FYS2130 forelesning 1. februar 2013
Noen kommentarer til kapittel 3: Numeriske løsningsmetoder

Numerisk løsning av annen ordens differensialligning:

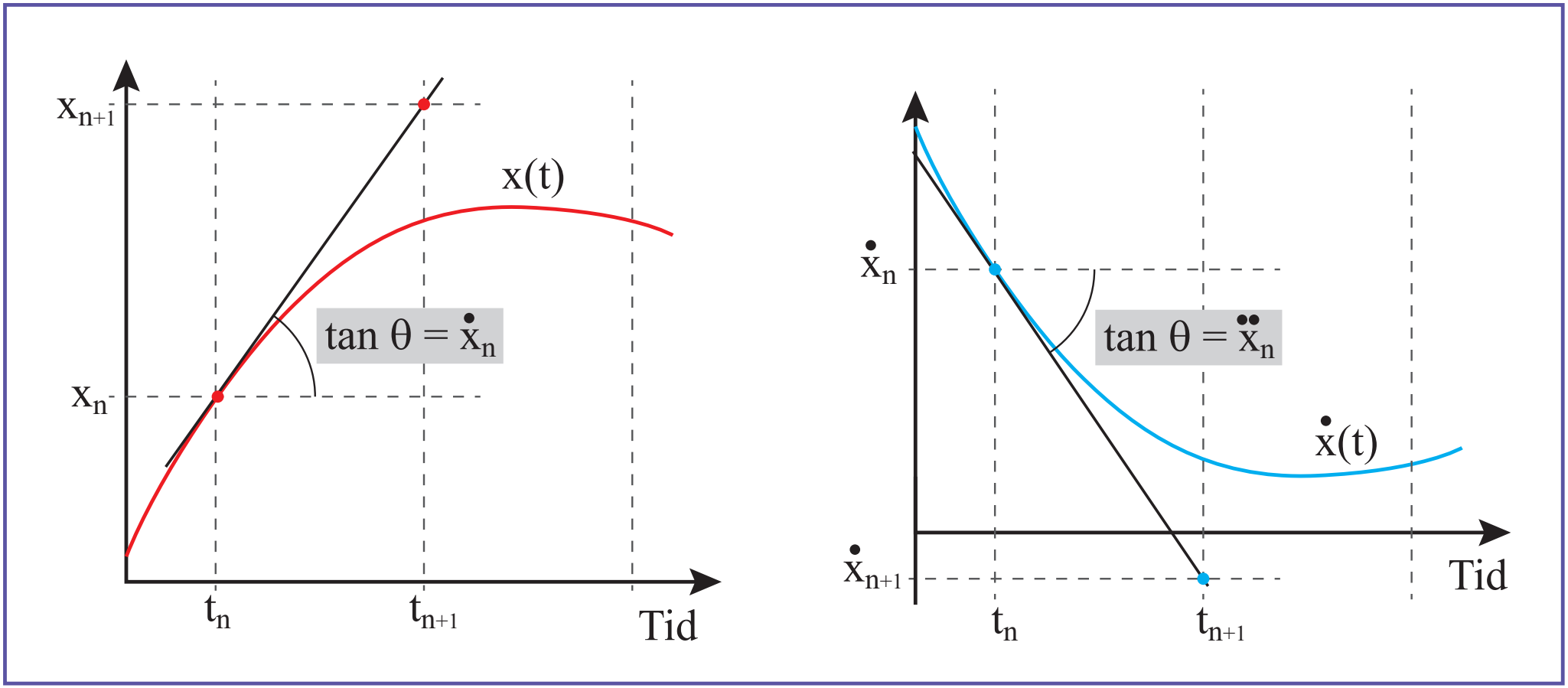
1.
Kan skrive differensialligningen som en sum av to koblede første ordens differensialligninger.

