Simulering i MATLAB og SIMULINK

Av Finn Haugen (finn@techteach.no)
TechTeach (http://techteach.no)

13. november 2004

Innhold

Forord

Dette dokumentet beskriver simulering i MATLAB [2] og SIMULINK [1]. Dokumentet utgjør tilleggsmateriale til læreboka *Dynamiske systemer – modellering, analyse og simulering*. Dette dokumentet og filer som benyttes eller som det henvises til i dokumentet, kan lastes ned via hjemmesiden for læreboka på **http://techteach.no**. Et eget dokument gir en introduksjon til simulering i LabVIEW, se http://techteach.no.

Skien, 13. november 2004

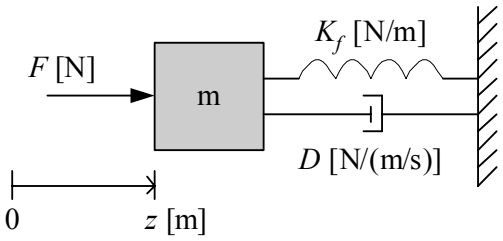
Finn Haugen

Kapittel 1

Simulering av differensiallikningsmodeller

1.1 Innledning

Det fins mange kommersielle likningsløsere — eller simulatorer — for differensiallikningsmodeller. Det forutsettes vanligvis at differensiallikningene er på tilstandsromform. Vi skal se på noen aktuelle verktøy i MATLAB [**?**] og SIMULINK [1]. Som gjennomgående eksempel skal responsen i posisjonen x_1 for et masse–fjær-demper-system beregnes, se figur 1.1. Kraftbalanse gir følgende matematiske modell:



Figur 1.1: Masse-fjær-demper

$$m\ddot{z} = -D\dot{z} - K_f z + F \tag{1.1}$$

Ved å innføre følgende de generelle variablene x_1 for posisjonen z, x_2 for hastigheten $\dot{z}$ og u for kraften F, kan modellen skrives som følgende

tilstandsrommodell, som er lineær:

$$\underbrace{\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix}}_{\dot{\underline{x}}} = \begin{bmatrix} x_1 \\ -\frac{K_f}{m}x_1 - \frac{D}{m}x_2 + \frac{1}{m}u \end{bmatrix} \tag{1.2}$$

$$= \underbrace{\begin{bmatrix} 0 & 1 \\ -\frac{K_f}{m} & -\frac{D}{m} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{\underline{x}} + \underbrace{\begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix}}_{B} u \tag{1.3}$$

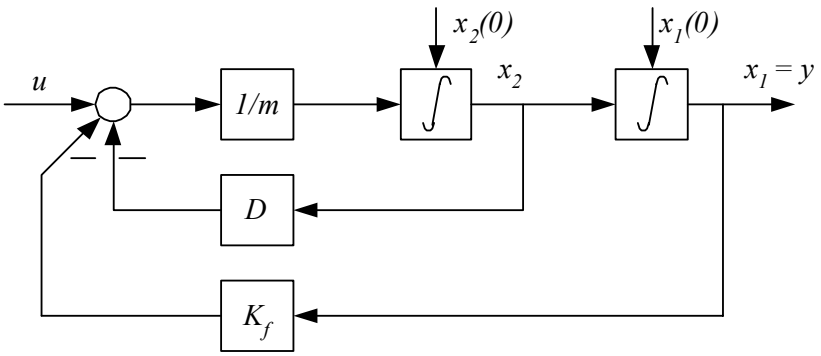
$$= A\underline{x} + Bu \tag{1.4}$$

$$y = x_1 \tag{1.5}$$

$$= \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{C} \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{\underline{x}} + \underbrace{\begin{bmatrix} 0 \end{bmatrix}}_{D} u \tag{1.6}$$

$$= C\underline{x} + Du \tag{1.7}$$

Modellen kan framstilles i et blokkdiagram som vist i figur 1.2.



