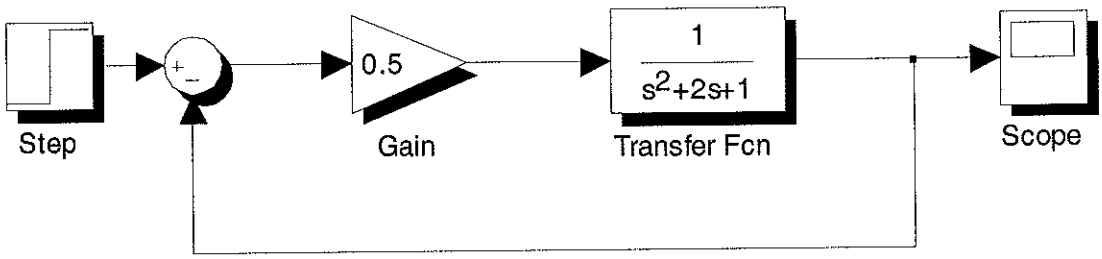
Bygg upp processens blockschema med hjälp av simulink enligt nedan.



Simulera processens stegsvar och kontrollera att det har samma form som väntat. Observera att scope-fönstret efter en simulering kan förstoras och att olika delar av kurvan kan zoomas in med hjälp av verktygen på verktygsraden. Simulera också hur processen reagerar på sinusformade insignaler och rampformade insignaler.

2.8 P-reglering

Processen ovan skall nu regleras med en P-regulator. Bygg därför upp följande blockschema på skärmen. Nu kan du studera hur det återkopplade systemets stegsvar (vid en börvärdesändring) påverkas av förstärkningen.



Prova följande värden på förstärkningen i P-regulatorn: 2, 8 och 20. Undersök hur den statiska noggrannheten (dvs det kvarstående felet), stabiliteten (storleken på översvängen) och snabbheten påverkas av förstärkningen. Storleken på översvängen mäts i procent av slutnivån. Alla siffror bestäms approximativt direkt ur kurvorna. Observera att du kan zooma in speciella delar av kurvan för att få noggrannare avläsning och att du sedan kan zooma ut igen med kikar-symbolen. **(OBS! Glöm inte att sätta step-time till 0 i stegfunktionsblocket, så att steget kommer vid tidpunkten noll)**

Förstärk-ning	Slut nivå	Kvarst. fel e_0	Storlek på översväng (%)	Tid för max översväng (sek)
2				
8				
20				

Vilken förstärkning ger bäst stabilitet ?.....

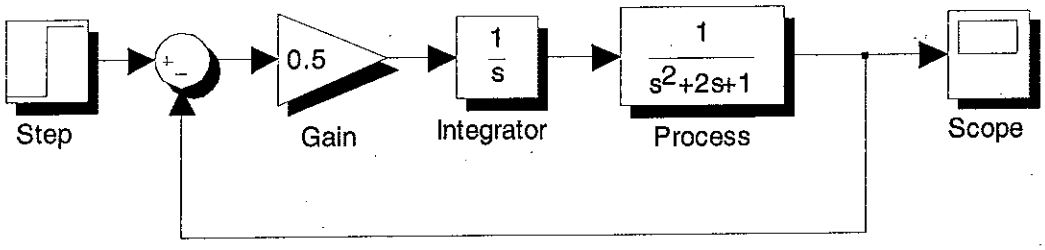
Vilken förstärkning ger minst kvarstående fel ?.....

Vilken förstärkning ger bäst snabbhet ?.....

Bestäm vilken förstärkning som behövs om det kvarstående
felet ska vara mindre än 0,1. (Bestäms approximativt) .....

2.9 I-reglering

Som väntat fick vi ett kvarstående fel då vi reglerade med en P-regulator (se t ex läroboken kap 4). För att få bort det kvarstående felet ska vi byta ut P-regulatorn mot en I-regulator. Koppla därför upp följande blockschema:



Prova först med en integrationskonstant K_I = 0,5 enligt ovan (observera att integrationskonstanten är inversen av integrationstiden, dvs $K_I = 1/T_I$). Kontrollera att det kvarstående felet vid stegformade börvärdesändringar försvinner då vi använder en I-regulator. (Öka först simuleringstiden till 20 sekunder).

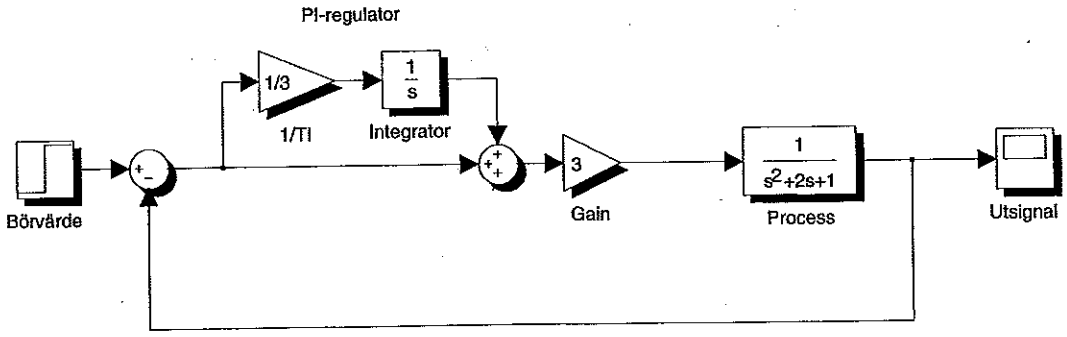
Undersök vilket värde integrationskonstanten K_I skall ha om översvängen inte får överstiga 10%. Svar: K_I =.....

