Prosjektoppgave besvarelse

Kadnidatnummer: 15267

May 7, 2018

Oppgave 1

Ved å løse differensial likningen

$$m\ddot{x}(t) + kx(t) = 0, \tag{1}$$

og plotte systemets bane i faserommet så får vi plottet i figur 1

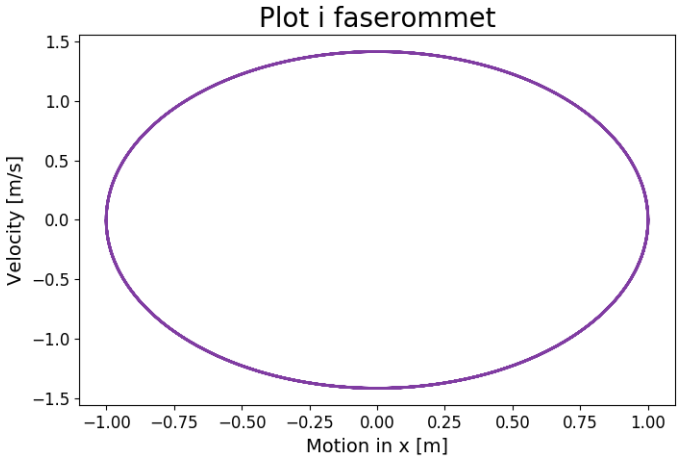

Figure 1: viser faseplottet for en fri harmonisk oscillator

Hvis vi varier initialverdiene for posisjonen og hastigheten, så vil det fortsatt være en ellipse. Hvis vi ser på totalenergien til systemet så ser vi hvorfor dette stemmer. Totalenergien er

$$
\begin{aligned}
E_{tot} &= E_k + E_p \\
E_{tot} &= \frac{1}{2}m\dot{x}^2 + \frac{1}{2}kx^2 \\
\frac{2E_{tot}}{mk} &= \frac{\dot{x}^2}{k} + \frac{x^2}{m}, \\
\textit{Hvis vi setter } C &= \frac{2E_{tot}}{mk} \\
\Rightarrow \frac{x^2}{m} + \frac{\dot{x}^2}{k} &= C \\
\frac{x^2}{mC} + \frac{\dot{x}^2}{kC} &= C \\
\frac{x^2}{\sqrt{mC}^2} + \frac{\dot{x}^2}{\sqrt{kC}^2} &= 1
\end{aligned}
$$

Det er formelen for en ellipse.

Der hvor banen krysser x aksen og y aksen så er den vinkelrett på den aksen den krysser. Dette skyldes at i de punktene så er totalenergien lokalisert i bare det ene uttrykket, og det andre er null. Ved $|x| = max$ så endrer bevegelsen retning og $E_{tot} = E_p$. Når $x = 0$ så er $E_{tot} = E_k$

Hvis vi plotter en kortere tidsperiode slik at kurven ikke når til startposisjonen så ser vi at den beveger seg med klokken. Selv om vi varierer startposisjon og starthastighet så er det ikke mulig å få banen til å gå i motsatt retning på grunn av hvordan vi har satt positive retninger og hva vi har satt på x og y aksen.

Oppgave 2

Hvis vi inkluderer dempning i programmet så vil banen i faseplottet gå gradvis mot punktet $x = 0m$ og $\dot{x} = 0m/s$ som en spiral.

Banen står vinkelrett bare når den krysser x aksen, ikke y aksen. Dette

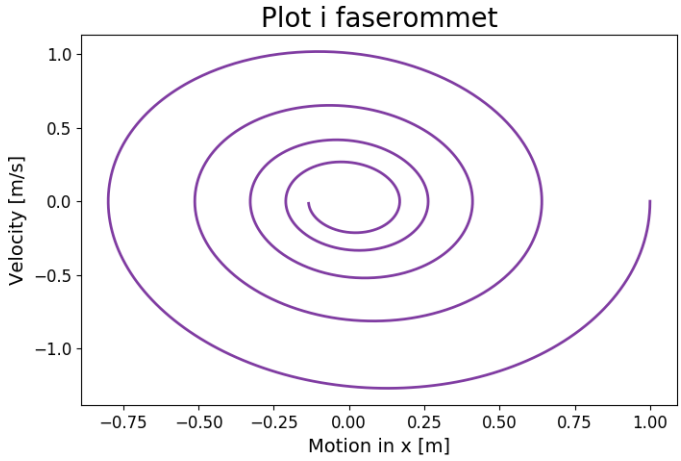

Figure 2: viser faseplott for svingning med dempning

er på grunn av dempings leddet som vi satt inn.

