MEK3440 Oblig

Soran Hussein Mohmmed / soranhm

April 2018

Oppgave 1

I denne oppgaven skal vi studere FitcHugh-Nagumo modelen som er gitt som

$$\begin{aligned}\dot{v} &= v - \frac{v^3}{3} - w + I \\ \dot{w} &= v + a - bw\end{aligned} \tag{1}$$

$$0 < b < 1 \qquad 1 - \frac{2}{3}b < a < 1 \tag{2}$$

Der v er membranspenning og w er "gjenopprettings" variabelen, hvilke modeller aktiverer en utad ionstrøm. Mens I modeller en injisert elektrisk strøm.

a)

Antar at $0 < b < 1$, og $1 - \frac{2}{3}b < a < 1$. Skal vise at hvis I = 0, så har systemet (1) kun et likevekts punkt (v_0^*, w_0^*), og at denne er sink.

$$u(t) = \begin{pmatrix} v(t) \\ w(t) \end{pmatrix}, f(u) = \begin{pmatrix} v - \frac{v^3}{3} - w \\ v + a - bw \end{pmatrix}$$

$$\begin{pmatrix} v - \frac{v^3}{3} - w \\ v + a - bw \end{pmatrix} \Leftrightarrow \dot{u} = f(u)$$

$$f_1(u^*) = 0 : v - \frac{v^3}{3} - w = 0 \rightarrow w = v - \frac{v^3}{3}$$

$$f_2(u^*) = 0 : v + a - bw = 0 \rightarrow w = \frac{v + a}{b}$$

Setter disse lik hverandre og bruker hintet at $P(v_0^*) = a$.

$$v - \frac{v^3}{3} - w = \frac{v + a}{b} \rightarrow bv - \frac{b}{3}v^3 = v + a \rightarrow a = (b-1)v - \frac{b}{3}v^3$$

$$a = P(v_0^*) = (b-1)v - \frac{b}{3}v^3$$

Ser at P'(v_0^*) alltid er negativt som sier at det aldri vil øke igjen , så dermed har vi kun et likevektspunkt. Har også plottet dette for b mellom 0 og 1 , og ser at det kun er et skjæringspunkt. For at dette skal være sink så må trace(Df(u^*)) <0.

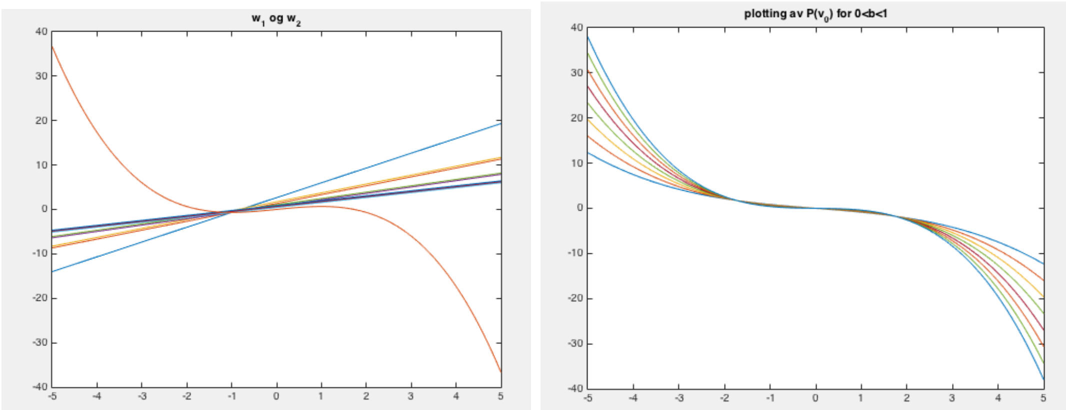


Figure 1: Plot av f_1 og f_2. Plotting P(v_0^*) med $0 < b < 1$

$$Df(u^*) = \begin{pmatrix} 1 - v_0^* & -1 \\ 1 & -b \end{pmatrix} \rightarrow trace(Df(u^*)) = (1 - v_0^{*2}) - b$$

setter $P(v_0^*) > 0$ får $(b-1)v_0^* > \frac{b}{3}v_0^{*3}$, bruker nå dette til å finne ut om v_0^* skal være større eller mindre enn null.