I-regulatorn har fördelen att den helt eliminerar det kvarstående felet. Samtidigt har den nackdelen att den arbetar långsamt. Stigtiden blir avsevärt längre med I-reglering än med P-reglering. Hur lång blir stigtiden för systemet med ovanstående värde på integrationskonstanten ? (Bestämmes approximativt ur kurvan)

Svar: Stigtiden är t_r =.....

2.10 PI-reglering

Genom att kombinera en P-regulator och en I-regulator kan man få ett system som eliminerar de kvarstående felen utan att snabbheten går förlorad. Undersök detta genom att koppla upp följande blockschema:

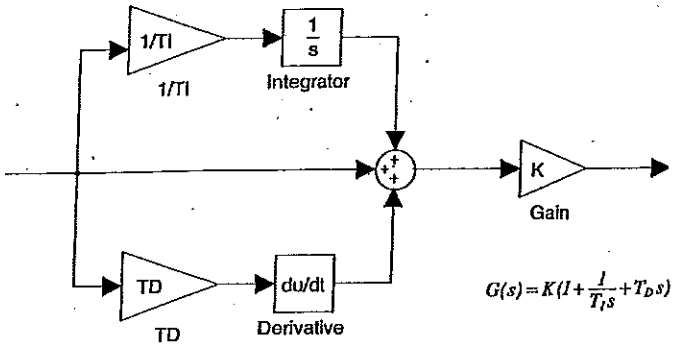


I ovanstående blockschema har vi (som synes) använt en PI-regulator med förstärkningen K = 3 och integrationstiden T_I = 3. Parametrarna har beräknats med en känd tumregelmetod. Hur lång blir stigtiden och översvängen med ovanstående regulator? (Bestämmes approximativt ur den simulerade kurvan)

Stigtid = ... Översväng = ...

2.11 PID-reglering

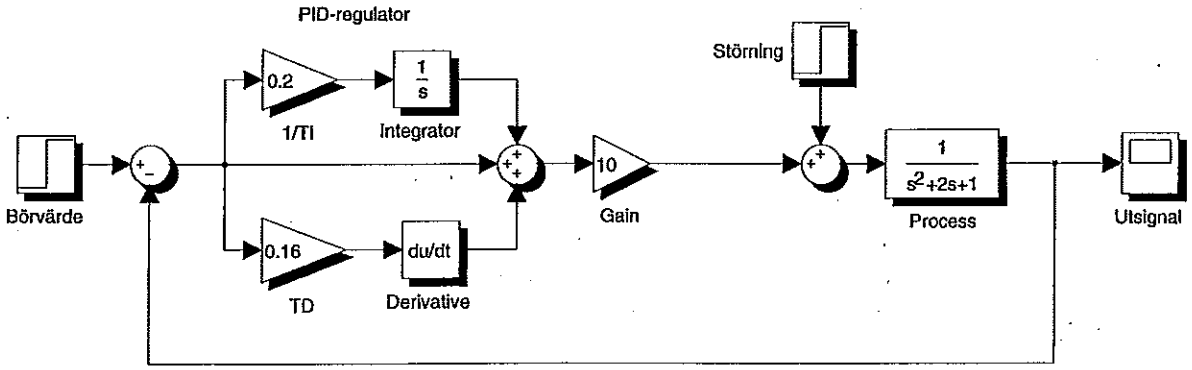
För att få ännu bättre prestanda ska vi byta ut PI-regulatorn mot en PID-regulator enligt nedanstående blockschema. Prova t ex följande värden på PID-parametrarna:
K = 10, TI = 5 och TD = 0,16. I just detta fall blir det ingen avgörande förbättring då D-verkan används, i andra fall kan derivataverkan dock, som du redan sett, ha mycket stor betydelse för egenskaperna hos ett reglersystem.



Stigtid = ... Översväng = ...

2.12 Process med störningar

Ett reglersystem ska normalt inte bara reagera på börvärdesändringar. Det är också viktigt att systemet kan eliminera de störningar som verkar på processen. Vi skall undersöka detta genom att bygga ut blockschemat enligt nedan, där vi lagt till ett block för att simulera hur systemet reagerar på stegformade störningar.