En attraktor er et set med numeriske verdier som et dynamisk system har en tendens til å utvikle seg mot for et bredt utvalg av initialbetingelser. En attraktor kan være et punkt(0 dim), et set met punkter, en kurve(2 dim), og flere ting innenfor matematikk. Punktet $x = 0m$ og $\dot{x} = 0m/s$ er en attraktor av dimensjon 0, fordi det er et singulært punkt. Kurven i oppgave 1 er også en attraktor i følge definisjonen, siden systemet alltid havner i den kurven uansett initialbetingelser og er av dimensjon 1.

Oppgave 3

Vi skal løse følgende likning alanytisk.

$$m\ddot{x}(t) + kx(t) = F_D cos(\omega_D t) \tag{2}$$

Begynner med å finne den generelle løsningen for den homogene likningen, og løser derfor den karakteristiske likningen $m\lambda^2 + k = 0$

$$
\begin{aligned}
m\lambda^2 + k &= 0 \\
\lambda^2 &= \frac{-k}{m} \\
\lambda &= \pm\sqrt{\frac{-k}{m}}
\end{aligned}
$$

Hvis vi har λ på formen $\lambda_{\pm} = \mu \pm i\eta$, så får vi den generelle løsningen

$$
\begin{aligned}
\lambda &= \pm i\sqrt{\frac{k}{m}} \\
x_h &= e^{\mu t}(Acos(\eta t) + Bsin(\eta t)) \\
x_h &= e^{0t}\left(Acos\left(\sqrt{\frac{k}{m}}t\right) + Bsin\left(\sqrt{\frac{k}{m}}t\right)\right) \\
x_h &= \underline{Acos\left(\sqrt{\frac{k}{m}}t\right) + Bsin\left(\sqrt{\frac{k}{m}}t\right)}
\end{aligned}
$$

Så den generelle løsningen for den homogene delen er

$$
x_h = Acos\left(\sqrt{\frac{k}{m}}t\right) + Bsin\left(\sqrt{\frac{k}{m}}t\right) \tag{3}
$$

Nå skal vi bruke metoden for ubestemte koeffisienter for å finne den partikulære løsningen. Siden høyre side av likningen er på formen $Acos(pt)$, så kan vi bruke x_p på formen Ce^{ipt}

$$
\begin{aligned}
x_p &= Ce^{ipt}, \quad p = \omega_D, fra\ \ likningen \\
x_p &= Ce^{i\omega_D t} \\
\dot{x_p} &= iC\omega_D e^{i\omega_D t} \\
\ddot{x_p} &= -C\omega_D^2 e^{i\omega_D t}
\end{aligned}
$$

Dette setter vi inn i difflikningen og løser for C

$$
\begin{aligned}
m\ddot{x} + kx &= F_D cos(\omega_D t) \\
m\left(-C\omega_D^2 e^{i\omega_D t}\right) + kCe^{i\omega_D t} &= F_D cos(\omega_D t) \\
-mC\omega_D^2 e^{i\omega_D t} + kCe^{i\omega_D t} &= F_D cos(\omega_D t) \\
Ce^{i\omega_D t}\left(-m\omega_D^2 + k\right) &= F_D cos(\omega_D t) \\
C &= \frac{F_D cos(\omega_D t)}{e^{i\omega_D t}\left(-m\omega_D^2 + k\right)}
\end{aligned}
$$

Så setter vi uttrykket for C inn i x_p og får at

$$x_p = \frac{F_D cos(\omega_D t)}{(k - m\omega_D^2)} \tag{4}$$

Så den fullstendige generelle løsningen er

$$Acos\left(\sqrt{\frac{k}{m}}t\right) + Bsin\left(\sqrt{\frac{k}{m}}t\right) + \frac{F_D cos(\omega_D t)}{(k - m\omega_D^2)} \tag{5}$$

Vi skulle løse den for $x(0) = 2m$ og $\dot{x}(0) = 0m/s$, så da gjør vi det og får

$$
\begin{aligned}
A &= 2 - \frac{F_D}{k - m\omega_D^2} \\
B &= 0
\end{aligned}
$$

Dette setter vi inn i løsningen og får

$$x(t) = 2cos\left(\sqrt{\frac{k}{m}}t\right) + \frac{F_D}{k - m\omega_D^2}\left(cos(\omega_D t) - cos\left(\sqrt{\frac{k}{m}}t\right)\right) \tag{6}$$

