Universitetet i Oslo

Svingninger og bølger

FYS2130

Oblig 3

Forfatter:
Eirik Lund

Studentkode:
eaflund

14. februar 2017

1 :

Runge-Kutta har en høyere presisjon fordi den evaluerer stigningstallet ved 4 forskjellige estimater.
Først ved begynnelsen, to i midten og en ved slutten, Euler gjør kun vurdering på halvveien.
Runge-Kutta får også med seg energibevaring for periodiske systemer, noe Euler ikke gjør.

Dette medfører ekstreme utslag i posisjon og energi ved å gjøre endringer i fjærkonstanten hvis man bruker Euler, mens Runge-Kutta klarer å kompansere for dette, derfor er Runge-Kutta en bedre numeriskmetode enn Euler.

2 :

Første bildet til venstre så har vi et plot med en ordinær pendel som med dempede svingninger. Vi ser også på faserommet for tilhørende system at amplituden er synkende.

For bildet under så har vi et system som har tilført for stor hastighet. Vi ser at faserommet bruker tid på å normalisere seg.

3 :

a) Oppgitte variabler :

$m = 0.1kg$

$k = 10\frac{N}{m}$

$b = 0.1\frac{kg}{s}$

$z(0) = 0.1m$

$\frac{dz}{dt} = 0\frac{m}{s}$

Vi har $\sum F = ma = m\ddot{z}$

$F_f = -bv - Dv^2$ hvor D = 0.

Vi kan da forme ligningen til : $m\ddot{z} + b\dot{z} + kz$

$\omega^2 = \frac{k}{m}$ og $\frac{b}{m} = 2\gamma$

Med oppgitte variabler så får vi at $\omega = 10$ og $\gamma = \frac{1}{2}$

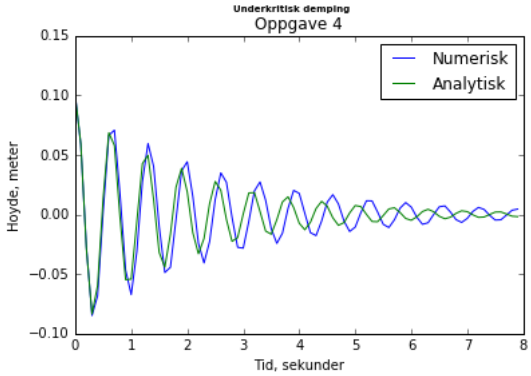
Vi observerer at $\gamma < \omega$ så det vil si at vi har en underkritisk demping.

Vi bruker ligningen : $z(t) = e^{-\gamma t} A cos(\omega' t + \phi)$

Hvor $A = z(0) = 10cm$, $\phi = 0$ og $\omega' = \sqrt{\omega^2 - \gamma^2}$

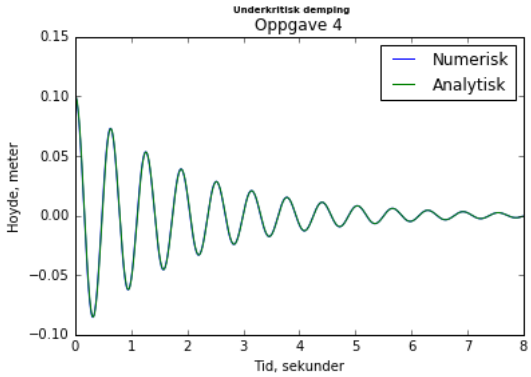
Bruker 8 sekunder som tidslengde.

Her er et plot med $\Delta t = \frac{1}{10}$

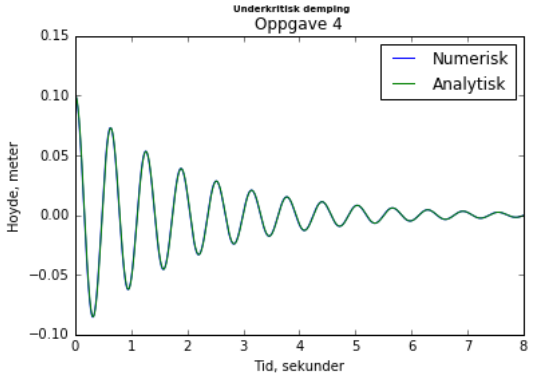


Som vi ser fra plottet så er tidssteget for lite og vi kommer ut av fase med den analytiske.