$$v_0^* > 0 :\rightarrow (b-1)v_0^* > \frac{b}{3}v_0^{*2}$$
$$v_0^* < 0 :\rightarrow (b-1)v_0^* < \frac{b}{3}v_0^{*2}$$

Ser at $v_0^* > 0$ ikke stemmer , så v_0^* må være mindre enn 0. Bruker nå trace til å finne hva slags verdi v_0^* skal ha. $(1 - v_0^{*2}) - b < 0 \rightarrow 1 - v_0^{*2} < b$ for at dette skal stemme og v_0^* skal være mindre enn null så får vi sink for $v_0^* < -1$.

b)

Nå har vi at $I \in R$, bruker (1) og (2) til å finne at det kun finnes et likevektspunkt.

$$\dot{v} = v - \frac{v^3}{3} - w + I$$
$$\dot{w} = v + a - bw$$
$$v - \frac{v^3}{3} - w + I = 0 \quad \rightarrow \quad w = v - \frac{v^3}{3} + I$$
$$v + a - bw = 0 \quad \rightarrow \quad w = \frac{v+a}{b}$$
$$\frac{v+a}{b} = v - \frac{v^3}{3} + I \rightarrow v + a = bv - \frac{b}{3}v^3 + bI \rightarrow a = (b-1)v - \frac{b}{3}v^3 + bI$$
$$a = P(v_I^*) = (b-1)v_I^* - \frac{b}{3}v_I^{*3} + bI$$

ser at P'(v_I^*) kun består av negative ledd som i a) , dermed har vi kun et likevekts punkt. Ser at begge leddene med v_i^* er negative og opphøyd i oddetall som gjør at de holder seg negative hvis v_i^* er postiv og postiv hvis v_i^* er negativ. Mens I leddet er postivit, dette gjør at hvis I øker så må v_i^* også øke, dermed har vi at v_i^* øker monotont med I.

c)

Bruker nå b) til å finne ut hva I må være for at vi skal få en kilde. For å få en kilde så må vi få trace($Df(u^*)$) > 0, dette gir da $1 - v_I^{*2} - b > 0 \rightarrow v_I^{*2} < 1 - b$. Enda en ting som trenges for å få kilde er Det($Df(u^*)$) > 0, og dette gir $b(v_I^{*2} - 1) + 1 > 0 \rightarrow v_I^{*2} > \frac{b-1}{b}$. Ved hjelp av disse to ser vi at $\frac{b-1}{b} < v_I^{*2} < 1 - b$. Bruker nå dette i P($v_I^*$) for å finne hva I må være for at vi skal få kilde. Bruker a > P(v_I^*).

$$0 < v_I^{*2} < 1 - b : a > (b-1)v_I^* - \frac{b}{3}v_I^*(1-b) + bI$$
$$v_I^* > 0 : a > (b-1)\sqrt{1-b} - \frac{b}{3}\sqrt{1-b}(1-b) + bI$$
$$bI < a - (b-1)\sqrt{1-b} + \frac{b}{3}\sqrt{1-b}(1-b)$$
$$I < \frac{a}{b} - \frac{1}{b}(\frac{b}{3} - 1)(b-1)\sqrt{1-b}$$
$$I < \frac{a}{b} - \frac{1}{b}(\frac{b}{3} - 1)(b-1)^{\frac{3}{2}}$$
$$v_I^* < 0 : I > \frac{a}{b} + \frac{1}{b}(1 - \frac{b}{3})(1-b)^{\frac{3}{2}}$$
$$\frac{b-1}{b} < v_I^{*2} < 0 : I < \frac{1}{b}(a + \frac{2}{3}(1-b)v_I^* = \frac{a}{b} + (\frac{2}{3b} - \frac{2}{3})v_I^*$$

$\frac{b-1}{b} < v_I^{*2} < 0$ gir et kompleks svar, så vi kan bare se bort fra den. Dermed får vi:

$$I \in \Big(\frac{a}{b} - \frac{1}{b}(\frac{b}{3} - 1)(b-1)^{\frac{3}{2}}, \frac{a}{b} + \frac{1}{b}(1 - \frac{b}{3})(1-b)^{\frac{3}{2}}\Big)$$

d)

Løser (1) numerisk med hjelp av Python, bruker a = 0.7 og b = 0.8, tester med flere T og N slik at det blir så bra plot som mulig. Velger en $I = \frac{a}{b}$ og en I = 0.2 for å lage sink og kilde. Velger N = 1000, T = 100 og h = $\frac{T}{N}$, slik at det blir liten nok i forhold til T og N. Har lagt ved programmet i slutten av Obligen, mens plottene er lagt til under her.