Så bruker vi en trigonometrisk identitet funnet på side 42 i Rottmann, som ser slik ut

$$cos\alpha - cos\beta = -2sin\frac{\alpha + \beta}{2}sin\frac{\alpha - \beta}{2} \tag{7}$$

og da får vi

$$x(t) = 2cos\left(\sqrt{\frac{k}{m}}t\right) - \frac{2F_D}{k - m\omega_D^2}\left(sin\frac{\left(\omega_D + \sqrt{\frac{k}{m}}\right)t}{2} sin\frac{\left(\omega_D - \sqrt{\frac{k}{m}}\right)t}{2}\right) \tag{8}$$

noe

Oppgave 4

Ved å løse likning 2 nummerisk og plotte den nummeriske løsningen minus den analytiske så får vi.

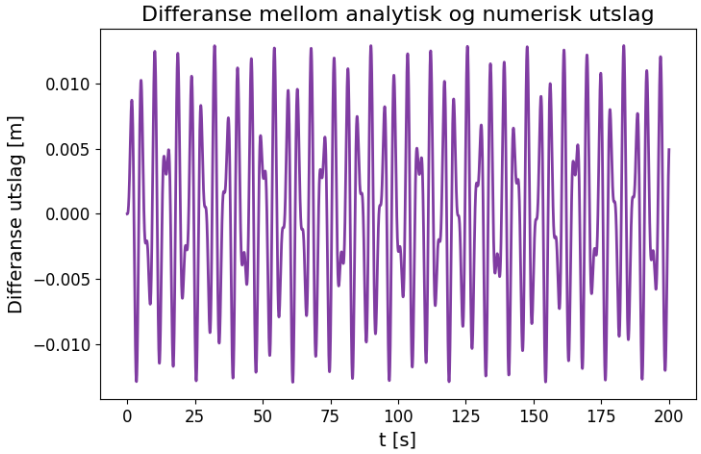

Figure 3: viser differansen mellom den nummeriske og analytiske løsningen

Så det nummeriske resultatet er ganske likt det analytiske. Plotter vi bevegelsen i faserommet får vi figur 4
Bevegelsen i figur 4 er periodisk. Hvis vi bytter ut drivefrekvensen med $\omega_D = \frac{2}{(\sqrt{5}-1)}\omega_0$ så får vi
Bevegelsen i figur 5 også en pariodisk bevegelse.

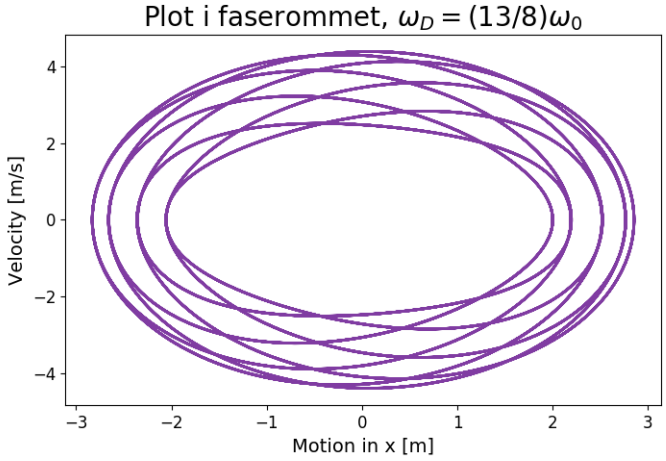

Figure 4: viser faseplottet med påtrykt kraft og $\omega_D = \frac{13}{8}\omega_0$

Oppgave 5

mer ting
ting

Oppgave 6

Oppgave 7

Oppgave 8

Oppgave 9

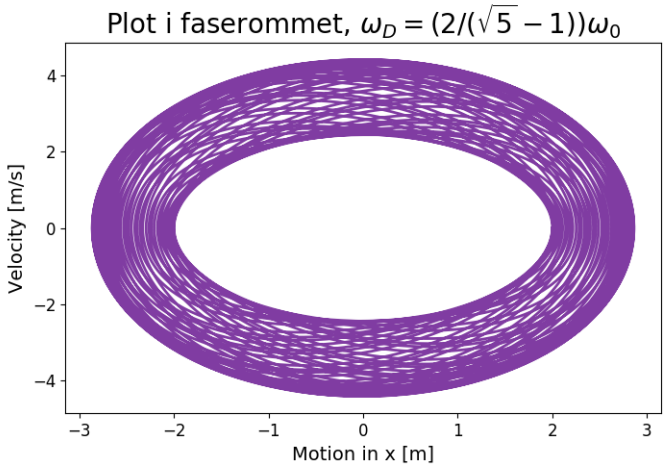

Figure 5: viser faseplottet med påtrykt kraft og $\omega_D = \frac{2}{(\sqrt{5}-1)}\omega_0$

