FYS2130 - Oblig 3

Aleksander Hansen

24. februar 2013

Oppgave 1

Eulers metode kan tenkes på som en Runge-Kutta metode av første orden. Eulers metode bruker bare ett punkt for å avansere langs løsningen, mens RK4 bruker et vektet gjennomsnitt av fire punkter. Dette reduserer feilene akkumulert ved hvert seg og gjør RK4 mer presis og stabil enn Eulers metoed.

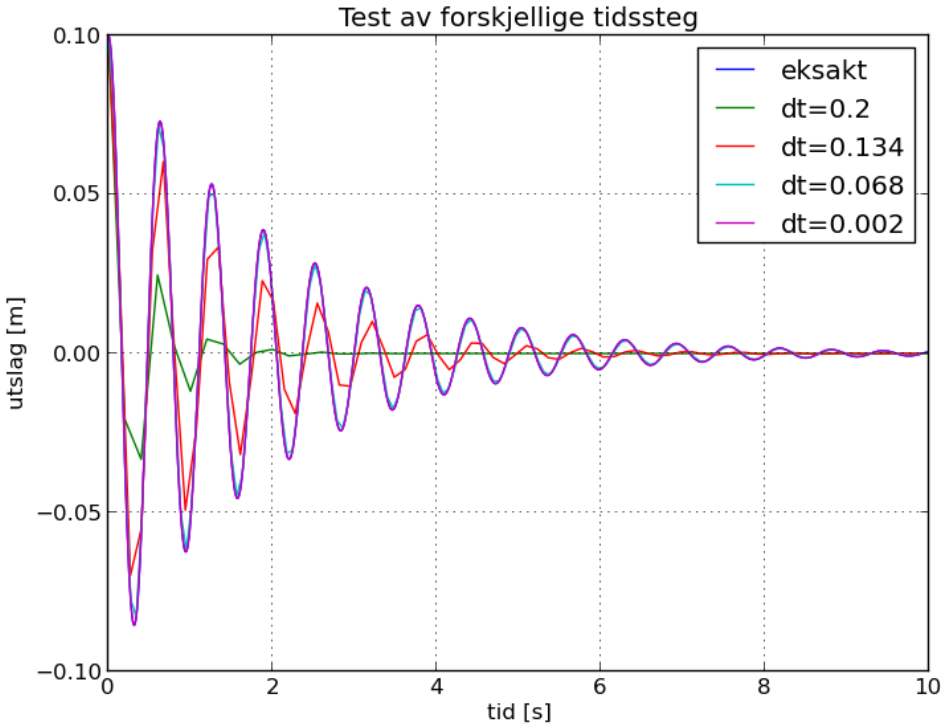
Oppgave 11

I fra de øverste figurene ser vi at bevegelsen er tilnærmet harmonisk, pga. ellipseformen til faserom-kurven og at dalene og toppene til konfigurasjonsrom-kurven er veldig nærme å være parabler. Vi ser også at systemet taper energi over tid

Den nederste figuren

Oppgave C

a) Programmet i sitt fulle er vedlagt på slutten av dokumentet. Resultatet finnes i figur 1.