Figur 1.2: Blokkdiagram av modellen for masse-fjær-demper-systemet

I simuleringene skal kraften u (inngangssignalet) være et sprang fra 0 til 4 N ved $t = 0$. Initialtilstanden er $x_1(0) = 0$ og $x_2(0) = 0$.

1.2 Simulering med MATLAB

1.2.1 Funksjonen step

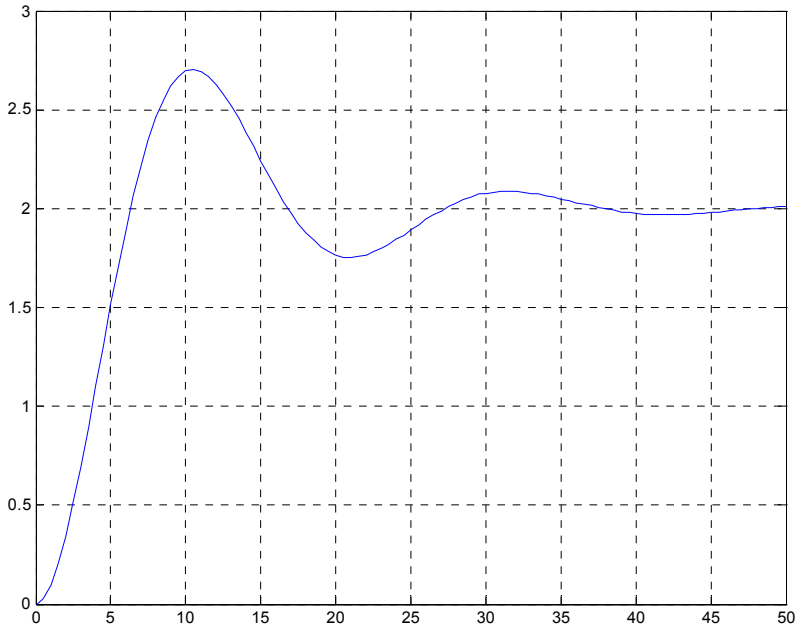
Denne funksjonen inngår i MATLABs Control System Toolbox. step kan beregne sprangresponsen for *lineære* tilstandsrommodeller (og transferfunksjonsmodeller). step påtrykker systemet automatisk et enhetssprang (sprang med høyde 1). MATLAB-skriptet nedenfor viser hvordan step kan benyttes for å beregne sprangresponsen i posisjonen $y = x_1$ for masse-fjær-demper-systemet.

```
%MATLAB-skript: mfd_step.m
U=4; %Spranghøyden i u.
m=20; Kf=2; D=4;
A=[0,1;-Kf/m,-D/m];
B=[0;1/m];
C=[1 0]; %Sørger for at utgangen er y=x1 (posisjonen).
D=[0];
%Definerer en LTI-modell med navn mfd vha. ss-funksjonen:
mfd=ss(A,B,C,D); %ss står for state-space (tilstandsrom)
tstopp=50;
[y1,t]=step(mfd,tstopp); %y er utgangsvektor. %t er tidsvektor.
%step-funksjonen antar at inngangen er enhetssprang.
y=y1*U; %Gir respons som svarer til inngangsspranghøyde U
%Plotting:
figure(1)
plot(t,y),ylabel('[m]'),title('Posisjon, y')
```

Figur 1.3 viser $y(t)$ som beregnet vha. step.

1.2.2 Funksjonen lsim

Funksjonen lsim (linear simulation) i MATLABs Control System Toolbox kan beregne responsen for *lineære* tilstandsrommodeller (og transferfunksjonsmodeller) for vilkårlige inngangssignaler som brukeren selv har generert i form av et array (vektor). MATLAB-skriptet nedenfor viser hvordan lsim kan benyttes for å beregne sprangresponsen i posisjonen $y = x_1$ for masse-fjær-demper-systemet.