Appendix

RK4.py

```
"""
The function for the Runge Kutta method

"""

# The Runge Kutta method
def RK4(diffEQ,xStart,vStart,tStart,dt):
    a1 = diffEQ(xStart,vStart,tStart)
    v1 = vStart

    xHalf1 = xStart + v1 * dt/2.0
    vHalf1 = vStart + a1 * dt/2.0

    a2 = diffEQ(xHalf1,vHalf1,tStart+dt/2.0)
    v2 = vHalf1

    xHalf2 = xStart + v2 * dt/2.0
    vHalf2 = vStart + a2 * dt/2.0

    a3 = diffEQ(xHalf2,vHalf2,tStart+dt)
    v3 = vHalf2

    xEnd = xStart + v3 * dt
    vEnd = vStart + a3 * dt
```

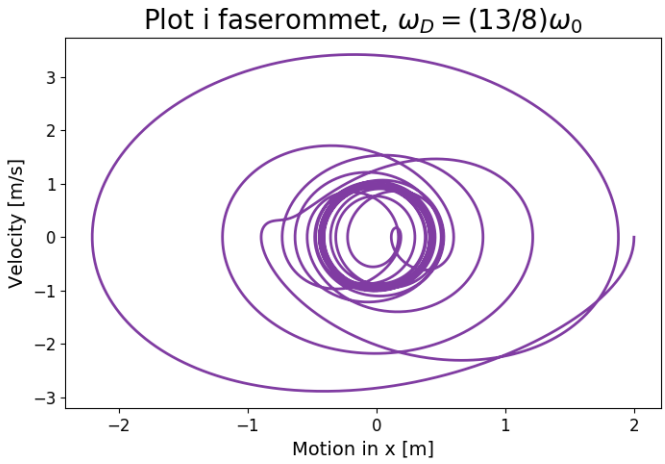

Figure 6: viser faseplottet med påtrykt kraft, dempning og $\omega_D = \frac{13}{8}\omega_0$

```
    a4 = diffEQ(xEnd,vEnd,tStart+dt)
    v4 = vEnd

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

    xEnd = xStart + vMiddle*dt
    vEnd = vStart + aMiddle*dt

    return xEnd, vEnd
```

oppgave1.py

```
#imports
import numpy as np
import matplotlib.pyplot as plt
from RK4 import RK4
import seaborn


#declare constants
dt = 10.**-2          #some timestep [s]
m = 0.500         #mass of 500 g [kg]
k = 1.            #stiffness constant [N/m]
x0 = 1.               #start position [m]
v0 = 0.               #start velocity [m/s]
T = 20.0              #total time [s]
N = int(T/dt)
```