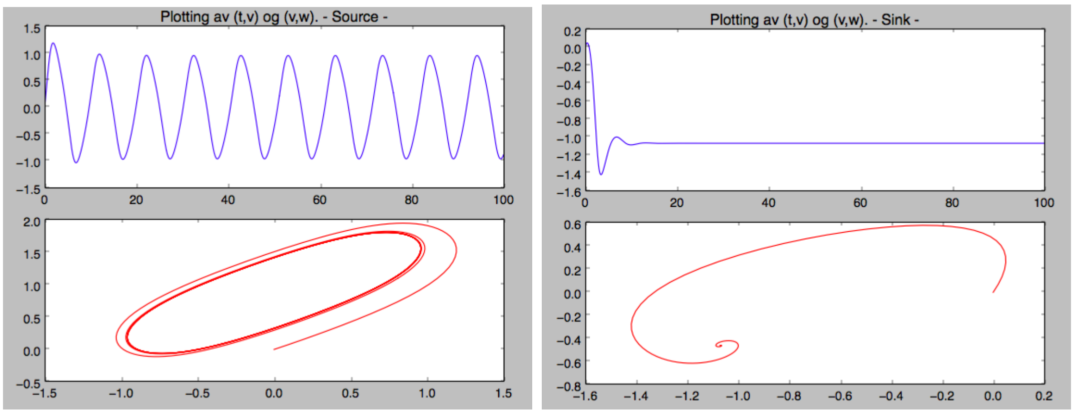


Figure 2: kilde og sink

e)

Antar nå at a = 0, I = 0 og 1 < b < 3, og skal vise at systemet (1) har 3 likevekts punkter , der en av de er sadelpunkt og de andre er sink. For å vise at det er sadelpunkt så må egenverdiene være $\lambda_- < 0 < \lambda_+$ og for å vise at det er sink så må egenverdiene være $\lambda_- < \lambda_+ < 0$:

$$\cancel{a} = P(v_I^*) = (b-1)v_I^* - \frac{b}{3}v_I^{*3} + \cancel{bI}$$

$$(b-1)v_I^* - \frac{b}{3}v_I^{*3} = 0$$

$$v_I^*\left((b-1) - \frac{b}{3}v_I^{*2}\right) = 0$$

$$(b-1) - \frac{b}{3}v_I^{*2} = 0$$

$$v_I^* =^{+}_{-} \sqrt{\frac{3(b-1)}{b}}, \qquad v_I^* = 0$$

Bruker nå de 3 likevektspunktene til å se om de er sadelpunkt eller sink:

$\underline{v_I^* = 0:}$ $\quad det \begin{pmatrix} 1-\lambda & -1 \\ 1 & -b-\lambda \end{pmatrix} = (1-\lambda)(-b-\lambda)+1 = -b-\lambda+\lambda b+\lambda^2+1$

$$\lambda^2+(b-1)\lambda+(1-b)=0 \rightarrow \quad \lambda = \frac{-(b-1)\pm\sqrt{(b-1)^2-4(1-b)}}{2}$$

Dette gir et postivit egenverdi og et negativt svar, dermed har vi sadelpunkt i $v_I^* = 0$. Pga når vi opphøyer egenverdien i anden når vi regner ut determinaten, så vil de to restrerende egenverdiene gi samma. Velger $\frac{3(b-1)}{b} = R$ for å gjøre det litt enkelt for meg.

$$det \begin{pmatrix} (1-R)-\lambda & -1 \\ 1 & -b-\lambda \end{pmatrix} = \Big((1-R)-\lambda\Big)(-b-\lambda)+1$$

