Oppg1

1a: Gi en algoritme for å beregne en tilnærming til objektets aksellerasjon a(t) = v 0 (t) ut fra de beregnede verdiene (ti , vi) av farten.

Siden vi har hastighet gitt av tid. Må vi derivere for å finne akselerasjonen. Dette gjøres ved å bruke numerisk derivasjon. Siden vi har data fra punkt data og ikke en kontinuerlig funksjon, bruker vi en steglende gidd ved t_i – t_i+1. Da er algoritmen:

```
def aksellerasjon (x,y):
  dy = [(y[i+1] - y[i])/ abs(x[i+1] - x[i]) for i in range(len(y)-1)]
  return(dy)
```

Som da returnerer den deriverte av y, som i vår oppgave er v.

Eksempel:

```
t = list(range(0,10,2))
v = [2,4,8,12,24]
print(t[1:],aksellerasjon(t,v))
```

```
python .\1a.py
[2, 4, 6, 8] [1.0, 2.0, 2.0, 6.0]
```

1b: Gi en algoritme for å beregne en tilnærming til objektets avstand s(t) fra startpunktet ut fra de beregnede verdiene når v(t) = s 0 (t) og s(t0) = 0.

For å finne avstanden fra startpunktet bruker jeg numerisk integrasjon. Som i 1a må jeg bruke integrasjonen over punkter og ikke en funksjon. Det gjør jeg med å regne ut dx for hver element i y. Det gir meg algoritmen:

```
def avstand(x,y):
  Y = [0]
  for i in range(1,len(y)):
    dx = x[i]-x[i-1]
    Y.append(y[i]*dx + Y[-1])
  return Y
```

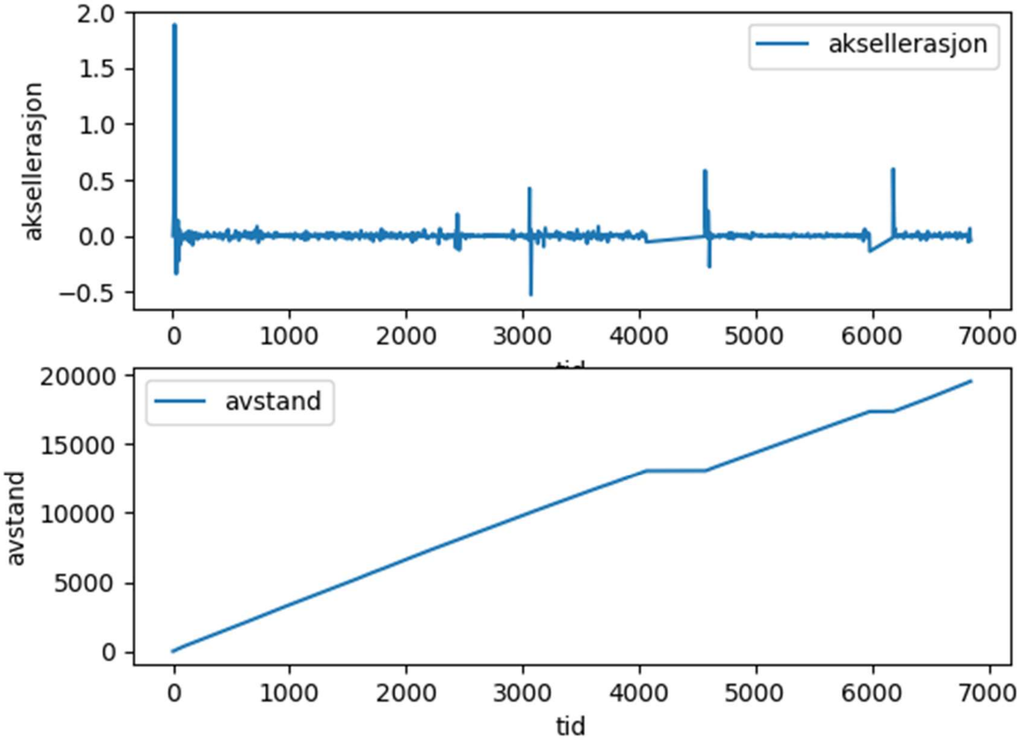
Eksempel:

```
t = list(range(0,10,2))
v = [2,4,8,12,24]
print(t, avstand(t,v))
```

```
python .\1b.py
[0, 2, 4, 6, 8] [0, 8, 24, 48, 96]
```