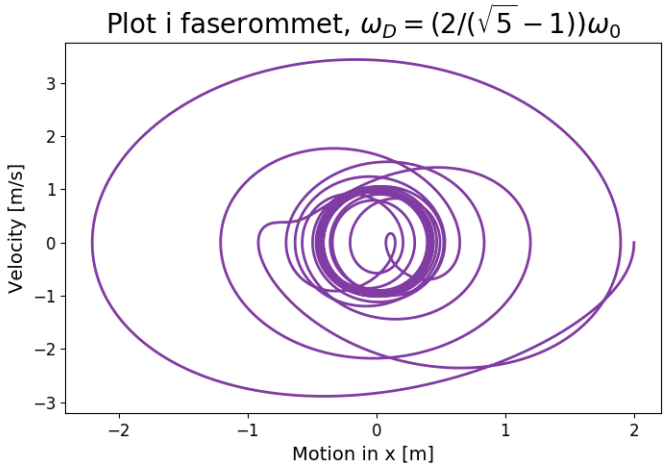

Figure 7: viser faseplottet med påtrykt kraft, dempning og $\omega_D = \frac{2}{(\sqrt{5}-1)}\omega_0$

```
#setting initialconditions
x = np.zeros(N); v = np.zeros(N); t = np.zeros(N)
x[0] = x0; v[0] = v0

"""
the function we have is
mx''(t) + kx(t) = 0
x''(t) = kx(t)/m
"""
#making functions

def diffEQ(xNow,vNow,tNow):
    aNow = -k*xNow/m
    return aNow

#running the loop
for i in range(N-1):
    x1 = x[i]; v1 =v[i]
    x[i+1],v[i+1] = RK4(diffEQ,x1,v1,t,dt)
    t[i+1] = t[i] + dt



#plotting
```

```
#plotting the motion in x against time
plt.figure(figsize=(8,5))
plt.plot(t,x, '#803CA2', linewidth=2.0)
plt.title('Plot av utslag', fontsize=20)
plt.xlabel("Time [s]", fontsize=14)
plt.ylabel("Motion in x [m]", fontsize=14)
plt.tick_params(axis = 'both', which = 'major', labelsize =
    12)
#plt.savefig('Oppgave1utslag.png')
plt.show()

#plotting the phaseplot
plt.figure(figsize=(8,5))
plt.plot(x,v, '#803CA2', linewidth=2.0)
plt.title('Plot i faserommet', fontsize=20)
plt.xlabel("Motion in x [m]", fontsize=14)
plt.ylabel("Velocity [m/s]", fontsize=14)
plt.tick_params(axis = 'both', which = 'major', labelsize =
    12)
#plt.savefig('Oppgave1fase.png')
plt.show()
```

oppgave2.py

```
#imports
import numpy as np
import matplotlib.pyplot as plt
from RK4 import RK4
import seaborn



#declare constants
dt = 10.**-2          #some timestep [s]
m = 0.500             #mass of 500 g [kg]
k = 1.                #stiffness constant [N/m]
x0 = 1.               #start position [m]
v0 = 0.               #start velocity [m/s]
T = 20.               #total time [s]
N = int(T/dt)         #number of things
b = 0.1               #dampening constant[kg/s]


#setting initialconditions
x = np.zeros(N); v = np.zeros(N); t = np.zeros(N)
x[0] = x0; v[0] = v0

"""
the function we have is
```

```
mx''(t) +bx'(t) + kx(t) = 0
x''(t) = -bx'(t)/m -kx(t)/m
"""
#making functions

def diffEQ(xNow,vNow,tNow):
    aNow = -b*vNow/m-k*xNow/m
    return aNow

#running the loop
for i in range(N-1):
    x1 = x[i]; v1 =v[i]
    x[i+1],v[i+1] = RK4(diffEQ,x1,v1,t,dt)
    t[i+1] = t[i] + dt



#plotting

#plotting the motion in x against time
plt.figure(figsize=(8,5))
plt.plot(t,x, '#803CA2', linewidth=2.0)
plt.title('Plot av utslag', fontsize=20)
plt.xlabel("Time [s]", fontsize=14)
plt.ylabel("Motion in x [m]", fontsize=14)
plt.tick_params(axis = 'both', which = 'major', labelsize =
    12)
#plt.savefig('Oppgave2utslag.png')
plt.show()

#plotting the phaseplot
plt.figure(figsize=(8,5))
plt.plot(x,v, '#803CA2', linewidth=2.0)
plt.title('Plot i faserommet', fontsize=20)
plt.xlabel("Motion in x [m]", fontsize=14)
plt.ylabel("Velocity [m/s]", fontsize=14)
plt.tick_params(axis = 'both', which = 'major', labelsize =
    12)
#plt.savefig('Oppgave2fase.png')
plt.show()
```