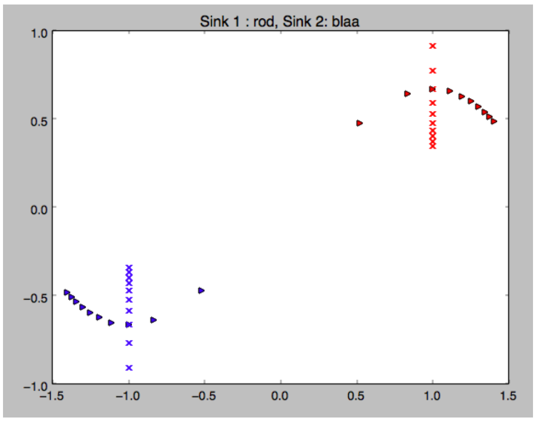
$$-b(1-R)-\lambda-\lambda(1-R)+\lambda b+\lambda^2 = \lambda^2+(b-(1-R))\lambda+(1-b(1-R))=0$$

$$\lambda = \frac{-(b-(1-R)\pm\sqrt{(b-(1-R))^2-4(1-b(1-R))}}{2}$$

Trenger kun den reele delene av egenveriden skal være mindre enn null for sink, dermed kan jeg kun se på $-b+1-R = -b+1-3(1-\frac{1}{b}) = -b-2+\frac{3}{b}$ bruker at b skal være mellom 1 og 3, og ser at $\frac{1}{b} < 1 \rightarrow -(b-(1-R)) < 0$. Dermed har vi sink for $v_I^* = \pm\sqrt{\frac{3(b-1)}{b}}$

f)

Her plotter jeg mange forskjellige startverdier (v_0, w_0) og velger ut flere b verdier mellom 1 og 3. Dermed ser jeg på om ende punktene er nærmest: punkt 1: $-\sqrt{\frac{3(b-1)}{b}}$ og punkt 2: $\sqrt{\frac{3(b-1)}{b}}$, der jeg plotter punkt 1 i blå x, og punkt 2 i rød x sammen med enepunktene til grafene.



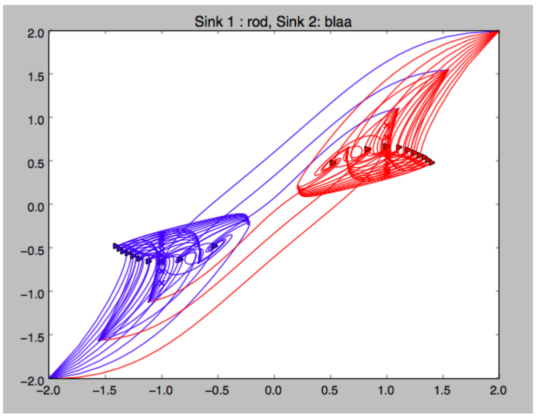


Figure 3: ploting av endepunkter til grafene og grafene

Oppgave 2

Skal studere en ikke-lineær pendullum, der pendullen er vektløs og henger fra en vegg som gjør at den kan rotere, og i enden er det en objekt med masse = 1. $\theta(t)$ er vinkelen når du ser ned på objektet. Dette gir ODE:

$$\ddot{\theta} = -sin(\theta) \tag{3}$$

der $\theta \in R$, og for eksempel når $\theta > 2\pi$ så har den gått en full runde rundt.

a)

etter p = $\dot{\theta}$ og reduserer systemet til en firste orden ODE:

$$\dot{\theta} = p$$
$$\dot{p} = \ddot{\theta} = -sin(\theta)$$

Det er en Hamiltonian system hvis det kan skrives på formen:

$$\dot{\theta} = \frac{\partial H}{\partial p}(p, \theta)$$
$$\dot{p} = -\frac{\partial H}{\partial \theta}(p, \theta)$$

Finner nå H(θ,p):

$$\dot{\theta} = \frac{\partial H}{\partial p}(p, \theta) = p \rightarrow H_1(p, \theta) = \frac{1}{2}p^2 + f_1(\theta)$$
$$\dot{p} = -\frac{\partial H}{\partial \theta}(p, \theta) = sin(\theta) \rightarrow H_2(p, \theta) = -cos(\theta) + f_2(p)$$
$$H(p, \theta) = \underline{\frac{1}{2}p^2 - cos(\theta)}$$

b)

likevekts punktene for systemet er:

$$\dot{\theta} = \frac{\partial H}{\partial p}(p, \theta) = p = 0 \rightarrow p = 0$$
$$\dot{p} = -\frac{\partial H}{\partial \theta}(p, \theta) = sin(\theta) = 0 \rightarrow sin(\theta) = 0$$