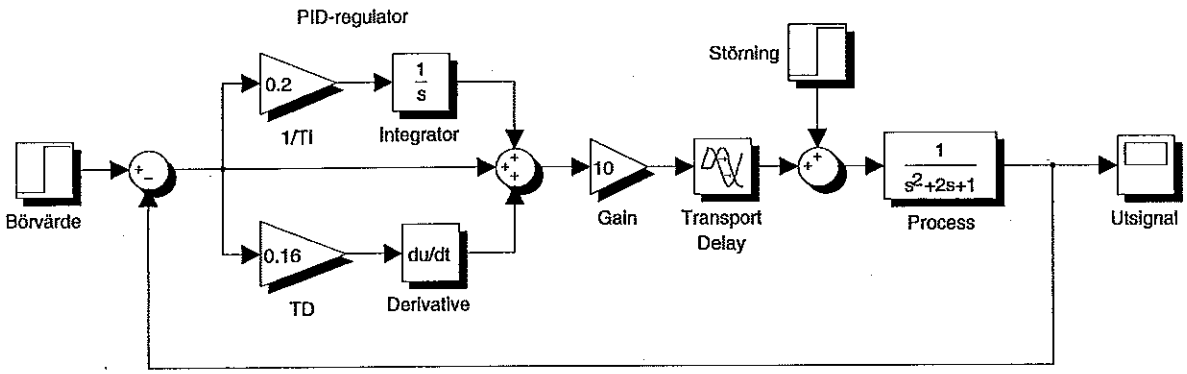


Undersök hur stort maxfelet blir vid en stegstörning på 2 enheter. Sätt störningens "final value" till 2. **(Nollställ först börvärdesändringen. Dvs sätt dess "final value" lika med noll.** Vi antar alltså att börvärdet är noll hela tiden.).

Maxfel vid störning = ...

2.13 Process med dödtid

Som du vet är det svårt att reglera processer med dödtid. Detta beror på att dödtiden fördröjer signalerna i loopen och att de ger ett kraftigt negativt bidrag till faskurvan, som gör att stabilitetsmarginalerna minskar. Vi ska därför undersöka vad som händer om vi kopplar in en dödtid i slingan. (Dödtiden kan t ex motsvara att givaren tar en viss tid på sig för att mäta utsignalen, eller att beräkningstiden i regulatorn inte är försumbar. Många processer har också egen dödtid.). Koppla upp nedanstående blockschema.



Använd samma PID-parametrar som tidigare. Undersök hur stabiliteten (översvängen vid börvärdesändringar) förändras då dödtiden ökas successivt från 0,1 till 0,4 sekunder (i steg om 0,02). Vid vilket ungefärligt värde på dödtiden blir systemet instabilt (dvs utsignalen börjar självsvänga med ökande amplitud vid börvärdesändringar)?

OBSERVERA FÖRST: I detta fall kan noggrannheten i simuleringen eventuellt förbättras genom att under menyvalet Simulation parameters/solver options använda metoden ode23 (Bogacki-Shampine) i stället för ode45 (som är default) och samtidigt minska relative tolerance till 1 e-4.

Svar: Dödtid vid instabilitet = ...

Som redovisning av uppgift 2.7-2.13 vill vi ha följande:

- Lämna in numeriskt svar på alla efterfrågade sifferuppgifter.
- Lämna in en figur på din kompletta simulinkmodell i uppgift 2.13.
- Skriv upp med egna ord upp vilka slutsatser och lärdomar du dragit av denna uppgift, bl a inverkan av dödtid, fördel med integrerande verkan mm.

INLÄMNINGSUPPGIFT 3

Syfte

Syftet med uppgiften är följande
1) Att fördjupa sig i användningen av datorbaserade hjälpmedel för simulering, analys och dimensionering av återkopplade system. Speciellt använder vi programmet Matlab med tillhörande toolboxar (Matlab Control Toolbox och Simulink).
2) Att ge fördjupade kunskaper om digital reglering, modellbaserad dimensionering, modellering och tillståndsmodeller.

Förberedelser

Uppgifterna kräver att du i förväg studerat relaterade sidor i kursboken. Observera att vissa uppgifter ska lösas för hand innan du sätter dig vid datorn. Några uppgifter löses helt och hållet för hand.

Redovisning

Redovisning ska inlämnas i enlighet med instruktioner i detta häfte och i kurs-PM. Rapporten skall vara fullständig, tydlig och välgjord.

Uppgifterna löses i grupper om två teknologer per grupp. I vissa uppgifter används olika parametrar för olika grupper. Parameterlista för olika grupper finns sist i häftet.

Uppgifterna ska lösas med intresse och eftertanke, som om du själv var anställd på företaget som ska utveckla systemet. (Annars lär man sig inget)

Lycka till.

3. Digitalt reglersystem för en traverskran.

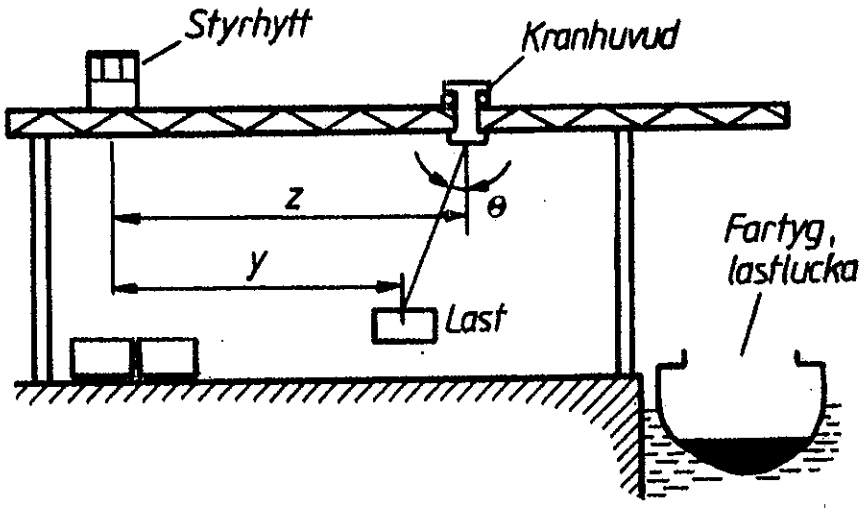
Uppgiften bygger speciellt på följande teori i upplaga 4 av Modern Reglerteknik:

- Stegsvarsanalys, kap 7.
- Modellbaserad dimensionering, kap 12.
- Transformering av analoga överföringsfunktioner, kap 19.