oppgave4.py

```
#imports
import numpy as np
import matplotlib.pyplot as plt
from RK4 import RK4
import seaborn
```

```
#declare constants
dt = 10.**-2          #some timestep [s]
m = 0.500             #mass of 500 g [kg]
k = 1.                #stiffness constant [N/m]
x0 = 2.               #start position [m]
v0 = 0.               #start velocity [m/s]
T = 200.              #total time [s]
N = int(T/dt)         #number of things
omega0 = np.sqrt(k/m)        #Svingefrekvens for HO
F_D = 0.7             #[N]
#omega_D = (13./8)*omega0
omega_D = (2./((np.sqrt(5)-1)))*omega0


#setting initialconditions
x = np.zeros(N); v = np.zeros(N); t = np.zeros(N)
x[0] = x0; v[0] = v0
F = np.zeros(N)
F[0] = F_D



"""
the function we have is
mx''(t) + kx(t) = F(t)
x''(t) = F(t)/m -kx(t)/m
"""
#making functions

def diffEQ(xNow,vNow,tNow):
    aNow = (F[i]-k*xNow)/m
    return aNow

#running the loop
for i in range(N-1):
    x1 = x[i]; v1 =v[i]
    F[i] = F_D *np.cos(omega_D*t[i])
    x[i+1],v[i+1] = RK4(diffEQ,x1,v1,t,dt)
    t[i+1] = t[i] + dt


#The analytic solution

def anal_x_solution(t):
    return 2*np.cos(np.sqrt(k/m)*t) + F_D/(k-m*omega_D**2)*(
       np.cos(omega_D*t)-np.cos(np.sqrt(k/m)*t))
```

```
#plotting

#plotting the motion in x against time
plt.figure(figsize=(8,5))
plt.plot(t,x, '#803CA2', linewidth=2.0)
plt.title('Plot av utslag $\omega_D = (2/(\sqrt{5}-1))\
    omega_0$', fontsize=20)
plt.xlabel("Time [s]", fontsize=14)
plt.ylabel("Motion in x [m]", fontsize=14)
plt.tick_params(axis = 'both', which = 'major', labelsize =
    12)
#plt.savefig('oppgave4utslagdel2.png')
plt.show()

#plotting the phaseplot
plt.figure(figsize=(8,5))
plt.plot(x,v, '#803CA2', linewidth=2.0)
plt.title('Plot i faserommet, $\omega_D = (2/(\sqrt{5}-1))\
    omega_0$', fontsize=20)
plt.xlabel("Motion in x [m]", fontsize=14)
plt.ylabel("Velocity [m/s]", fontsize=14)
plt.tick_params(axis = 'both', which = 'major', labelsize =
    12)
#plt.savefig('Oppgave4fasedel2.png')
plt.show()


#plotting the difference between the numerical and analytical
    result
plt.figure(figsize=(8,5))
plt.plot(t, x-anal_x_solution(t), '#803CA2', linewidth=2.0)
plt.xlabel('t [s]', fontsize=14)
plt.ylabel('Differanse utslag [m]',fontsize=14)
plt.title('Differanse mellom analytisk og numerisk utslag',
    fontsize=16)
plt.tick_params(axis = 'both', which = 'major', labelsize =
    12)
#plt.savefig('Oppgave4_differanse.png')
plt.show()
```

oppgave5.py

```
#imports
import numpy as np
import matplotlib.pyplot as plt
from RK4 import RK4
import seaborn
```

```
#declare constants
dt = 10.**-2          #some timestep [s]
m = 0.500             #mass of 500 g [kg]
k = 1.                #stiffness constant [N/m]
x0 = 2.               #start position [m]
v0 = 0.               #start velocity [m/s]
T = 100.              #total time [s]
N = int(T/dt)         #number of things
b = 0.1               #[kg/s]
omega0 = np.sqrt(k/m)       #Svingefrekvens for HO
F_D = 0.7             #[N]
#omega_D = (13./8)*omega0
omega_D = (2./((np.sqrt(5)-1)))*omega0

#setting initialconditions
x = np.zeros(N); v = np.zeros(N); t = np.zeros(N)
x[0] = x0; v[0] = v0
F = np.zeros(N)
F[0] = F_D



"""
the function we have is
mx''(t) +bx'(t) + kx(t) = F(t)
x''(t) = (F(t) -bx'(t) -kx(t))/m
"""
#making functions

def diffEQ(xNow,vNow,tNow):
    aNow = (F[i]-b*vNow-k*xNow)/m
    return aNow

#running the loop
for i in range(N-1):
    x1 = x[i]; v1 =v[i]
    F[i] = F_D *np.cos(omega_D*t[i])
    x[i+1],v[i+1] = RK4(diffEQ,x1,v1,t,dt)
    t[i+1] = t[i] + dt


#plotting

#plotting the motion in x against time
plt.figure(figsize=(8,5))
plt.plot(t,x, '#803CA2', linewidth=2.0)
plt.title('Plot av utslag, $\omega_D = (2/(\sqrt{5}-1))\
   omega_0$', fontsize=20)
```

```
plt.xlabel("Time [s]", fontsize=14)
plt.ylabel("Motion in x [m]", fontsize=14)
plt.tick_params(axis = 'both', which = 'major', labelsize =
    12)
#plt.savefig('Oppgave5utslag2.png')
plt.show()

#plotting the phaseplot
plt.figure(figsize=(8,5))
plt.plot(x,v, '#803CA2', linewidth=2.0)
plt.title('Plot i faserommet, $\omega_D = (2/(\sqrt{5}-1))\
    omega_0$', fontsize=20)
plt.xlabel("Motion in x [m]", fontsize=14)
plt.ylabel("Velocity [m/s]", fontsize=14)
plt.tick_params(axis = 'both', which = 'major', labelsize =
    12)
#plt.savefig('Oppgave5fase2.png')
plt.show()
```