$\sin(\theta) = 0$ når $\theta = \pi \overset{+}{-} k\pi$ for alle k = 0,1,2,....Bruker nå disse til å finne egenverdiene og se om det er sadel/kilde eller sluk:

$$\begin{pmatrix} 0 & 1 \\ -cos(\theta) & 0 \end{pmatrix}$$

$$\underline{(0,0)} : \begin{pmatrix} -\lambda & 1 \\ -1 & -\lambda \end{pmatrix} = \lambda^2 + 1 = 0$$

$$\underline{(0,\pi)} : \begin{pmatrix} -\lambda & 1 \\ 1 & -\lambda \end{pmatrix} = \lambda^2 - 1 = 0$$

for (0,π) $\rightarrow \lambda = \overset{+}{-} 1 \rightarrow$ sadelpunkt.
for (0,0) $\rightarrow$ to postive ustabile node (kilde). Vi får sadel når n er partall og kilde når n er odde tall i $\theta = n\pi$.

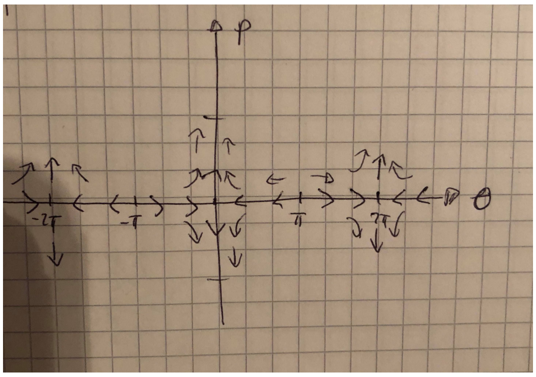


Figure 4:

c)

Lyapunov function: L(x,y) = H(x,y) - H(x^*, y^*), i vår tilfelle får vi L(p,θ) = H(p,θ) - H(p^*, θ^*). Vi har både oddetall (0,Dπ) og partall (0,Pπ).
Oddetall: H(p^*, θ^*) $= \frac{1}{2} * 0^2 - cos(D * \pi) = 1 \rightarrow L(p,\theta) = \frac{1}{2}p^2 - cos(\theta) + 1$, setter $\theta = D\pi : L(p, D\theta) = \frac{1}{2}p^2 + 1 - 1 = \frac{1}{2}p^2 > 0$. Setter nå p = 0 og $\theta = \pi + \epsilon : L(0, \pi + \epsilon) = 0 - cos(\pi + \epsilon) - 1 =< 0$, Dette sier oss ingenting om punktene.
Partall : $H(p^*, \theta^*) = \frac{1}{2} * 0^2 - cos(P * \pi) = -1 \rightarrow L(p,\theta) = \frac{1}{2}p^2 - cos(\theta) + 1 \rightarrow L(p, P\theta) = \frac{1}{2}p^2 > 0$. Definerer området p $\in$ (-a,a), $a \in R$ og $\theta \in (n\pi - b, n + b), b < \frac{\pi}{2}$, ser nå at L(p,$\theta$) ≤ 0 i nabolaget. Dermed er $(p^*_{2\pi}, \theta^*_{2\pi})$ Stabil.

d)

Bruker python til å skrive en programm som kjører Forward Euler på problemet vårt. medn initial verdier $(\theta_0, p_0) = (0, p_0)$ med $p_0 = \frac{k}{2}, k = 0, ..., 8$. Velger T = 10, N = 100, plotter også den numeriske energyen $E_n := H(p_n, \theta_n)$ mot tid.

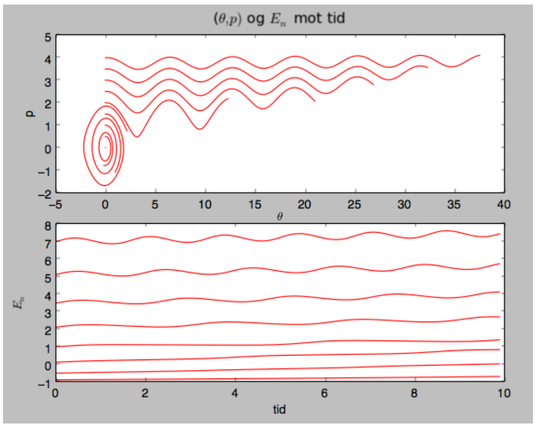


Figure 5: Ploting av (θ,p) over og E_n under

e)