1c: Last ned fila running.txt og kjør denne koden, og bruk algoritmene fra a) og b) til å lage to plott: Ett der du plotter objektets akselerasjon mot tid, og ett der du plotter objektets avstand fra startpunktet mot tid.

```
from Oppg1a import *
from Oppg1b import *

import numpy as np
import matplotlib.pyplot as plt
t = []
v = []
infile = open('running.txt','r')
for line in infile:
   tnext, vnext = line.strip().split(',')
   t.append(float(tnext))
   v.append(float(vnext))
infile.close()

V = np.asarray(avstand(t, v))
dv = np.asarray(aksellerasjon(t, v))

if __name__ == "__main__":
   plt.subplot(211)
   plt.plot(t[1:],dv,label='aksellerasjon')
   plt.legend()

   plt.subplot(212)
   plt.plot(t, V, label='avstand')
   plt.legend()

   plt.show()
```


aksellerasjon
2.0
1.5
1.0
0.5
0.0
−0.5
0
1000
2000
3000
4000
5000
6000
7000
tid
avstand
20000
15000
10000
5000
0

Oppg2

2a:

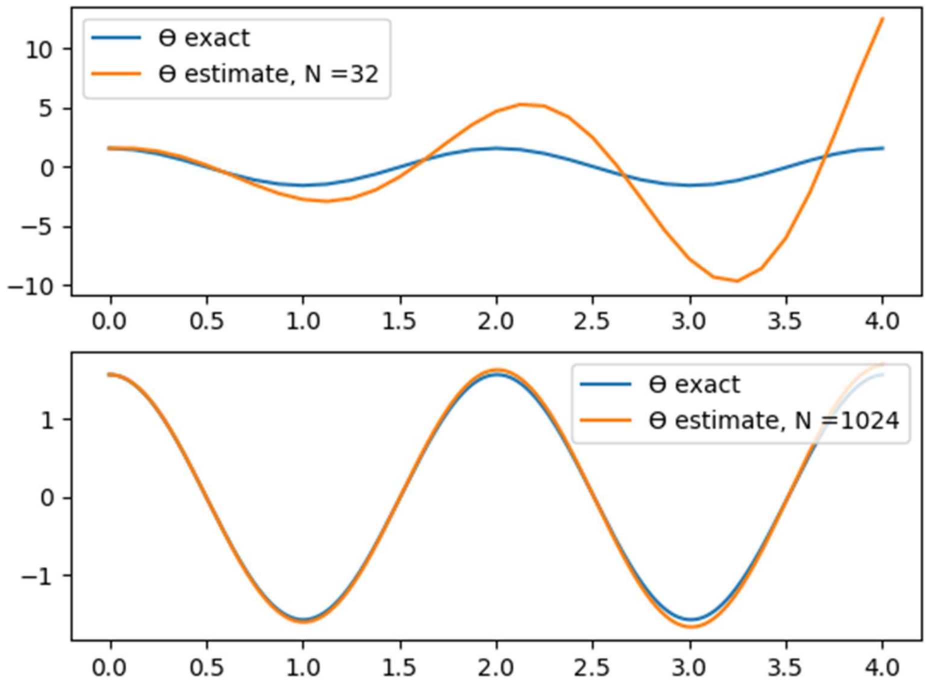
2b:

2c: Skriv en Python-funksjon lin_pendel_euler som med funksjonskallet v, theta = lin_pendel_euler(v0, theta0, g, L, N, h).

```
import numpy as np
import matplotlib.pyplot as plt

def lin_pendel_euler(v0, theta0, g, L, N, h=1e-14):
  T = h*N #siden vi har h = T/N
  v, theta = [0]*(N+1), [0]*(N+1)
  v[0], theta[0] = v0, theta0

  for k in range(N):
    v[k+1] = v[k] - g*h*theta[k]
    theta[k+1] = theta[k] + h*(v[k]/L)
  return v, theta
```