Figur 1.3: $y(t)$ beregnet med funksjonen step i MATLABs Control System Toolbox. (Fil: mfd_step.m)

```
%MATLAB-skript: mfd_lsim.m
tstopp=50;
t=[0:.1:tstopp]'; %Generer array for t
U=4;%Spranghoyden i u.
%Lager sprangtidsserie:
u=U*ones(length(t),1);%ones-funksjonen lager matrise av 1-ere
m=20; Kf=2; D=4;
A=[0,1;-Kf/m,-D/m];
B=[0;1/m];
C=[1 0];%Sørger for at utgangen er y=x1 (posisjonen).
D=[0];
%Definerer LTI-modell med navn mfd vha. ss-funksjonen:
mfd=ss(A,B,C,D);
x0=[0;0];
y=lsim(mfd,u,t);%Simulerer med lsim. Respons i y.
%Plotting:
```

```
figure(1)
plot(t,y),ylabel('[m]'),title('Posisjon, y')
grid
```

Plottet av $y(t) = x_1(t)$ blir identisk med plottet vist i figur 1.3 (som ble generert med MATLAB-skriptet mfd_step.m).

1.2.3 Funksjonen impulse

Funksjonen impulse virker som step og lsim, bortsett fra at inngangsssignalet ved impulse er en impuls med styrke 1. Impulse-funksjonen beskrives ikke nærmere her.

1.2.4 Funksjonen ode45

Funksjonen ode45 er én av de numeriske løsningsfunksjonene for *ulineære* differensiallikninger (på tilstandsromform) i MATLAB.[1] ode45 implementerer en såkalt Runge-Kutta (4,5) metode. ode45 er velegnet for mange typer systemer og kan benyttes som standardmetode. ODE står for Ordinary Differential Equations.

Vi skal benytte ode45 til å beregne sprangresponsen i posisjonen $y = x_1$ for masse-fjær-demper-systemet. Først må vi lage en MATLAB-funksjon som definerer tilstandsrommodellen, se (mfd.m) nedenfor.

```
%MATLAB-funksjonen mfd.m
function xdot=mfd(t,x)
global m D Kf
u=4;
xdot(1)=x(2);
xdot(2)=-(Kf/m)*x(1)-(D/m)*x(2)+(1/m)*u;
xdot=[xdot(1);xdot(2)];
```

Vi lager (og kjører) så et skript som løser settet av differensiallikninger og plotter responsen $x_1(t) = y(t)$, se mfd_ode45.m nedenfor.

```
%MATLAB-skript: mfd_ode45.m
```

[1] MATLAB-kommandoen help funfun lister opp andre differensialllikningsløsere.

```
clear
t0=0; %Starttid
t1=50; %Sluttid
x0=[0;0]; %Initialtilstand
global m Kf D;
m=20;Kf=2;D=4;
[t,x]=ode45(@mfd,[t0,t1],x0);
figure(1)
plot(t,x(:,1)) %Plotter x1=y som funksjon av t.
grid
```

Plottet av $y(t) = x_1(t)$ blir identisk med plottet vist i figur 1.3.

1.3 Simulering med SIMULINK

1.3.1 Innledning

SIMULINK er et simuleringsverktøy som er integrert med MATLAB. Det forutsettes her at du har grunnleggende ferdigheter i bruk av SIMULINK [1].

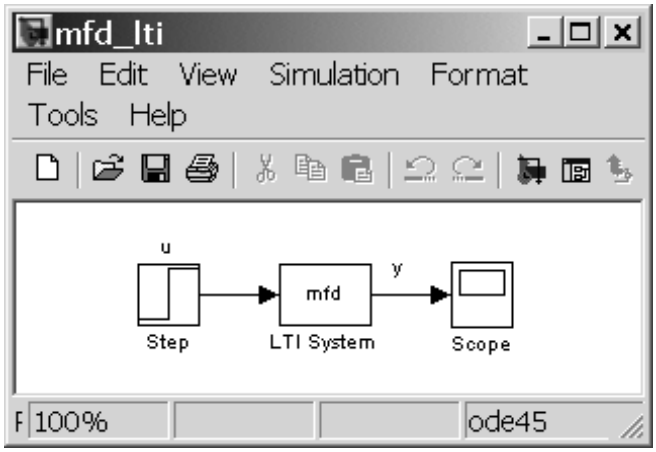
Noen ord om måten jeg har valgt å gjennomføre simuleringene på i SIMULINK: Etter min erfaring er det mest hensiktsmessig bruke *to filer*, slik:

- En modellfil (*.mdl) som definerer SIMULINK-modellen
- Et MATLAB-skript (*.m) som som initialiserer (bl.a. definerer modellparametrene, som brukes i blokkene) og gjennomfører simuleringen og evt. etterbehandler data fra simuleringen (f.eks. lagrer til fil).