Med en $\Delta t = \frac{1}{100}$ så fikk vi et mye bedre samsvar med den analytiske.



Siste jeg teste for $\Delta t = \frac{1}{1000}$ som treffer så si eksakt som den analytiske løsningen.



b) Fortsette med å bruke en $\Delta t = \frac{1}{1000}$
Det som bestemmer om vi har under,over eller kritisk - demping er som følgende:

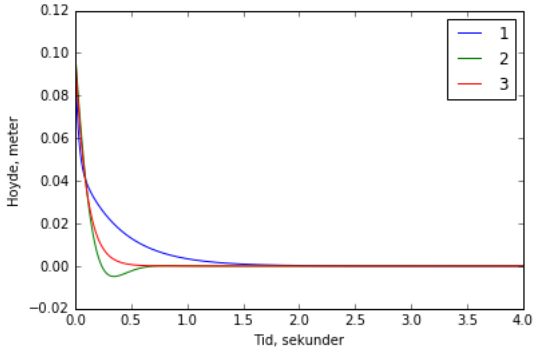
$\gamma < \omega$ (under) , $\gamma > \omega$ (over) og $\gamma = \omega$ (kritisk).

Finner kritisk ved å løse $\gamma = \omega$

$$\frac{b}{2m} = \sqrt{\frac{k}{m}}, b = \sqrt{4km} = 2.0$$

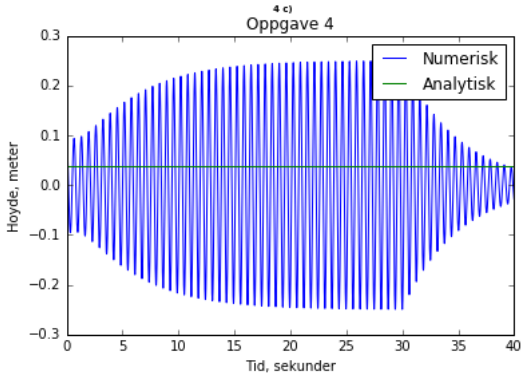
Vi ser ved ulike verdier for b så får vi et plot som illustrerer fint ulike dempinger.

$b = 1$ er underkritisk, $b = 2$ er kritisk og $b = 3$ er overkritisk

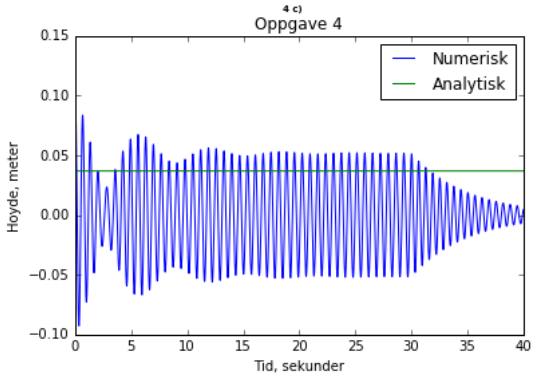


c) Vi inkluderer $v_0 cos(\omega t)$ så får vi at $\omega_f = \omega$
Jeg har bedt programmet å skru av påtrykte kraften ved 30 sekunder.

Grønne linjen er amplituden gangen med $\frac{1}{e}$



Her har jeg satt $\omega_f = \frac{9}{10}\omega$



Vi ser for begge plottene at svingningene dør etter den påtrykte kraften er skrudd av.

$$Q = \sqrt{\frac{mk}{b^2}} = 25.0$$

d) Løser denne oppgaven numerisk, se plot nedenfor.