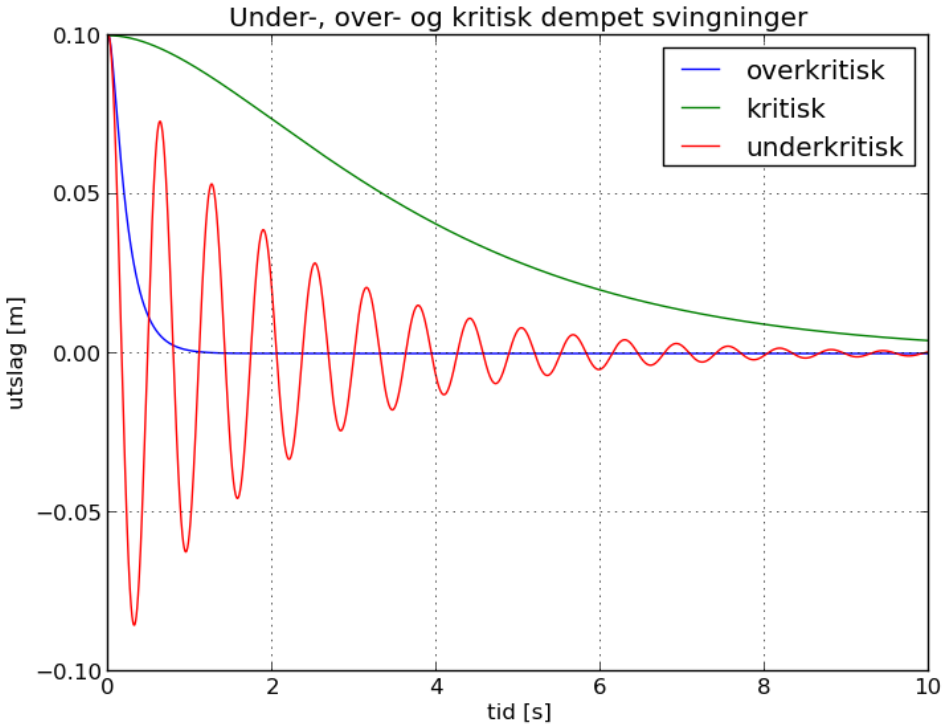


Figur 1: Resultater fra kjøring med forskjellig tidssteg.

b) For overkritisk demping må $b > 2\sqrt{(km)}$. Lar vi $k = 10$ og $m = 0.1$ medfører det at $b > 2$. $b = 2.5$ ble brukt i koden. For kritisk demping er $\gamma = \omega$. Lar vi nå $b = 0.1$ og $m = 0.1$, kan vi løse for k slik at vi oppnår kritisk demping.

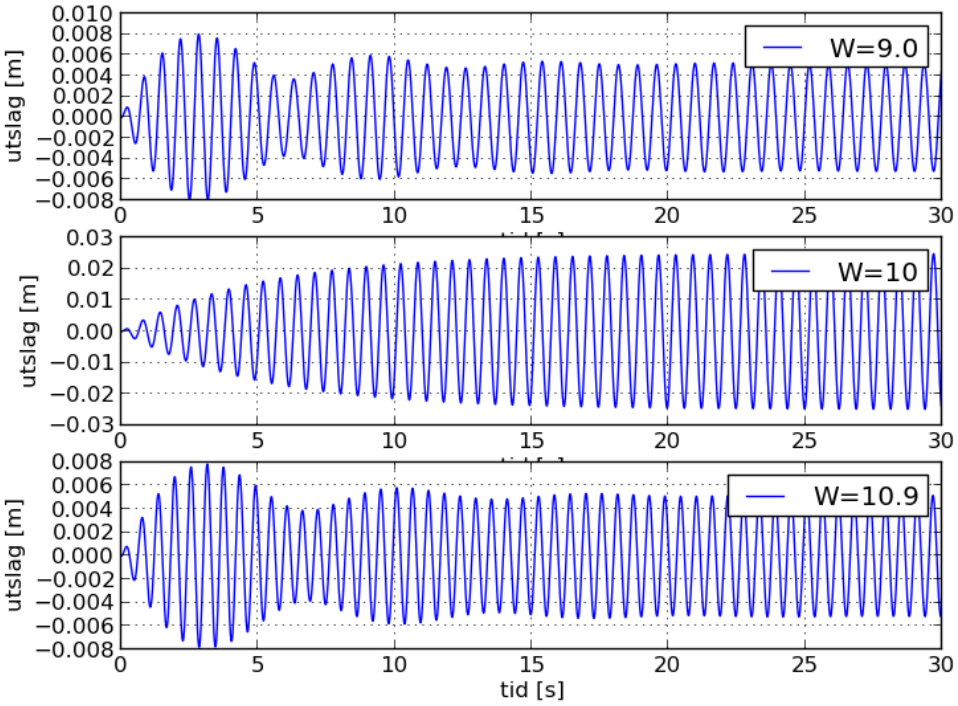
$$\frac{b}{2m} = \sqrt{\frac{k}{m}} \quad \Rightarrow \quad k = \frac{b^2}{4m} = \frac{0.1}{0.4} = 0.025$$

For underkritisk demping er $\gamma < \omega$. $b = 0.1$, $k = 10$ og $m = 0.1$ ble brukt. Resultatene kan sees i figur 2.