I det følgende beskrives forskjellige SIMULINK-simuleringer svarende til alternative måter å representere masse-fjær-demper-systemets modell på (i SIMULINK).

1.3.2 Modellrepresentasjon med LTI-blokk

SIMULINK har en LTI-blokk[2] (som er tilgjengelig når Control System Toolbox er installert) som kan brukes for en effektiv representasjon av lineære modeller i form av *LTI-objekter*. LTI-objekter genereres vha. egnede funksjoner i Control System Toolbox, f.eks. `ss` (state-space) for tilstandsrommodeller og `tf` (transfer function) for transferfunksjoner. Figur 1.4 viser SIMULINK-blokkdiagrammet med bl.a. LTI-blokka. MATLAB-skriptet som definerer LTI-objektet og kjører simuleringen av dette blokkdiagrammet, er gjengitt nedenfor.



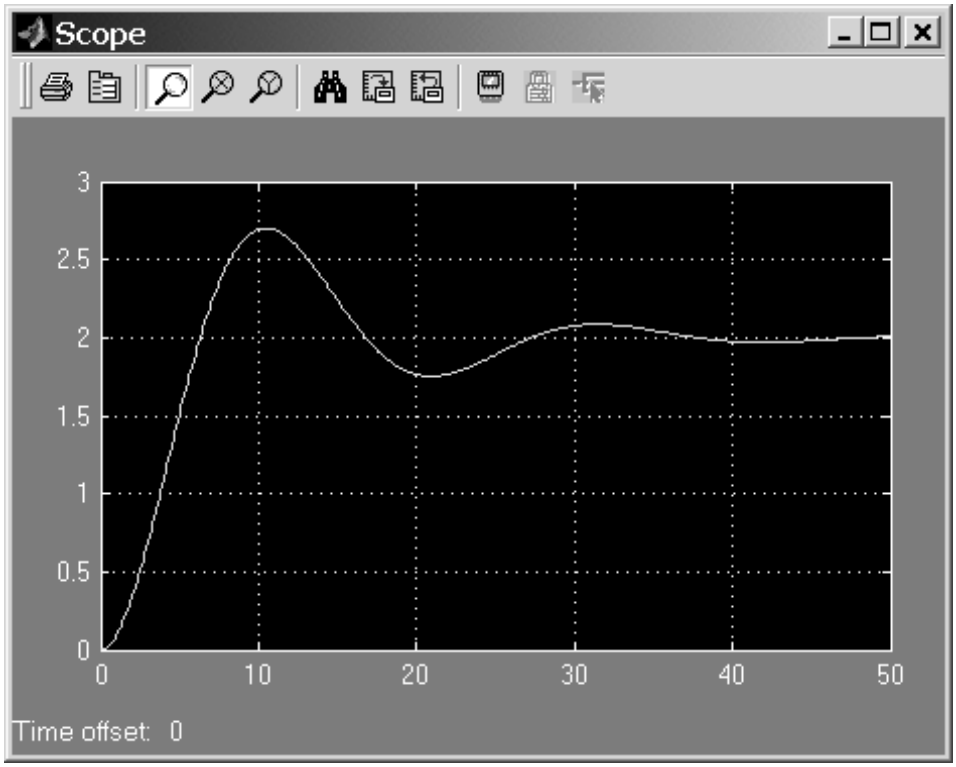
Figur 1.4: SIMULINK-blokkdiagram for masse-fjær-demper-systemet, hvis tilstandsrommodell er representert med en LTI-blokk

```
%MATLAB-skript skriptsim_mfd_lti.m
tid_sprang_u=0;
u0=0;
u1=4;
m=20; Kf=2; D=4;
A=[0,1;-Kf/m,-D/m];
B=[0;1/m];
C=[1 0];%Sørger for at utgangen er y=x1 (posisjonen).
D=[0];
%Definerer en LTI-modell med navn mfd vha. ss-funksjonen:
mfd=ss(A,B,C,D); %ss står for state-space (tilstandsrom)
```