För uppgifter där kurvor mm ska tas fram används Simulink.

Alla beräkningar med tillhörande resultat (på uppgift 1-4) samt alla framtagna kurvor skall ingå i en tydlig och välgjord rapport tillsammans med lämpliga kommentarer och slutsatser.

Antag att Du blivit anställd på ett företag som är specialister på tillverkning och konstruktion av traverskranar i olika storlekar. Företagets kunder är bl a godscentraler på järnvägsstationer, hamnförvaltningar, varulager, pappersbruk, stålverk och andra företag där man hanterar stora mängder containrar och gods. Nu har du fått i uppgift att dimensionera ett datorbaserat intelligent reglersystem för en äldre traverskran, som finns i hamnen i Oostende (i Belgien). Traverskranen används idag för att lasta lådor och containrar i och ur fartyg, se figur.



Det traditionella förfaringssättet vid lastning av en container är att kranföraren manuellt "kör" lasten fram till lastluckan på fartyget, varifrån den sänks ned i lastrummet. Ett problem är dock att lasten lätt börjar pendla då man bromsar in strax före lastluckan. Detta leder i sin tur till följande två problem:

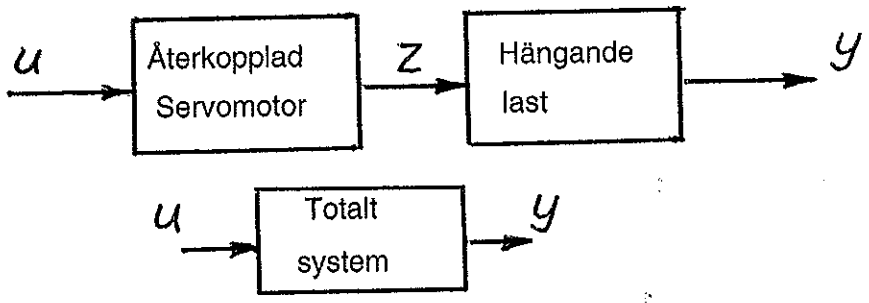
1) Det tar längre tid att lasta fartyget eftersom man måste vänta på att pendlingarna "dör ut" innan man kan sänka ned lasten i lastluckan (som ofta är trång). Detta är ett stort problem eftersom fartygen kostar mycket pengar även då de står stilla.

2) Containern kan stöta i kanten på lastluckan och därmed skada fartyget eller innehållet i containern.

För att komma till rätta med problemet skall man konstruera ett digitalt reglersystem som eliminerar lastens pendlingar. Det nya reglersystemet skall vara utformat så att traversföraren direkt via ett tangentbord kan mata in önskad position på lasten (antal meter från utgångspunkten) varefter reglersystemet automatiskt ska se till att lasten flyttar sig till den önskade positionen. Reglersystemet ska se till att insvängningen till den nya positionen sker utan några tendenser till svängningar.

Dagens system

Med dagens system har traversföraren en styrspak, med vilken han matar in önskad position på kranhuvudet. Genom att flytta kranhuvudet (som i sin tur innehåller en återkopplad servomotor) kan man i sin tur påverka läget på lasten. Styrsignalen u från styrspaken till kranhuvudets servomotor är en spänningssignal (0-10 V), som direkt motsvarar positionen på kranhuvudet i antal meter från utgångspunkten. Om man t ex ger signalen $u = 5$ V till kranhuvudet kommer den, (efter en viss insvängningstid), att ställa sig i positionen $z = 5$ meter från utgångspunkten.



I figuren gäller att

u = styrsignal (V)