Vedlagt script.

```
# -*- coding: utf-8 -*-
"""
Created on Mon Feb 13 21:59:14 2017

@author: ealun
"""

import numpy as np
import matplotlib.pyplot as plt
from numpy.lib.scimath import sqrt as csqrt

T = 30 #seconds
m = 0.1 #kg
k = 10. #N/m
L = 0 # l_0
V0 = 0.1
D = 0.0

h = 6.62607004e-34 #planck


omega = np.sqrt(k/m)
gamma = 0.5*(b/m)
omegamerket = np.sqrt(omega**2 - gamma**2)

gammaover = 0.5*(over/m)
gammaunder = 0.5*(under/m)
gammakritisk = 0.5*(kritisk/m)

omgov = csqrt(omega**2 - gammaover**2)
omgun = np.sqrt(omega**2 - gammaunder**2)
omgkr = np.sqrt(omega**2 - gammakritisk**2)



if gamma > omega:
    disp = str('Overkritisk demping')

if gamma == omega:
    disp = str('Kritisk demping')
```

```
if gamma < omega:
    disp = str('Underkritisk demping')


def RK4(T,dt,a): #RK4 code from book

    x0 = np.array([0,0.1]) #10 cm
    v0 = np.array([0,0]) # 0 m/s

    time = int(float(T)/dt)

    t = np.zeros(time)
    x = np.zeros((time,np.size(x0)))
    v = np.zeros((time,np.size(x0)))



    x[0] = x0
    v[0] = v0


    for i in xrange(0,int(time)-1):

        v1 = v[i]
        a1 = a(t[i],x[i],v[i])

        v2 = v[i] + a1*dt/2.0
        a2 = a(t[i] + 0.5*dt, x[i] + v2*dt*0.5, v2)

        v3 = v[i] + a2*dt/2.0
        a3 = a(t[i] + 0.5*dt, x[i] + v3*dt*0.5, v3)

        v4 = v[i] + a3*dt
        a4 = a(t[i] + dt, x[i] + v3*dt, v4)

        ahalf = 1.0/6.0*(a1 + 2*a2 + 2*a3 + a4 )
        vhalf = 1.0/6.0*(v1 + 2*v2 + 2*v3 + v4 )


        t[i+1] = t[i] + dt
        x[i+1] = x[i] + vhalf*dt
        v[i+1] = v[i] + ahalf*dt

    return t,x,v
```

```
def a(t,x,v):

    if t < 30: #skru av pAtrykt kraft etter 30 sekunder
        F_t = V0*np.cos(omega*t) #*0.9

    else:
        F_t = 0

    F =  k*(L-x) - b*v -D*v**2

    return (F+F_t)/m


def plotter():
    phi = 0
    fig = plt.figure()
    fig.suptitle(disp , fontsize=7, fontweight='bold')

    plt.plot(t,x[:,1])
    plt.hold('on')
    plt.plot(t,np.exp(-gamma*t)*x[0,1]*np.cos(omegamerket*t + phi))

    plt.xlabel("Tid, sekunder")
    plt.ylabel("Hoyde, meter")

    plt.title("Oppgave 4")
    plt.legend(['Numerisk', 'Analytisk',disp])
    plt.show()

    """ Oppgave 4 b)
    """

    fig = plt.figure()

    plt.plot(t,np.exp(-gammaover*t)*x[0,1]*np.cos(omgov*t + phi)) #
        analytisk overkritisk
    plt.hold('on')
    plt.plot(t,np.exp(-gammaunder*t)*x[0,1]*np.cos(omgun*t + phi)) #
        under
    plt.hold('on')
    plt.plot(t,np.exp(-gammakritisk*t)*x[0,1]*np.cos(omgkr*t + phi)) #
        kritisk
```

```
    plt.xlabel("Tid, sekunder")
    plt.ylabel("Hoyde, meter")

    plt.legend([under, kritisk, over])
    plt.show()

    """ Oppgave 4 c)
    """

    fig = plt.figure()
    fig.suptitle('4 c)', fontsize=7, fontweight='bold')

    plt.plot(t,x[:,1])
    plt.hold('on')
    funksjon = np.ones(np.size(t))*1/np.exp(1)*x[0,1]
    plt.plot(t,funksjon)

    plt.xlabel("Tid, sekunder")
    plt.ylabel("Hoyde, meter")

    plt.title("Oppgave 4")
    plt.legend(['Numerisk', '$1/e$'])
    plt.show()


    return plotter


if __name__ == '__main__':

    #Test variables
    dt = 0.001
    b = 0.04 #kg/S

    under = 1
    kritisk = 2
    over = 3

    t,x,v = RK4(T,dt,a)
    plotter = plotter()

    omegas = np.linspace(5,15,11)
    Emax = np.zeros_like(omegas)

    for i in xrange(0,len(omegas)):
```

```
        t,x,v = RK4(T,dt,a)


        omega = omegas[i]

        Emax[i] = np.max(0.5*10*np.max(x[0,:]))


    plt.figure()
    plt.plot(omegas/(2*np.pi),Emax)

    plt.show()
```