oppgave6.py

```
#imports
import numpy as np
import matplotlib.pyplot as plt
from RK4 import RK4
import seaborn


#declare constants
dt = 10.**-4          #some timestep [s]
m0 = 0.00001          #mass of waterdrop [kg]
k = 0.475             #stiffness constant [N/m]
x0 = 0.001            #start position [m]
v0 = 0.001            #start velocity [m/s]
T = 3.           #total time [s]
N = int(T/dt)         #number of things
b = 0.001             #[kg/s]
g = 9.81              #gravitational acceleration [m/s^2]
psi = 0.00055         #m'(t) [kg/s]
dpsi =T*psi/N         #mass gain step

#setting initialconditions
x = np.zeros(N); v = np.zeros(N); t = np.zeros(N)
x[0] = x0; v[0] = v0
m = np.zeros(N)
m[0] = m0


"""
```

```
the functions we have are
m(t)x''(t)+(b+psi)x'(t)+kx(t)=m(t)g
x''(t) = (m(t)g - (b+psi)x'(t) - kx(t))/m(t)
"""
#making functions

def diffEQ(xNow,vNow,tNow):
    aNow = (m[i]*g-(b+psi)*vNow-k*xNow)/m[i]
    return aNow

#running the loop
for i in range(N-1):
    x1 = x[i]; v1 =v[i]
    x[i+1],v[i+1] = RK4(diffEQ,x1,v1,t,dt)
    t[i+1] = t[i] + dt
    m[i+1] = m[i] + dpsi


#plotting

#plotting the motion in x against time
plt.figure(figsize=(8,5))
plt.plot(t,x, '#803CA2', linewidth=2.0)
plt.title('Plot av utslag', fontsize=20)
plt.xlabel("Time [s]", fontsize=14)
plt.ylabel("Motion in x [m]", fontsize=14)
plt.tick_params(axis = 'both', which = 'major', labelsize =
    12)
#plt.savefig('Oppgave6utslag.png')
plt.show()

#plotting the phaseplot
plt.figure(figsize=(8,5))
plt.plot(x,v, '#803CA2', linewidth=2.0)
plt.title('Plot i faserommet', fontsize=20)
plt.xlabel("Motion in x [m]", fontsize=14)
plt.ylabel("Velocity [m/s]", fontsize=14)
plt.tick_params(axis = 'both', which = 'major', labelsize =
    12)
#plt.savefig('Oppgave6fase.png')
plt.show()
```

oppgave7.py

```
#imports
import numpy as np
import matplotlib.pyplot as plt
from RK4 import RK4
import seaborn
```

```
#declare constants
dt = 10.**-4          #some timestep [s]
m0 = 0.00001          #mass of waterdrop [kg]
k = 0.475             #stiffness constant [N/m]
x0 = 0.001            #start position [m]
v0 = 0.001            #start velocity [m/s]
T = 20.               #total time [s]
N = int(T/dt)         #number of things
b = 0.001             #[kg/s]
g = 9.81              #gravitational acceleration [m/s^2]
psi = 0.00055         #m'(t) [kg/s]
dpsi =T*psi/N         #mass gain step
x_c = 0.0025          #critical distant for drop fall
beta = 50             # [s/m]
rho = 1000            # density of water [kg/m^3]


#setting initialconditions
x = np.zeros(N); v = np.zeros(N); t = np.zeros(N)
x[0] = x0; v[0] = v0
m = np.zeros(N)
m[0] = m0


"""
the functions we have are
m(t)x''(t)+(b+psi)x'(t)+kx(t)=m(t)g
x''(t) = (m(t)g - (b+psi)x'(t) - kx(t))/m(t)
"""
#making functions

def diffEQ(xNow,vNow,tNow):
    aNow = (m[i]*g-(b+psi)*vNow-k*xNow)/m[i]
    return aNow


t_c = 0

#running the loop
for i in range(N-1):
    x1 = x[i]; v1 =v[i]
    oldmass = m[i]
    dm = beta*oldmass*v[i]
    A = (3*dm**4)
    B = (4*np.pi*rho*oldmass**3)
    dx =  (A/B)**(1./3)
    if x[i] > x_c:
```

```
            t_c_old = t_c
            t_c = t[i]
            #print (i)
            #print ('xi=',x[i])
            #print ('dx =',dx)
            t_diff = t_c - t_c_old
            #print ('tid mellom dråpe=', t_diff)
            m[i+1] = oldmass - dm + dpsi
            x1 = x[i] - dx

        else:
            m[i+1] = m[i] + dpsi

        x[i+1],v[i+1] = RK4(diffEQ,x1,v1,t,dt)
        t[i+1] = t[i] + dt


#plotting

#plotting the motion in x against time
plt.figure(figsize=(8,5))
plt.plot(t,x, '#803CA2', linewidth=2.0)
plt.title('Plot av utslag', fontsize=20)
plt.xlabel("Time [s]", fontsize=14)
plt.ylabel("Motion in x [m]", fontsize=14)
plt.tick_params(axis = 'both', which = 'major', labelsize =
    12)
#plt.savefig('Oppgave7utslag.png')
plt.show()

#plotting the phaseplot
plt.figure(figsize=(8,5))
plt.plot(x,v, '#803CA2', linewidth=2.0)
plt.title('Plot i faserommet', fontsize=20)
plt.xlabel("Motion in x [m]", fontsize=14)
plt.ylabel("Velocity [m/s]", fontsize=14)
plt.tick_params(axis = 'both', which = 'major', labelsize =
    12)
#plt.savefig('Oppgave7faseplott.png')
plt.show()
```