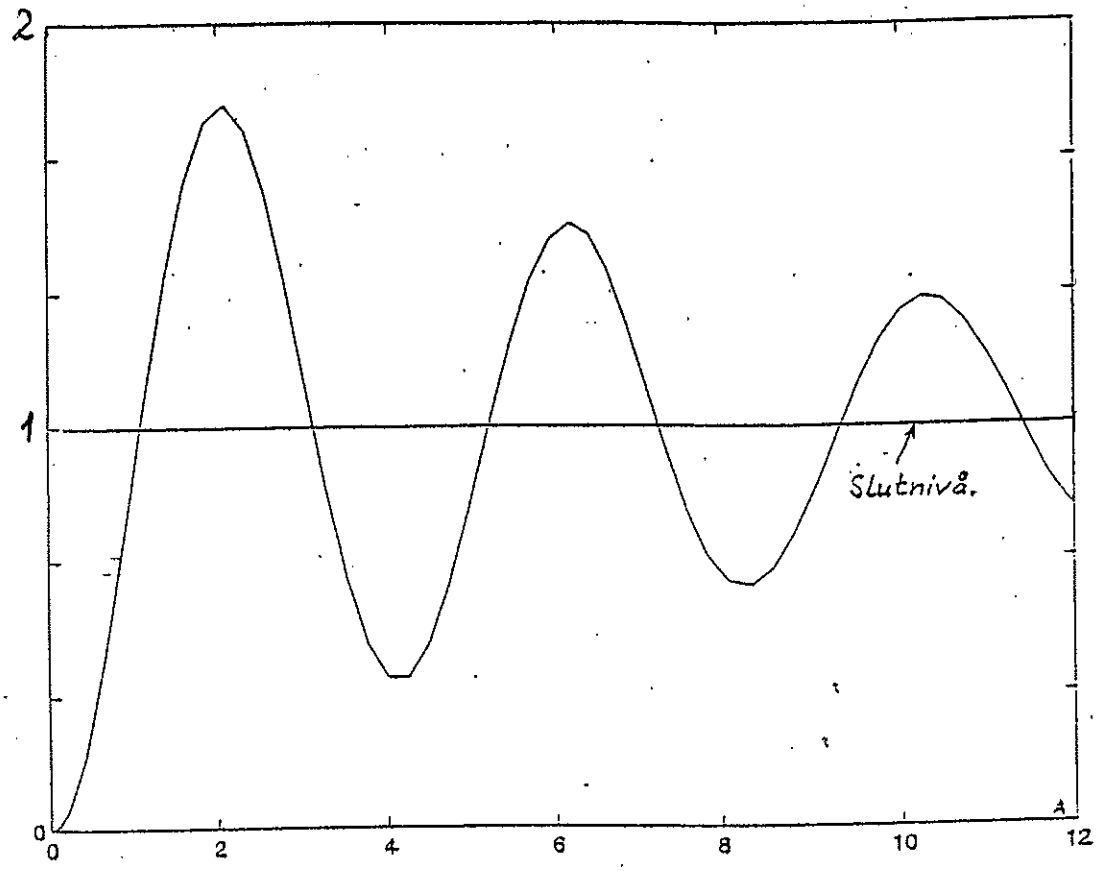
z = kranhuvudets position (m)

y = lastens position (m)

Figur på dagens system

För att bestämma överföringsfunktionen för hela processen (från u till y) har man mätt upp processens stegsvar. Stegsvarsexperimenten har tillgått på följande sätt:

Man har ändrat spänningen till kranhuvudets servomotor u stegformat från 0 till 1 V vid tidpunkten noll. Därefter har man med ett optiskt mätsystem mätt upp hur snabbt lasten förflyttat sig som funktion av tiden, dvs $y(t)$. Resultatet visas i nedanstående figur. Som väntat är detta en process med stor översväng.



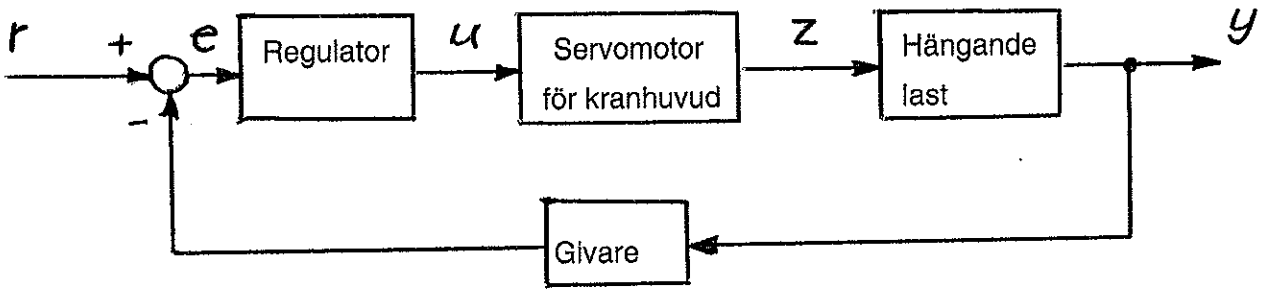
Uppgifter:

3.1

Bestäm den analoga överföringsfunktionen för processen med hjälp av de formler som ges i avsnittet om stegsvarsanalys i läroboken. Använd en andra ordningens överföringsfunktion med komplexa poler. I efterhand ska Du med Simulink eller Matlab kontrollera att den överföringsfunktion du räknat fram har exakt samma stegsvar som ovan.

3.2

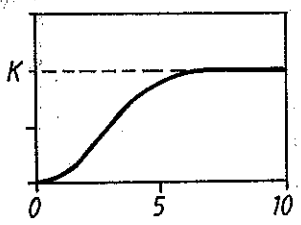
Dimensionera ett återkopplat reglersystem för processen enligt följande specifikation: Börvärdet r skall utgöras av önskad position på **lasten**, mätt i meter från utgångsläget. Ärvärdet är aktuell position på lasten y. Vid en stegformad börvärdesändring r skall reglersystemet se till att lasten flyttas från startläget till slutläget utan några pendlingar. Det förutsätts att lastens position y kan mätas med hjälp av ett optiskt system. Se figuren nedan



Figur på system att bygga

Använd följande metod för att dimensionera regulatorn: Beräkna först en analog regulator G_a till processen med hjälp av formler för **modellbaserad reglering** (kap 12.4 i läroboken). Det totala systemet skall ha nedanstående överföringsfunktion, som motsvarar ett monotont insvängningsförlopp utan översväng. Parametern T ges av parameterlistan längst bak i häftet. Kontrollera i efterskott med Simulink eller Matlab att det totala återkopplade reglersystemet verkligen får ett stegsvar av önskad typ.

$$G_{TOT} = \frac{Y}{R} = \frac{1}{(1+Ts)^2}$$



3.3

Skriv om den analoga överföringsfunktionen för regulatorn G_a på tidsdiskret form med hjälp av **bilineär transform**. Använd samplingsintervallet h = 0,2 sekunder. Redovisa hur den tidsdiskreta regulatorn ser ut. Transformeringen kan göras för hand eller med Matlab.