[2] LTI = Linear Time-Invariant

```
%Setter simuleringsparametre:
tidsskritt=0.1;
options=simset('solver','ode5','fixedstep',tidsskritt);
tstopp=50;
%Simulerer systemet mfd_lti.mdl:
sim('mfd_lti',tstopp,options)
```

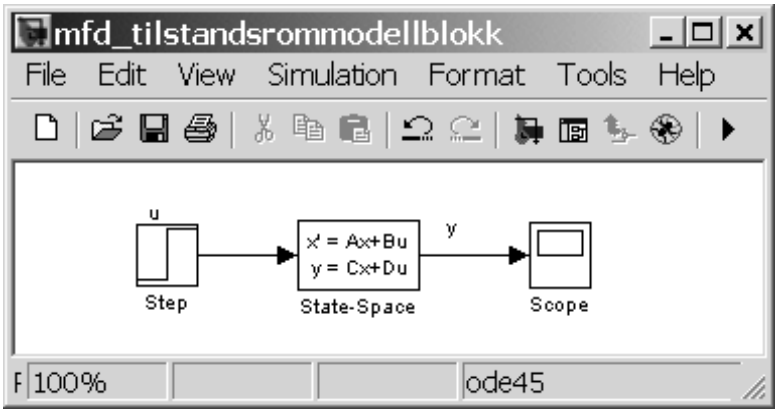
Figur 1.5 viser responsen i posisjonen $y(t) = x_1(t)$. (Plottet vises ved å dobbelklikke på Scope-blokken i SIMULINK-blokkdiagrammet, se figur 1.4).



Figur 1.5: SIMULINK-simulering av responsen i posisjonen $y = x_1$ for masse-fjær-demper-systemet

1.3.3 Modellrepresentasjon med innebygd tilstandsrommodellblokk

Blokken StateSpace i blokkbiblioteket Continuous kan brukes til å representere lineære tilstandsrommodeller i SIMULINK (som alternativ til bruk av LTI-blokken beskrevet ovenfor). Figur 1.6 viser blokkdiagrammet med StateSpace-blokken. Figur 1.7 viser blokkens parametervindu (som



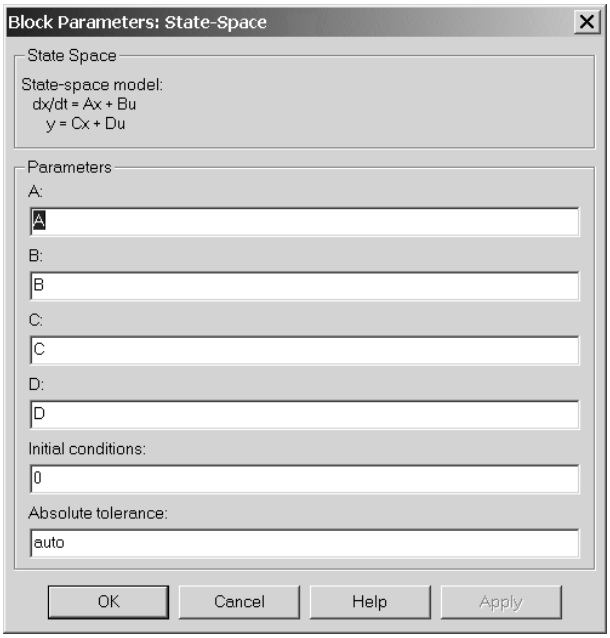
Figur 1.6: SIMULINK-blokkdiagram for masse-fjær-demper-systemet der blokken StateSpace benyttes for å representere tilstandsrommodellen

åpnes ved å dobbelklikke på blokken). MATLAB-skriptet som definerer tilstandsrommodellen og kjører simuleringen, blir det samme som skriptet skriptsim_mfd_lti.m som er gjengitt ovenfor, bortsett fra at uttrykket mfd=ss(A,B,C,D) sløyfes. Responsen i posisjonen $y(t) = x_1(t)$ blir som vist i figur 1.5.