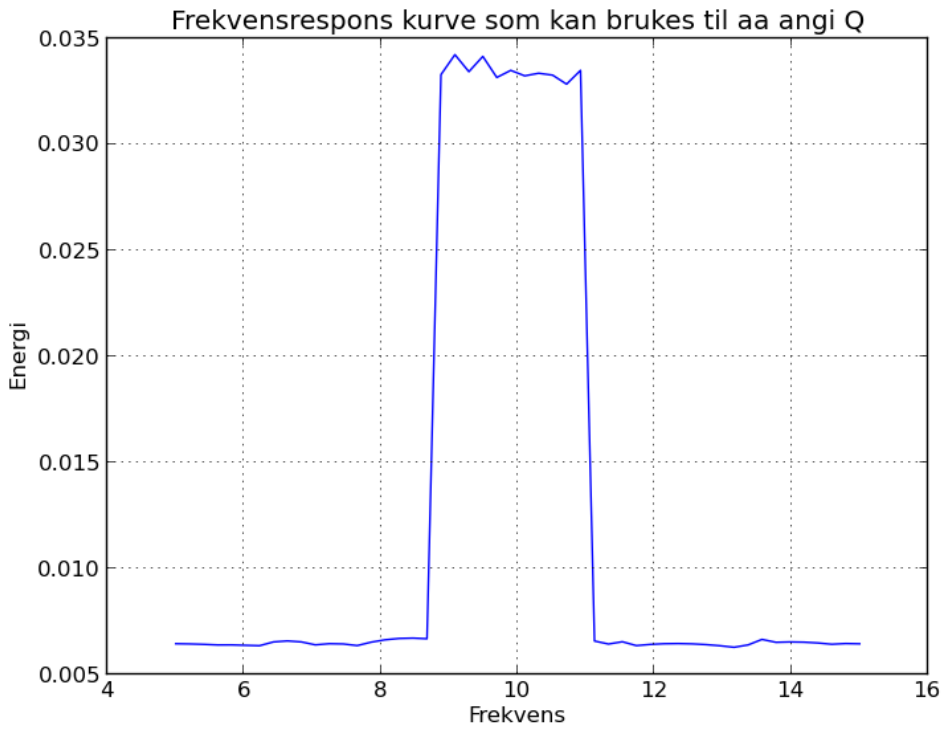
Figur 2: Under-, over- og kritisk demping.

c) En påtrykt frekvens på $\omega = 2\pi f_r es = \sqrt{\frac{k}{m}} = \sqrt{\frac{10}{0.1}} = 10$ ble brukt. Resultatet kan finnes i figur 3.



Figur 3: Tvungene svingninger med forskjellig påtrykt frekvens.

d) Fra figur 4 kan det se ut som halverdibredden er ca. 2.2. Resonansfrekvensen var 10. Kvalitetsfaktoren er da: $Q = \frac{f_0}{\Delta f} = \frac{10}{2.2} \approx 4.5$



Figur 4: Maks energien til fjærpendel som funksjon av påtrykt frekvens.

e) Alt vi trenger å gjøre for å implementere ikke-lineære friksjonsledd, er å legge det aktuelle leddet til diff. likningen. F.eks kan vi legge til leddet:

```
-sign(v)*D*v**2
```

Samnt kanskje oppdatere noen av variabler osv.