Använd simulink för att (i efterskott) ta fram kurvor på det totala reglersystemets insvängningsförlopp vid en stegformad börvärdesändring på 1 m. Tag dels fram kurvor som visar hur utsignalen y svänger in sig vid en stegformad börvärdesändring, dels kurvor på hur (den styckvis konstanta) styrsignalen u ser ut vid samma typ av börvärdesändring. Fundera en stund på om kurvorna verkar rimliga.

Ledtrådar

Uppgift 3.2 Uppgiften kan lösas för hand eller med Matlab. Om du räknar med Matlab ska du komma ihåg att förkorta bort gemensamma faktorer i täljaren och nämnaren med hjälp av Matlab-kommandot minreal. Annars kan det bli problem vid diskretiseringen.

Formel för modellbaserad reglering: **GR=GT / (GP*(1-GT))**

Förkortning av gemensamma faktorer **GR=minreal(GR)**

Uppgift 3.3 Uppgiften kan göras för hand eller med Matlab. Om matlab används så görs diskretiseringen med kommandot c2d (continous to discrete) enligt följande:

HR=c2d(GR,0.2,'tustin')

Glöm ej att sätta samplingsiden till 0,2 i Simulink, annars blir det helt fel.

3.4

Skriv om överföringsfunktionen för den tidsdiskreta regulatorn till en **differensekvation**, dvs till en ekvation som direkt kan programmeras in i ett datorbaserat styrsystem för traverskranen. Differensekvationen skall visa hur styrsignalen u från regulatorn varje samplingsintervall skall beräknas utgående från aktuella och tidigare felsignalvärden e samt tidigare värden på styrsignalen u.

Parameterlista

Använd nedanstående lista för att bestämma vilken storlek på konstanten T din grupp ska använda. Listan är i alfabetisk ordning. Välj efter första bokstaven på ditt efternamn. Om ni är flera i gruppen, så välj den parameter som motsvarar det efternamn som kommer **sist** i alfabetisk ordning.

A	1,07	H	1,05	O	1,03	V	1,07
B	1,12	I	1,1	P	1,15	X	1,12
C	1,18	J	1,15	Q	1,23	Y	1,17
D	1,31	K	1,2	R	1,41	Z	1,22
E	0,9	L	1,25	S	0,92	Å	1,27
F	0,95	M	1,3	T	0,97	Ä	1,32
G	1,0	N	1,35	U	1,02	Ö	1,37

Några kommentarer

Även om uppgiften ovan är realistisk till sin karaktär har den på flera sätt förenklats, för att räkningarna inte ska bli för omfattande. Några av de förenklingar vi gjort är följande:

1) Vi har inte tagit hänsyn till att periodtiden för en pendel beror på hur lång den är. Stegsvaret ovan gäller alltså endast vid en viss längd på pendeln, d v s för ett visst avstånd från kranhuvudet till tyngdpunkten på lasten. Om detta avstånd varierar påverkas också överföringsfunktionen, vilket man då får ta hänsyn till i regulatoralgoritmen.

2) Stegsvaret ovan gäller endast för relativt små lastförflyttningar. Vid längre lastförflyttningar kommer servomotorn i kranhuvudet att uppnå sin maximala hastighet i början av insvängningen, vilket gör att det tar längre tid att nå målet. Detta är alltså en olinear begränsning som vi inte tagit hänsyn till (men som heller inte är svår att ta hänsyn till).

3) Stegsvaret ovan motsvarar en andra ordningens överföringsfunktion. I själva verket motsvarar denna andra ordningens överföringsfunktion endast själva *pendeln*, dvs sambandet mellan positionen på kranhuvudet z och positionen på lasten y (vilket kan visas med Newtons lagar). Vid en mer noggrann analys bör man också ta med dynamiken i servomotorn, dvs sambandet mellan u och z, som normalt representeras av en egen överföringsfunktion av första eller andra ordningen. Den totala överföringsfunktionen från u till y blir därför i verkligheten snarare av tredje eller fjärde ordningen än av andra ordningen.

För att bygga ett reglersystem som i praktiken kan användas positionsregleringen av en traverskran fordras därför en något större arbetsinsats än vad ovanstående uppgift krävt. Om pendeln har varierande längd fordras en regulator som kan ändra reglerparametrar beroende på linlängden (s k parameterstyrning). Vidare kan man inte bortse ifrån att överföringsfunktionen från u till z i praktiken är av högre ordning än vad vi förutsatt i denna uppgift. (se även bifogad artikel ur Ny Teknik, 98)