1.3.4 Modellrepresentasjon med detaljert blokkdiagram

Dersom du har tegnet et detaljert blokkdiagram av den matematiske modellen som skal benyttes i en simulator, kan du implementere blokkdiagrammet direkte et SIMULINK-blokkdiagram.

Figur 1.2 viser et detaljert matematisk blokkdiagram for masse-fjær-demper-systemet. Figur 1.8 viser hvordan dette blokkdiagrammet kan implementeres direkte i SIMULINK. (Bakgrunnen for

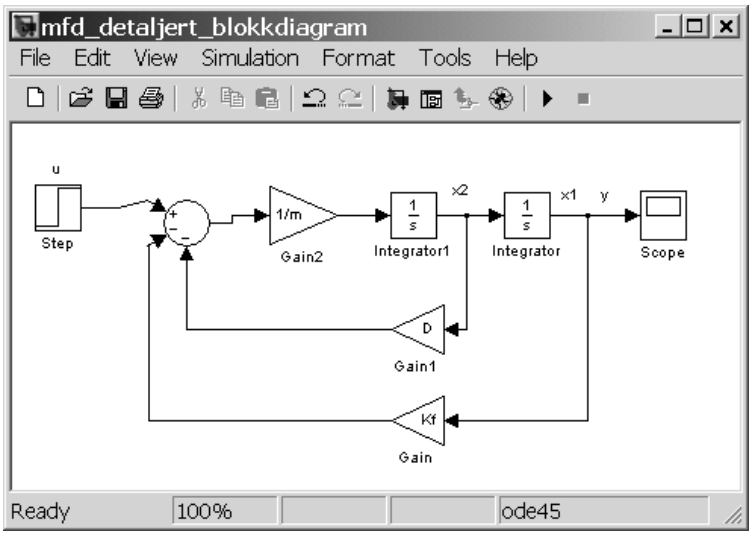


Figur 1.7: Parametervinduet for tilstandsrommodellblokken i SIMULINK (vinduet åpnes ved å dobbelklikke på blokken)

at symbolet $\frac{1}{s}$ brukes for integratoren, er at $\frac{1}{s}$ er Laplace-transferfunksjonen for en integrator.)

MATLAB-skriptet gjengitt nedenfor definerer modellparametrene og gjennomfører simuleringen. Responsen i posisjonen $y(t) = x_1(t)$ blir som vist i figur 1.5.

```
%MATLAB-skript skriptsim_mfd_detaljert_diagram.m
tid_sprang_u=0;
u0=0;
u1=4;
m=20; Kf=2; D=4;
%Setter simuleringsparametre:
tidsskritt=0.1;
options=simset('solver','ode5','fixedstep',tidsskritt);
tstopp=50;
%Simulerer systemet mfd_detaljert_blokkdiagram.mdl:
sim('mfd_detaljert_blokkdiagram',tstopp,options)
```



Figur 1.8: Implementering av masse-fjær-demper-systemets blokkdiagrammodell i SIMULINK

Men hvorfor bruke et detaljert blokkdiagram når det fins mer effektive modellblokker, som LTI-blokken eller StateSpace-blokken? En grunn kan være at det er enkelt å legge inn ulineære funksjoner for f.eks. hysterese eller metning på passende steder i et detaljert blokkdiagram. Ellers bør en bruke blokker som mest effektivt representerer modellen, som en LTI-blokk.