Program til oppgave C

```
# -*- coding: utf-8 -*-
from numpy import zeros, cos, sin, sqrt, pi
from pylab import *

class SpringPendulum:

    '''
```

```
Denne klassen implementerer Runge-Kuttas metode av 4. orden for å løse
en 2. ordens ODE på formen: x''(t) + (b/m)*x'(t) + (k/m)*x(t) = 0
'''

def __init__(self, x0, v0, t0, dt, T, b=0.1, k=10, m=0.1, F=0, W=0):
    self.b, self.k, self.m, self.dt, self.F, self.W = b, k, m, dt, F, W
    self.n = int((T-t0)/dt)
    n = self.n
    self.x = zeros(n+1)
    self.v = zeros(n+1)
    self.t = zeros(n+1)
    x, v, t = self.x, self.v, self.t
    x[0], v[0], t[0] = x0, v0, t0

# Løser diff. likningen:
def solve(self):
    for i in xrange(self.n):
        self.x[i+1], self.v[i+1], self.t[i+1] = self.advance(i)
    return self.x, self.v, self.t

# RK4 maskineriet:
def advance(self, i):
    x1, v1, t, dt = self.x[i], self.v[i], self.t[i], self.dt
    a1 = self.f(x1, v1, t)
    x2 = x1 + v1*(dt/2.0)
    v2 = v1 + a1*(dt/2.0)
    a2 = self.f(x2, v2, t + dt/2.0)
    x3 = x1 + v2*(dt/2.0)
    v3 = v1 + a2*(dt/2.0)
    a3 = self.f(x3, v3, t + dt/2.0)
    x4 = x1 + v3*dt
    v4 = v1 + a3*dt
    a4 = self.f(x4, v4, t + dt)
    A = (1/6.0) * (a1 + 2*a2 + 2*a3 + a4)
    V = (1/6.0) * (v1 + 2*v2 + 2*v3 + v4)
    new_x = x1 + V*dt
    new_v = v1 + A*dt
    new_t = t + dt
    return new_x, new_v, new_t

# Bevegelseslikningen til en dempet fjærpendel:
def f(self, x, v, t):
    return -(self.b/self.m)*v - (self.k/self.m)*x + self.F*cos(self.W*t)
```

```
#### Oppgave C.a ####
# Numerisk løsninger med forskjellig dt:
z0 = 0.1
v0 = 0.0
t0 = 0.0
T = 10

DT = linspace(0.2, 0.002, 4)
num = []
for i in xrange(len(DT)):
    num.append(SpringPendulum(z0, v0, t0, DT[i], T, 0.1, 10, 0.1))
    num[i].solve()


# Eksakt løsning:
c1 = 0.005
c2 = 0.1
omega = sqrt(399)/2
t = linspace(t0, T, 1000)
x_exact = zeros(len(t))
for i in xrange(len(t)):
    x_exact[i] = (e**(-t[i]/2))*(c1*sin(omega*t[i]) + c2*cos(omega*t[i]))


# Plot:
figure(1)
plot(t, x_exact, label='eksakt')
hold('on')
for i in xrange(len(num)):
    plot(num[i].t, num[i].x, label='dt='+str(DT[i]))
hold('off')
xlim([t0,T])
ylim([-z0, z0])
xlabel('tid [s]')
ylabel('utslag [m]')
title('Test av forskjellige tidssteg')
grid('on')
legend()
savefig('fig1.png')
```

```
#### Oppgave C.b ####
dt = 0.002
overkritisk = SpringPendulum(z0, v0, t0, dt, T, 2.5, 10, 0.1)
overkritisk.solve()
kritisk = SpringPendulum(z0, v0, t0, dt, T, 0.1, 0.025, 0.1)
kritisk.solve()
underkritisk = SpringPendulum(z0, v0, t0, dt, T, 0.1, 10, 0.1)
underkritisk.solve()

figure(2)
plot(overkritisk.t, overkritisk.x, label='overkritisk')
hold('on')
plot(kritisk.t, kritisk.x, label='kritisk')
plot(underkritisk.t, underkritisk.x, label='underkritisk')
hold('off')
xlim([t0,T])
ylim([-z0, z0])
xlabel('tid [s]')
ylabel('utslag [m]')
title('Under-, over- og kritisk dempet svingninger')
grid('on')
legend()
savefig('fig2.png')



#### Oppgave C.c ####
w_f = 10
W = [0.9*w_f, w_f, 1.09*w_f]
tvungen = []
T = 30
for i in xrange(len(W)):
    tvungen.append(SpringPendulum(0, 0, 0, 0.002, T, 0.04, 10, 0.1, 0.1, W[i]))
    tvungen[i].solve()

figure(3)

for i in xrange(len(W)):
    subplot(3, 1, i+1)
    plot(tvungen[i].t, tvungen[i].x, label='W='+str(W[i]))
    xlim([0,T])
    xlabel('tid [s]')
    ylabel('utslag [m]')
    grid('on')
```

```
    legend()

savefig('fig3.png')


####Oppgave C.d ####
W = linspace(0.5*w_f, 1.5*w_f, 50)
E = zeros(len(W))
k = 10
# Finner E_max for w_f[i]:
for i in xrange(len(W)):
    m = SpringPendulum(0, 0, 0, 0.002, T, 0.04, k, 0.1, 0.1, W[i])
    m.solve()
    E[i] = 0.5*k*abs(m.x[0])
    for j in xrange(m.n):
        if m.x[j+1] > E[i]:
            E[i] = 0.5*k*m.x[j+1]

figure(4)
plot(W,E)
xlabel('Frekvens')
ylabel('Energi')
title('Frekvensrespons kurve som kan brukes til aa angi Q')
grid('on')
savefig('fig4.png')
```