2.
Starter så med initialbetingelsene og integrerer opp trinn for trinn, basert på den underliggende differensialligningen.

$$\ddot{x} \equiv \frac{\mathrm{d}^2 x}{\mathrm{d}t^2} = f(x(t), \dot{x}(t), t).$$

$$\frac{\mathrm{d}\dot{x}}{\mathrm{d}t} = f(x(t), \dot{x}(t), t)$$

$$\frac{\mathrm{d}x}{\mathrm{d}t} = g(x(t), t)$$



$$\dot{x}(t_n) = \lim_{\Delta t \to 0} \frac{x(t_{n+1}) - x(t_n)}{\Delta t}$$

$$\ddot{x}(t_n) = \lim_{\Delta t \to 0} \frac{\dot{x}(t_{n+1}) - \dot{x}(t_n)}{\Delta t}$$

$$x_{n+1} = x_n + \dot{x}_n \, \Delta t$$

$$\dot{x}_{n+1} = \dot{x}_n + \ddot{x}_n \, \Delta t$$

3.0.1 Matlab-kode for Eulers metode

```
function [xnp,vnp,tnp] = euler(xn,vn,tn,delta_t,param)

vnp = vn + an(xn,vn,tn,param)*delta_t;
xnp = xn + vn*delta_t;
tnp = tn + delta_t;
return;
```

3.0.2 Matlab-kode for Euler-Cromers metode

```
function [xnp,vnp,tnp] = eulerCrom(xn,vn,tn,delta_t,param)

vnp = vn + an(xn,vn,tn,param)*delta_t;
xnp = xn + vnp*delta_t;
tnp = tn + delta_t;
return;
```

3.0.3 Matlab-kode for Eulers midtpunktmetode

```
function [xp,vp,tp] = eulerMidt(xn,vn,tn,delta_t,param)

vnh = vn + an(xn,vn,tn,param)*delta_t/2;
xnh = xn + vn*delta_t/2;

vnp = vn + an(xnh,vnh,tn+delta_t/2,param)*delta_t;
xnp = xn + vnh*delta_t;
tnp = tn + delta_t;
return;
```

3.0.4 Eksempel på Matlab-kode for annen-deriverte

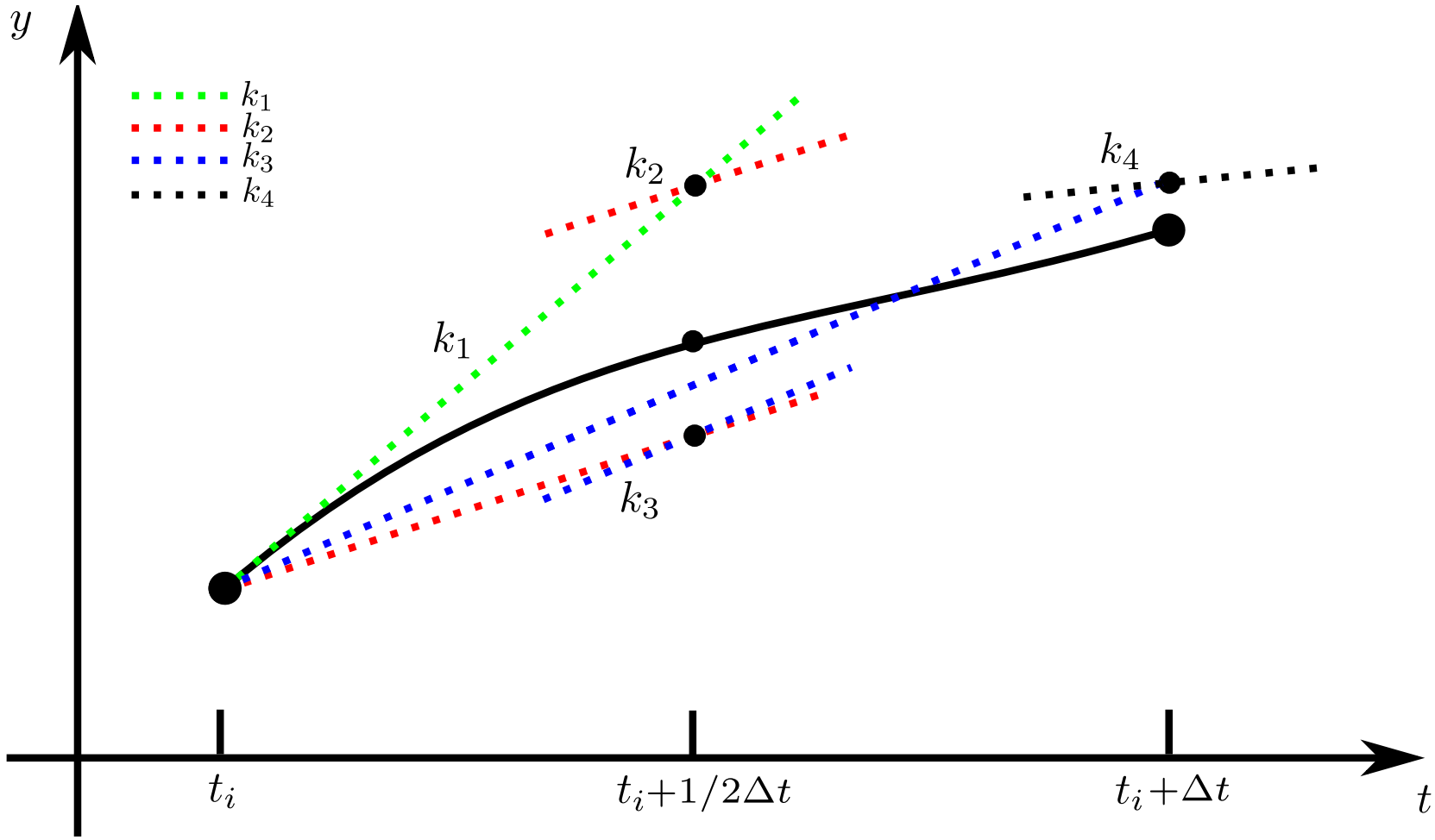
```
function [dvdt] = an(xn,vn,tn,delta_t,param)
dvdt = -param.kn*xn - param.bn*vn + param.Fmaxn*sin(param.omegad*tn);
return;
```