1.3.5 Modellrepresentasjon med S-funksjon

S-Functions-blokken i SIMULINK kan brukes til å kalle opp en S-funksjon (System Function), som er en MATLAB-funksjon der brukeren definerer modellen via tekstuttrykk, på noe av samme måten som ved bruk av ODE-funksjonene i MATLAB, se s. 11. Bruk av S-funksjoner beskrives ikke nærmere her. En referanse er [1].

Kapittel 2

Simulering av transferfunksjonsmodeller

2.1 Innledning

Både MATLAB og SIMULINK har simuleringsfunksjoner for transferfunksjoner. De beskrives nedenfor.

Som gjennomgående eksempel skal responsen i posisjonen y for masse-fjær-demper-systemet vist i figur 1.1 simuleres. Den matematiske modellen basert på kraftbalanse er:

$$m\ddot{y} = -D\dot{y} - K_f z + u \tag{2.1}$$

Systemets transferfunksjon fra kraften u til posisjonen y er $H(s)$:

$$y(s) = \underbrace{\frac{1}{ms^2 + Ds + K_f}}_{H(s)} u(s) \tag{2.2}$$

2.2 Simulering med MATLAB

Beregning av tidsresponser for tilstandsrommodeller med step, lsim og impulse (som inngår i MATLABs Control System Toolbox [5]) ble beskrevet

i kap. 1. step, lsim og impulse kan brukes også for transferfunksjoner. Vi nøyer oss her med å se på bruken av step (lsim og impulse kan brukes på tilsvarende måte).

step påtrykker systemet et enhetssprang (sprang med høyde 1). MATLAB-skriptet nedenfor viser hvordan step kan benyttes for å beregne sprangresponsen i posisjonen y for masse-fjær-demper-systemet (det er et sprang med høyde 4 i kraften u). Transferfunksjonen er definert som et LTI-objekt vha. funksjonen tf.

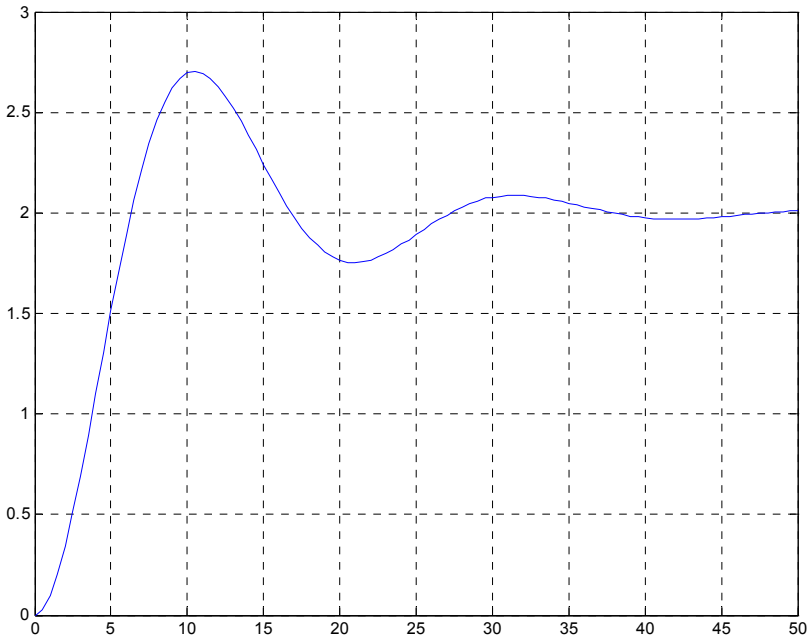
```
%MATLAB-skript: mfd_tf_step.m
U=4; %Spranghoyden i u.
m=20; Kf=2; D=4;
%Definerer en LTI-modell med navn mfd vha. tf-funksjonen:
tellerpolynom=[1]; %Koeffisientene i tellerpolynomet:
nevnerpolynom=[m,D,Kf]; %Koeffisientene i tellerpolynomet:
mfd=tf(tellerpolynom,nevnerpolynom); %tf står for transfer function
tstopp=50;
[y1,t]=step(mfd,tstopp);%y1 er utgangsvektor.%t er tidsvektor.
%step-funksjonen antar at inngangen er enhetssprang.
y=y1*U; %Gir respons som svarer til inngangsspranghøyde U
%Plotting:
figure(1)
plot(t,y),ylabel('[m]'),xlabel('t [sek]'),title('Posisjon, y')
grid
```

Figur 2.1 viser $y(t)$ som beregnet vha. step.