plotter nå det forrige mot den nye:

$$\theta_{n+1} = \theta_n + h\frac{\partial H}{\partial p}(p_{n+1}, \theta_n)$$
$$p_{n+1} = p_n - h\frac{\partial H}{\partial \theta}(p_{n+1}, \theta_n)$$

plotter dette mot den gamle og ser at det blir en liten forsivning i grafen, når jeg samenligner med energien ser jeg at den nye er bedre pga tilnærmingen er mye næremere enn det gamle, og stemmer bedre med den teoritiske delen. Den gamle (θ,p) og E_n er i rød mens de nye er i blå

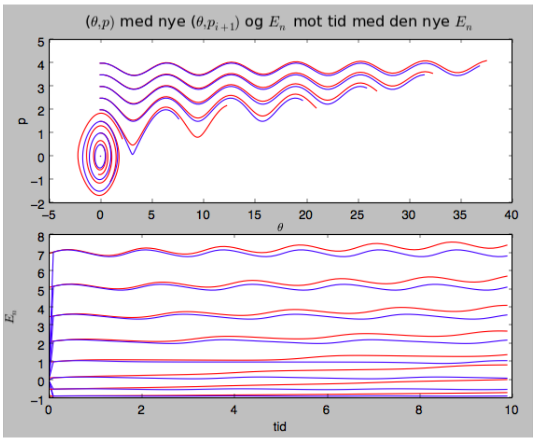


Figure 6: Ploting av (θ,p) over og E_n under

```
# -*- coding: utf-8 -*-
# OPPGAVE ! D og F
import matplotlib
import matplotlib.pyplot as plt # For plotting
import numpy as np # For arrays
import scipy.linalg as la # For linear algebra

def f1(v,w,a,b,I):
    return v - (v**3)/3. - w + I

def f2(v,w,a,b,I):
    return v + a - b*w

def simulate_fwd_euler(T,N,h,a,b,I,v0,w0):
    """
    INPUT:
        T (float): End time.
        N (integer): Number of time steps.
        x0 (np.array): NumPy array of size 2, initial condition.
        A (np.array): NumPy array of size 2x2, system matrix.
    OUTPUT:
        y (np.array): NumPy array of size Nx2, numerical solution at each time step.
    """
    # Initialize solution array
    y = np.zeros((N,1), dtype=float)
    z = np.zeros((N,1), dtype=float)
    y[0] = v0
    z[0] = w0
    # Do each time step:
    for i in range(N-1):
        # Execute fwd_euler from earlier
        y[i+1] = y[i] + h * f1(y[i],z[i],a,b,I) # Return solution array
        z[i+1] = z[i] + h * f2(y[i],z[i],a,b,I) # Return solution array
    return y,z

a = 0.7; b = 0.8
I = a/b
I2 = 0.2
N = 1000; T = 100. ; h = T/N;
t = np.linspace(0, T, N)

# D
v0 = 0; w0 = 0;
v2,w2 = simulate_fwd_euler(T,N,h,a,b,I,v0,w0)
v3,w3 = simulate_fwd_euler(T,N,h,a,b,I2,v0,w0)
print 'Sink: I = ', I2,' (a/b), Source: I = ',I
plt.figure(1)
plt.subplot(211)
plt.plot(t,v2,'b')
plt.title('Plotting av (t,v) og (v,w). - Source -')
plt.subplot(212)
plt.plot(v2,w2,'r')

plt.figure(2)
```

```
plt.subplot(211)
plt.plot(t,v3,'b')
plt.title('Plotting av (t,v) og (v,w). - Sink -')
plt.subplot(212)
plt.plot(v3,w3,'r')

# F
plt.figure(3)
v_0 = np.linspace(-2,2,10)
w_0 = np.linspace(-2,2,10)
b2 = [1.1,1.6,2.1,2.6,2.9]
I3 = 0; a2 = 0; b2 =1.8
for i,j in zip(v_0, w_0):
    for b3 in np.linspace(1.1,2.9,10):
        punkt2 = ((3*(b-1))/b)**(1/2);
        wrt = punkt2/b3;
        color = ''; leg = ''
        v4,w4 = simulate_fwd_euler(T,N,h,a2,b3,I3,i,j)
        if abs(punkt2-v4[-1])<abs(-punkt2-v4[-1]):
            color = 'r' # punkt 1
        else:
            color = 'b' # punkt 2
        plt.plot(v4,w4,color,label='v0 = %s, w0 = %s' % (i,j))