I hovedprogrammet er stukturen "param" definert slik:
param.kn = k/m, param.bn = b/m, Fmaxn = Fmax/m, param.omegad = omegaD.

3.0.5 Matlab-kode for Runge-Kuttas metode

```
function [xnp,vnp,tnp] = rk4r(xn,vn,tn,delta_t,param)
ffa = eval(['@' param.fn]); % Gir navn på funksjon som inneholder
                            % navn på uttrykket for annen derivert
halv_delta_t = 0.5*delta_t;
t_p_halv = tn + halv_delta_t;

x1 = xn;
v1 = vn;
a1 = ffa(x1,v1,tn,param);

x2 = x1 + v1*halv_delta_t;
v2 = v1 + a1*halv_delta_t;
a2 = ffa(x2,v2,t_p_halv,param);

x3 = x1 + v2*halv_delta_t;
v3 = v1 + a2*halv_delta_t;
a3 = ffa(x3,v3,t_p_halv,param);

tp = tn + delta_t;
x4 = x1 + v3*delta_t;
v4 = v1 + a3*delta_t;
a4 = ffa(x4,v4,tp,param);

% Returnerer (tilnærmet) (x,v,t) i slutten av intervallet.
delta_t6 = delta_t/6.0;
xnp = xn + delta_t6*(v1 + 2.0*(v2+v3) + v4);
vnp = vn + delta_t6*(a1 + 2.0*(a2+a3) + a4);

return;
```

Etter forelesningen sendte Morten Ledum meg en illustrering av fjerde ordens Runge Kutta. Morten skriver: "I Computational Phys. compendiet til Morten har han en figur (http://tinyurl.com/compfys, s.251) som hjalp meg veldig til å forstå de fire stegene i metoden." Herved er hans innspill delt med alle andre.