Gungande last går att styra

Forskare har löst problemen med pendelrörelser under kranar

AV HÅKAN ABRAHAMSON
NY TEKNIK/HANNOVER

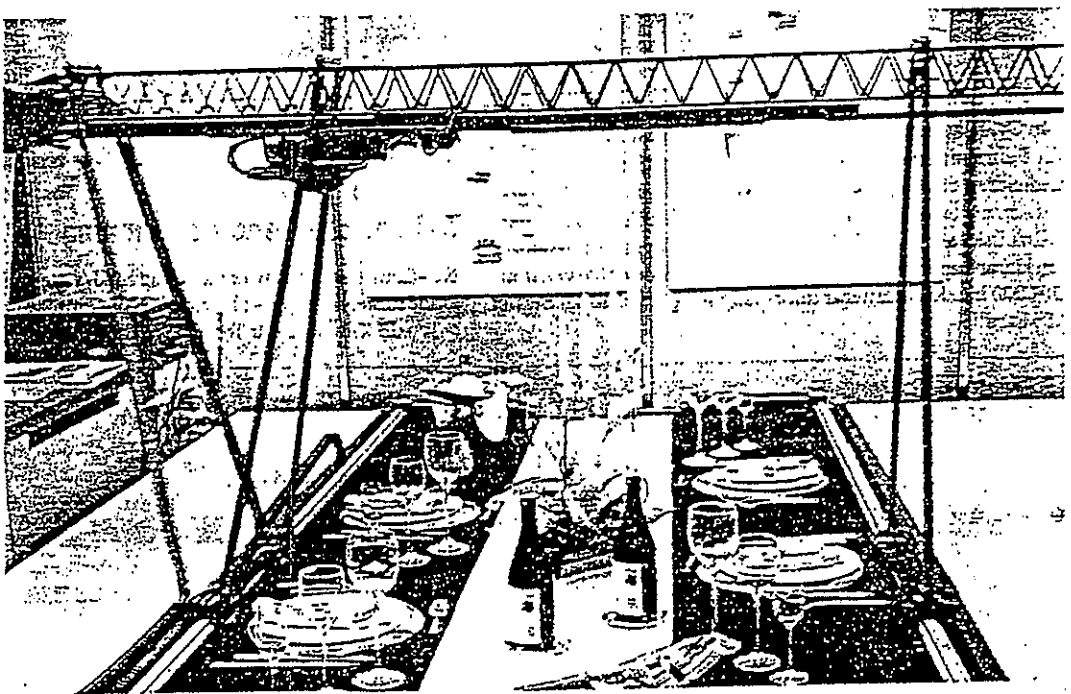
En grupp forskare vid universitetet i Ulm har löst problemen med gungande last under en kran. På Hannovermässan visade de på ett illusoriskt sätt hur det fungerar i praktiken genom att låta en modell av en bockkran föra en tyngd över ett dukat middagsbord.

När tyngden pendlade mellan kristallglasen drog publiken efter andan. Men reglertekniken hade hela tiden fullständig kontroll över tyngdens pendelrörelser och inga glas krossades.

FORSKARNA HAR använt sig av programmet Matlab för att räkna ut hur lastens pendelrörelser, öppning och vridning ska dämpas. Första steget var att skriva ett program som uppnäver pendlingen. I nästa steg gav de sig på att dämpa öppnings- och vridrörelserna.

Tredje steget var att bibehålla de uppnådda egenskaperna även när kranen är i rörelse, och göra det möjligt att programmera en speciell färdväg.

Förutom modellen med en dryg meters spännvidd och en färdväg på drygt två meter så har Ulmforskarna



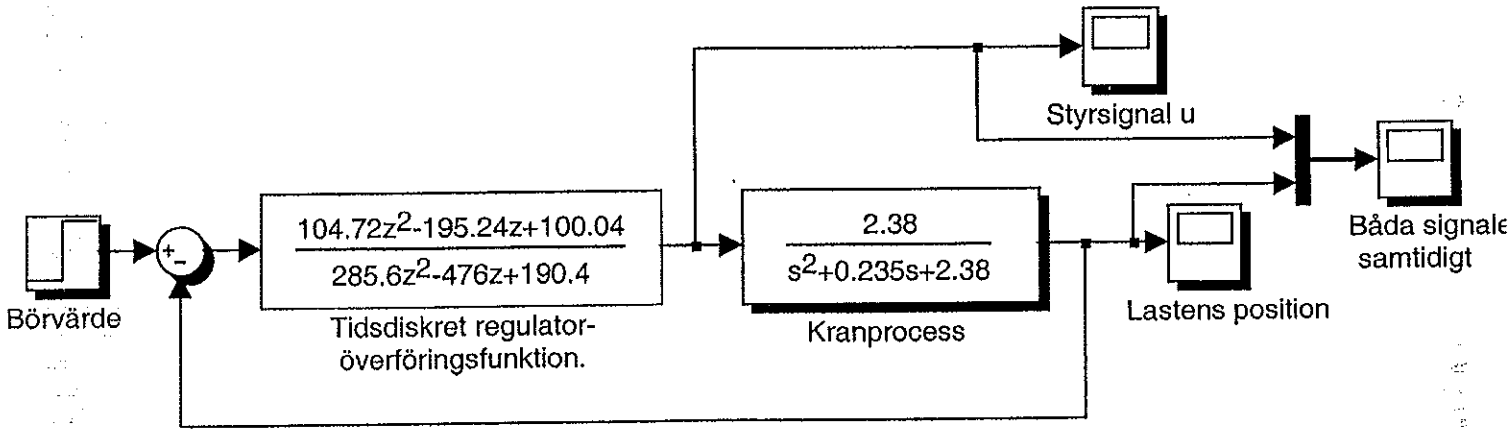
Ett dukat bord på Hannovermässan användes för att visa precisionen hos en bockkran. Reglerteknik minimerar pendelrörelserna i en hängande tyngd som manövreras mellan glas och flaskor.

provat teorierna på en 5 tons bockkran i ett lager som mäter 8 x 36 m. Arbetshöjden är 8 meter.