        plt.plot(v4[-1],w4[-1],'>',color = color)
        plt.plot(punkt2,wrt,'rx',-punkt2,-wrt,'bx')
        #plt.plot(v4[0],w4[0],'s')
plt.title('Sink 1 : rod, Sink 2: blaa')
#plt.legend(loc=4)
plt.show()


# -*- coding: utf-8 -*-
# OPPGAVE D OG E
import matplotlib
import matplotlib.pyplot as plt # For plotting
import numpy as np # For arrays
import scipy.linalg as la # For linear algebra

def simulate_fwd_euler(T,N,t0,p0):
    """
    INPUT:
        T (float): End time.
        N (integer): Number of time steps.
        x0 (np.array): NumPy array of size 2, initial condition.
        A (np.array): NumPy array of size 2x2, system matrix.
    OUTPUT:
        y (np.array): NumPy array of size Nx2, numerical solution at each time step.
    """
    # Initialize solution array
    t= np.zeros((N,1), dtype=float)
    p = np.zeros((N,1), dtype=float)
    t2 = np.zeros((N,1), dtype=float)
    p2 = np.zeros((N,1), dtype=float)
```

```python
    H = np.zeros((N,1), dtype=float)
    H2 = np.zeros((N,1), dtype=float)
    tid = np.zeros((N,1), dtype=float)

    t[0] = t0; p[0] = p0
    t2[0] = t0; p2[0] = p0
    h = T/N
    H[0] = 0.5 * p0**2 - np.cos(t0)
    # Do each time step:
    for i in range(N-1):
        t[i+1] = t[i] + h * p[i]          # Return solution array (do/dt)
        p[i+1] = p[i] + h * (- np.sin(t[i])) # Return solution array (dp/dt)
        p2[i+1] = p2[i] - h * (np.sin(t2[i])) # Return solution array (dp/dt)
        t2[i+1] = t2[i] + h * p2[i+1]          # Return solution array (do/dt)
        H[i+1] = 0.5 * p[i+1]**2 - np.cos(t[i+1])
        H2[i+1] = 0.5 * p2[i+1]**2 - np.cos(t2[i+1])
        tid[i+1] = tid[i] + h
    return t,p,H,H2,tid,t2,p2

t0 = 0; N = 100; T = 10.

# D
plt.figure(1)
plt.subplot(211)
for k in range(0,9):
    p0 = k/2.
    t,p,H,H2,tid,t2,p2 = simulate_fwd_euler(T,N,t0,p0)
    plt.plot(t,p,'r')
plt.title(r'$(\theta,p)$ og $E_n$ mot tid ')
plt.xlabel(r'$\theta$')
plt.ylabel(('p'))
plt.subplot(212)
for k in range(0,9):
    p0 = k/2.
    t,p,H,H2,tid,t2,p2 = simulate_fwd_euler(T,N,t0,p0)
    plt.plot(tid,H,'r')
plt.ylabel(r'$E_n$')
plt.xlabel(('tid'))

#E
plt.figure(2)
plt.subplot(211)
for k in range(0,9):
    p0 = k/2.
    t,p,H,H2,tid,t2,p2 = simulate_fwd_euler(T,N,t0,p0)
    plt.plot(t,p,'r',t2,p2,'b')
plt.title(r'$(\theta,p)$ med nye $(\theta,p_{i+1})$ og $E_n$ mot tid med den nye $E_n$ ')
plt.xlabel(r'$\theta$')
plt.ylabel(('p'))
plt.subplot(212)
for k in range(0,9):
    p0 = k/2.
    t,p,H,H2,tid,t2,p2 = simulate_fwd_euler(T,N,t0,p0)
    plt.plot(tid,H,'r',tid,H2,'b')
```

```
plt.ylabel(r'$E_n$')
plt.xlabel(('tid'))
plt.show()
```