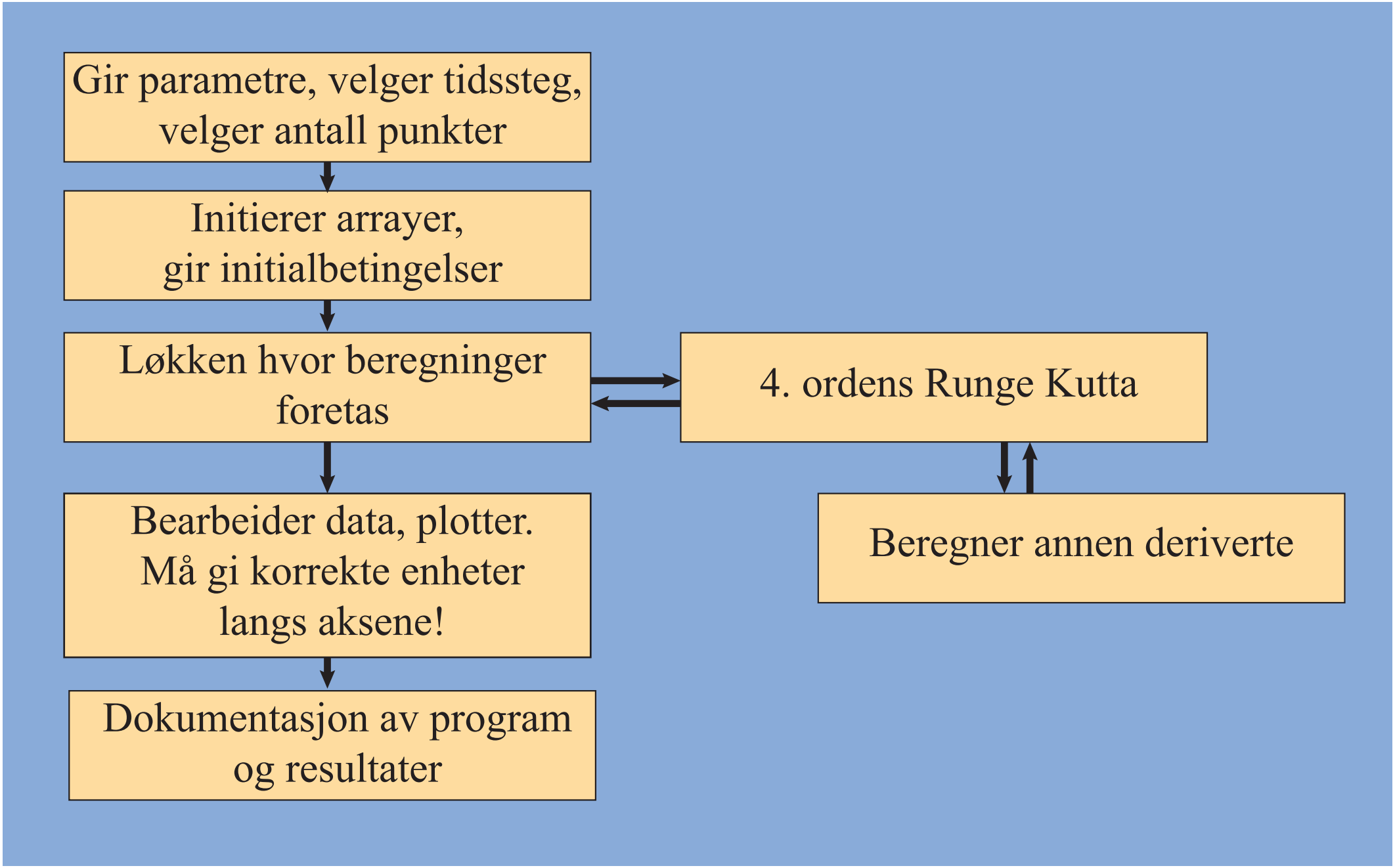


3.0.6 Eksempel på funksjon som inneholder differensial–ligningen

```
function dvdt = ffaVariantX(xn,vn,tn,param)

%*************************************************
% Funksjon for bruk i FYS2130 våren 2013.  Versjon 31012013
% Returnerer venstresiden i en diffligning for dv/dt
% Benyttes av Runge-Kutta 4 numerisk løsning av to koblede
% diffligninger, f.eks. for svingebevegelse for ulike varianter forhold.
%*************************************************

% Denne versjonen passer for dempet og påtrykt svingning av en fjærpendel
dvdt = -param.kn*xn - param.bn*vn + param.Fmaxn*sin(param.omegad*tn);

return;
```

```
%*********************************************************
AntallPerioder=80;      % Antall perioder beregningene bør gå over (ca 40)
n_pr_T=1000;            % Velger 1000 punkt pr periode (ca)
k=1.16;                 % "Fjærstivheten" (default 1.16)
m=85.6e-3;              % Loddets masse  (default 0.0856)
b=0.25*0.025;           % Friksjons-faktor  (default 0.02)
Fmax=0.4e-00;           % Amplituden for påtrykt kraft (default 2)
omega0=sqrt(k/m)        % Grunn-vinkelfrekvensen beregnes og skrives ut
omegaD=1.2*omega0;      % Velger (relativ) frekvens for påtrykt kraft
funknavn='ffaVariantX'; % Navn på Matlab-funksjonen som gir annen derivert
filnavn='sving11c002.txt'; % Navn på fil hvor parametre og resultater
                        % skal lagres for senere dokumentasjon
% MERK: Også ffa bør dokumenteres, f.eks. slik:
fvLigning1 = 'dvdt = -param.kn*xc - param.bn*vc + ';
fvLigning2 = 'param.Fmaxn*sin(param.omegad*tc);';
%*********************************************************

% Parametre nedenfor denne linjen skal ikke endres uten at programmet får
% nytt versjonsnummer (eller at endringen dokumenteres på annet vis).

delta = b/(2.0*m)       % Parameter for å bedømme om man har kritisk dempning
                        % eller ikke ( i så fall er delta=omega0). Skrives ut.
param.kn=k/m;           % Normerte størrelser inn i en struktur
param.bn=b/m;           % "param"
param.fn=funknavn;
param.Fmaxn=Fmax/m;