2.3 Simulering med SIMULINK

SIMULINK er et blokkdiagrambasert simuleringsverktøy som bygger på MATLAB. SIMULINK er kort beskrevet på side 1.3 osv. SIMULINK har to muligheter for simulering av transferfunksjoner:

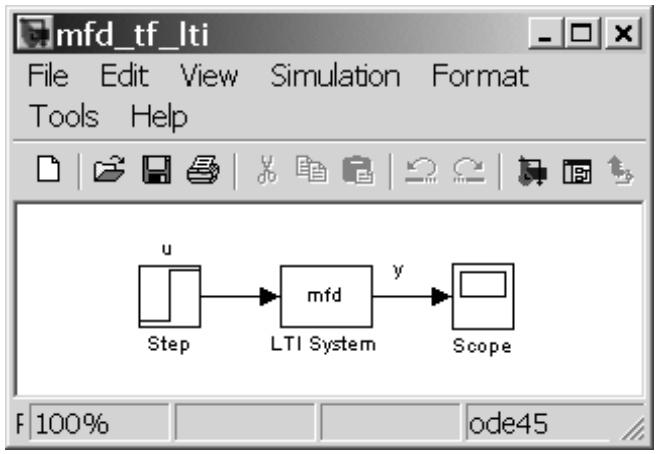
- Bruk av LTI-blokk med et LTI-objekt av typen transferfunksjon generert med funksjonen tf i Control System Toolbox.
- Bruk av innebygde transferfunksjonsblokker.



Figur 2.1: Posisjonen $y(t)$ for masse-fjær-systemets transferfunksjon (2.2). $y(t)$ er beregnet med step-funksjonen i MATLAB.

Vi nøyer oss her med å se på bruk av LTI-blokken, som gir en noe mer fleksibel modellrepresentasjon enn transferfunksjonsblokker. Figur 2.2 viser blokkdiagrammet i SIMULINK. MATLAB-skriptet som kjører simuleringen, er gjengitt nedenfor.

```
%MATLAB-skript skriptsim_mfd_tf_lti.m
tid_sprang_u=0;
u0=0;
u1=4;
m=20; Kf=2; D=4;
%Definerer en LTI-modell med navn mfd vha. tf-funksjonen:
tellerpolynom=[1]; %Koeffisientene i tellerpolynomet:
nevnerpolynom=[m,D,Kf]; %Koeffisientene i tellerpolynomet:
mfd=tf(tellerpolynom,nevnerpolynom); %tf står for transfer function
```



Figur 2.2: SIMULINK-blokkdiagram for masse-fjær-demper-systemet, hvis transferfunksjonsmodell er representert med en LTI-blokk

```
%Setter simuleringsparametre:
tidsskritt=0.1;
options=simset('solver','ode5','fixedstep',tidsskritt);
tstopp=50;


%Simulerer systemet mfd_tf_lti.mdl:
sim('mfd_tf_lti',tstopp,options)
```

Den beregnede responsen $y(t)$ blir som ved bruk av step-funksjonen, se figur 2.1.

Bibliografi

[1] Finn Haugen: **Lær SIMULINK trinn for trinn**, Tapir Akademisk Forlag, 2003

[2] Finn Haugen: **Lær MATLAB trinn for trinn**, Tapir Akademisk Forlag, 2003

[3] Finn Haugen: **Lær LabVIEW trinn for trinn**, Tapir Akademisk Forlag, 2003

[4] Finn Haugen: **Dynamiske systemer - modellering, analyse og simulering**, Tapir Akademisk Forlag, 2003

[5] Finn Haugen: **Tutorial for Control System Toolbox**, TechTeach (http://techteach.no), 2002