I DEN VERKLIGA MILJÖN lyckades man få de önskade egenskaperna vid hastigheter om 1 m/s och accelerationer på 0,7 m/s^2. Styr och reglersystemet är lämpligt för kranar i automatiska varulager.

Det är fullt möjligt att placera en griparm från en robot för att hantera varorna och ändå ha full kontroll över lastens rörelser, hävdar forskarna.

De hoppas nu att någon av de etablerade krantillverkarna ska intressera sig för lastdämpningsprogrammet. ■



Exempel på hur en liknande simulinkmodell kan se ut med en tidsdiskret regulator och en analog process.

Förteckning över kommandon

Nedanstående lista visar alla kommandon i Control System Toolbox, version 4.0, med en kort beskrivning av respektive kommando:

Creation of LTI models.

ss - Create a state-space model.
zpk - Create a zero/pole/gain model.
tf - Create a transfer function model.
dss - Specify a descriptor state-space model.
filt - Specify a digital filter.
set - Set/modify properties of LTI models.
ltiprops - Detailed help for available LTI properties.

Data extraction.

ssdata - Extract state-space matrices.
zpkdata - Extract zero/pole/gain data.
tfdata - Extract numerator(s) and denominator(s).
dssdata - Descriptor version of SSDATA.
get - Access values of LTI model properties.

Model characteristics.

class - Model type ('ss', 'zpk', or 'tf').
size - Input/output dimensions.
isempty - True for empty LTI models.
isct - True for continuous-time models.
isdt - True for discrete-time models.
isproper - True for proper LTI models.
issiso - True for single-input/single-output systems.
isa - Test if LTI model is of given type.

Conversions.

ss - Conversion to state space.
zpk - Conversion to zero/pole/gain.
tf - Conversion to transfer function.
c2d - Continuous to discrete conversion.
d2c - Discrete to continuous conversion.
d2d - Resample discrete system or add input delay(s).

Overloaded arithmetic operations.

+ and - - Add and subtract LTI systems (parallel connection).
* - Multiplication of LTI systems (series connection).
\ - Left divide -- sys1\sys2 means inv(sys1)*sys2.
/ - Right divide -- sys1/sys2 means sys1*inv(sys2).
' - Pertransposition.
.' - Transposition of input/output map.
[..] - Horizontal/vertical concatenation of LTI systems.

inv - Inverse of an LTI system.

Model dynamics.

pole, eig - System poles.
tzero - System transmission zeros.
pzmap - Pole-zero map.
dcgain - D.C. (low frequency) gain.
norm - Norms of LTI systems.
covar - Covariance of response to white noise.
damp - Natural frequency and damping of system poles.
esort - Sort continuous poles by real part.
dsort - Sort discrete poles by magnitude.
pade - Pade approximation of time delays.

State-space models.

rss,drss - Random stable state-space models.
ss2ss - State coordinate transformation.
canon - State-space canonical forms.
ctrb, obsv - Controllability and observability matrices.
gram - Controllability and observability gramians.
ssbal - Diagonal balancing of state-space realizations.
balreal - Gramian-based input/output balancing.
modred - Model state reduction.
minreal - Minimal realization and pole/zero cancellation.
augstate - Augment output by appending states.

Time response.

step - Step response.
impulse - Impulse response.
initial - Response of state-space system with given initial state.
lsim - Response to arbitrary inputs.
ltiview - Response analysis GUI.
gensig - Generate input signal for LSIM.
stepfun - Generate unit-step input.

Frequency response.

bode - Bode plot of the frequency response.
sigma - Singular value frequency plot.
nyquist - Nyquist plot.
nichols - Nichols chart.
ltiview - Response analysis GUI.
evalfr - Evaluate frequency response at given frequency.
freqresp - Frequency response over a frequency grid.
margin - Gain and phase margins.

System interconnections.

append - Group LTI systems by appending inputs and outputs.
parallel - Generalized parallel connection (see also overloaded +).
series - Generalized series connection (see also overloaded *).

feedback - Feedback connection of two systems.
star - Redheffer star product (LFT interconnections).
connect - Derive state-space model from block diagram description.

Classical design tools.

rlocus - Evans root locus.
rlocfind - Interactive root locus gain determination.
acker - SISO pole placement.
place - MIMO pole placement.
estim - Form estimator given estimator gain.
reg - Form regulator given state-feedback and estimator gains.

LQG design tools.

lqr,dlqr - Linear-quadratic (LQ) state-feedback regulator.
lqry - LQ regulator with output weighting.
lqrd - Discrete LQ regulator for continuous plant.
kalman - Kalman estimator.
kalmd - Discrete Kalman estimator for continuous plant.
lqgreg - Form LQG regulator given LQ gain and Kalman estimator.

Matrix equation solvers.

lyap - Solve continuous Lyapunov equations.
dlyap - Solve discrete Lyapunov equations.
care - Solve continuous algebraic Riccati equations.
dare - Solve discrete algebraic Riccati equations.

Demonstrations.

ctrldemo - Introduction to the Control System Toolbox.
jetdemo - Classical design of jet transport yaw damper.
diskdemo - Digital design of hard-disk-drive controller.
milldemo - SISO and MIMO LQG control of steel rolling mill.
kalmdemo - Kalman filter design and simulation.