param.omegad=omegaD;
T0=2*pi/omega0;          % Svingetid for enkel harmonisk bevegelse
```

```
x=zeros(N,1);
t=zeros(N,1);
v=zeros(N,1);

% Setter initialbetingelsene
x(1)=double(startpos);
v(1)=double(startfart);
t(1)=0;

% Dernest kjøres løkken for å følge videre utvikling
for j=1:N-1
    [x(j+1), v(j+1), t(j+1)]=rk4r(x(j),v(j),t(j),delta_t,param);
end

% Diverse plot, først for posisjon vs tid, sammenlikner med analytisk
% x_analyt = startpos*cos(2*pi*t/T0);  % Ikke generelt!
plot(t,x,'-r');
%axis([0.0 5.0 -0.55 1.1])  % Bør kuttes ut vanligvis!
%plot(t,x,'-r',t,x_analyt,'-b');
%legend('numerisk løsning','analytisk løsning');
title('Posisjon vs tid');  % Skift ut tallet etter hva som beregnes
xlabel('Tid (sek)');
ylabel('Utsving (meter)');

figure; % Et nytt plott, for faserom-diagram
plot(x,v);
xlabel('Posisjon (m)');
ylabel('Hastighet (m/s)');
title('Faserom-presentasjon av svingebevegelsen');
```


Gir parametre, velger tidssteg, velger antall punkter
Initierer arrayer, gir initialbetingelser
Løkken hvor beregninger foretas
4. ordens Runge Kutta
Beregner annen deriverte
Bearbeider data, plotter. Må gi korrekte enheter langs aksene!
Dokumentasjon av program og resultater

Krav:

1. Sjekke innvirkning av valg av tidssteg.

2. Sjekke om programmet gir samme resultat som analytisk der dette lar seg sjekke.

3. Variablenavn lett gjenkjennelige sml med annen beskrivelse.

4. Generelle moduler, for å lette gjenbruk.

5. Passe med kommentarer. Tilstrekkelig dokumentasjon av kombinasjonen program og resultater.

6. Ha øye for ressursbruk (regnetid og hukommelse).

Feilsøking:

1. Sjekk biter av programmet underveis mens du skriver det.

2. Forsøk å skjønne feilmeldingen dersom du får en slik. Det kan være mer hjelp i disse meldingene enn du straks innser.

3. Velg å skrive ut en del størrelser underveis, for å se at verdiene er slik de skal være.

4. Vær tålmodig og systematisk, ikke få panikk og søke vilt.

5. Spør om hjelp etter at du har forsøkt litt på egen hånd. Ikke kast bort tid på endeløs feilsøking alene dersom du ikke har en god nok strategi på å finne feilen.

Lykke til!