oppgave8.py

```
#imports
import numpy as np
import matplotlib.pyplot as plt
from RK4 import RK4
import seaborn
```

```
#declare constants
dt = 10.**-4          #some timestep [s]
m0 = 0.00001          #mass of waterdrop [kg]
k = 0.475             #stiffness constant [N/m]
x0 = 0.001            #start position [m]
v0 = 0.001            #start velocity [m/s]
T = 20.               #total time [s]
N = int(T/dt)         #number of things
b = 0.001             #[kg/s]
g = 9.81              #gravitational acceleration [m/s^2]
M = 100               #numer of steps
psi_list = np.linspace(0.00055, 0.00075, M)          #m'(t) [
    kg/s]
#dpsi =T*psi/N        #mass gain step
x_c = 0.0025          #critical distant for drop fall
beta = 50             # [s/m]
rho = 1000            # density of water [kg/m^3]


#setting initialconditions
x = np.zeros(N); v = np.zeros(N); t = np.zeros(N)
x[0] = x0; v[0] = v0
m = np.zeros(N)
m[0] = m0



"""
the functions we have are
m(t)x''(t)+(b+psi)x'(t)+kx(t)=m(t)g
x''(t) = (m(t)g - (b+psi)x'(t) - kx(t))/m(t)
"""
#making functions

def diffEQ(xNow,vNow,tNow):
    aNow = (m[i]*g-(b+psi)*vNow-k*xNow)/m[i]
    return aNow


#t_drop = np.zeros(M)

t_c =0

#running the loop
for psi in psi_list:
    print(psi)
    psi_plot_list = [psi]*50
    t_diff = []
```

```
    #dpsi =(T*psi[j])/N
    for i in range(N-1):
        x1 = x[i]; v1 =v[i]
        oldmass = m[i]
        dm = beta*oldmass*v[i]
        A = (3*dm**4)
        B = (4*np.pi*rho*oldmass**3)
        dx =  (A/B)**(1./3)
        if x[i] > x_c:
            t_c_old = t_c
            t_c = t[i]
            #print (i)
            #print ('xi=',x[i])
            #print ('dx =',dx)

            m[i+1] = oldmass - dm + psi*dt
            if m[i+1] <= 0:
                print ("mass cannot be 0 or negative")
                m[i+1]=0.00001

            x1 = x[i] - dx
            t_diff.append(t_c - t_c_old)

        else:
            m[i+1] = m[i] + psi*dt

        x[i+1],v[i+1] = RK4(diffEQ,x1,v1,t,dt)
        t[i+1] = t[i] + dt
    t_diff_plot = t_diff[-50:]
    plt.plot(psi_plot_list, t_diff_plot)
plt.show()



#plotting

"""
#plotting the motion in x against time
plt.figure(figsize=(8,5))
plt.plot(t,x, '#803CA2', linewidth=2.0)
plt.title('Plot av utslag', fontsize=20)
plt.xlabel("Time [s]", fontsize=14)
plt.ylabel("Motion in x [m]", fontsize=14)
plt.tick_params(axis = 'both', which = 'major', labelsize =
   12)
#plt.savefig('Oppgave8utslag_psi55.png')
plt.show()

#plotting the phaseplot
```

```
plt.figure(figsize=(8,5))
plt.plot(x,v, '#803CA2', linewidth=2.0)
plt.title('Plot i faserommet', fontsize=20)
plt.xlabel("Motion in x [m]", fontsize=14)
plt.ylabel("Velocity [m/s]", fontsize=14)
plt.tick_params(axis = 'both', which = 'major', labelsize =
    12)
#plt.savefig('Oppgave8faseplott_psi55.png')
plt.show()
"""
```