2d: Plott den numeriske løsningen for vinkelutslaget som en funksjon av tid sammen med den eksakte løsningen du fant i b). Kommenter forskjellene.

```
from Oppg2c import *

import numpy as np
import matplotlib.pyplot as plt

def lin_pendel(t, v0, theta0, g, L):
   return theta0 * np.cos(np.sqrt(g/L)*t) + v0/np.sqrt(g*L) * np.sin(np.sqrt(g/L)*t) #løsning fra 2b

g, L, v0, theta0 = 9.81, 1, 0, np.pi/2
N = [2**5, 2**10]
T = 4
h = [T/n for n in N]
t = [np.linspace(0,T,n+1) for n in N]


plt.subplot(2, 1, 1)
v_hat, theta_hat = lin_pendel_euler(v0, theta0, g, L, N[0], h[0])
theta = lin_pendel(t[0], v0, theta0, g, L)
plt.plot(t[0], theta, label='Θ exact')
plt.plot(t[0], theta_hat, label='Θ estimate, N ={}'.format(N[0]))
plt.legend()

plt.subplot(2, 1, 2)
v_hat, theta_hat = lin_pendel_euler(v0, theta0, g, L, N[1], h[1])
theta = lin_pendel(t[1], v0, theta0, g, L)
plt.plot(t[1], theta, label='Θ exact')
plt.plot(t[1], theta_hat, label='Θ estimate, N ={}'.format(N[1]))
plt.legend()

plt.show()
```



Ser at estimatet til theta har en økende svingning. Noe som ikke er synlig i den eksakte theta. Da en pendel ikke kan svinge bredere enn i starten. Dermed er ikke theta et perfekt estimat.

2e: beregne feilen i Eulers metode

```
from Oppg2c import *
from Oppg2d import *

import numpy as np
import matplotlib.pyplot as plt



class epsilon:
    def __init__(self,f, f_hat,v0, theta0, g, L, T):
        self.f = f
        self.f_hat = f_hat
        self.v0, self.theta0, self.L= v0, theta0, L
        self.T = T

    def __call__(self, h, N):
            #feilestiamtet
        v0, theta0, L = self.v0, self.theta0, self.L
        T = self.T
        t = np.linspace(0,T,int(N+1))
        f = self.f(t, v0, theta0, g, L)
        f_hat = self.f_hat(v0, theta0, g, L, N, h)[1]
        return abs(f[-1]-f_hat[-1])
if __name__ == "__main__":
    N = [2**i for i in range(4,11)]
    T = 4
    h = [T/n for n in N]
    g, L, v0, theta0 = 9.81, 1, 0, np.pi/2

    print('''-------------------------
|   N       h     e(p) |''')
    for i in range(len(N)):
        e = epsilon(lin_pendel,lin_pendel_euler,v0, theta0, g, L, T)
        print('|{:.2e} {:.4f} {:7.4f}|'.format(N[i], h[i], e(h[i], N[i]) ))
```

For å beregne feilestiamtet satte jeg opp en klasse med en init for å ta inn konstantene. Så en call metode for å kalle funksjonen. Så avslutter med en print slik at vi får error estimatet i en tabell.

```
run Oppg2e.py
-------------------------
|   N          h        e(p) |
|1.60e+01 0.2500 27.3939|
|3.20e+01 0.1250 10.9065|
|6.40e+01 0.0625  3.5644|
|1.28e+02 0.0312  1.3131|
|2.56e+02 0.0156  0.5614|
|5.12e+02 0.0078  0.2598|
|1.02e+03 0.0039  0.1250|
```

2f: Bruk tallene du fant i oppgave e) til å estimere konvergensraten. . Hva tror du konvergensraten til Eulers metode er?

```
from Oppg2c import *
from Oppg2d import *
from Oppg2e import *

import numpy as np
import matplotlib.pyplot as plt


class konvergens(epsilon):
    def p(self, h1, N):
        e = self.__call__
        h2 = h1/2
        return np.log(e(h1, N)/e(h2, N*2))/np.log(h1/h2)


if __name__ == "__main__":
        N = [2**i for i in range(4,11)]
        T = 4
        h = [T/n for n in N]

        print('''-----------------------------------
|    N       h     e(p)     p   |''')
        for i in range(len(N)):
            e = konvergens(lin_pendel,lin_pendel_euler,v0, theta0, g, L, T)
            p = e.p(h[i], N[i])
            print('| {:.2e}{:8.4f}{:8.4f}{:8.4f}|'.format(N[i], h[i], e(h[i], N[i]), p))
```

Definerer en ny classe som arver fra epsilon. Definerer en ny metode p som kalkulerer p med samme init som epsilon. Så gjør et tilsvarende plott som i 2e, med p i tillegg.

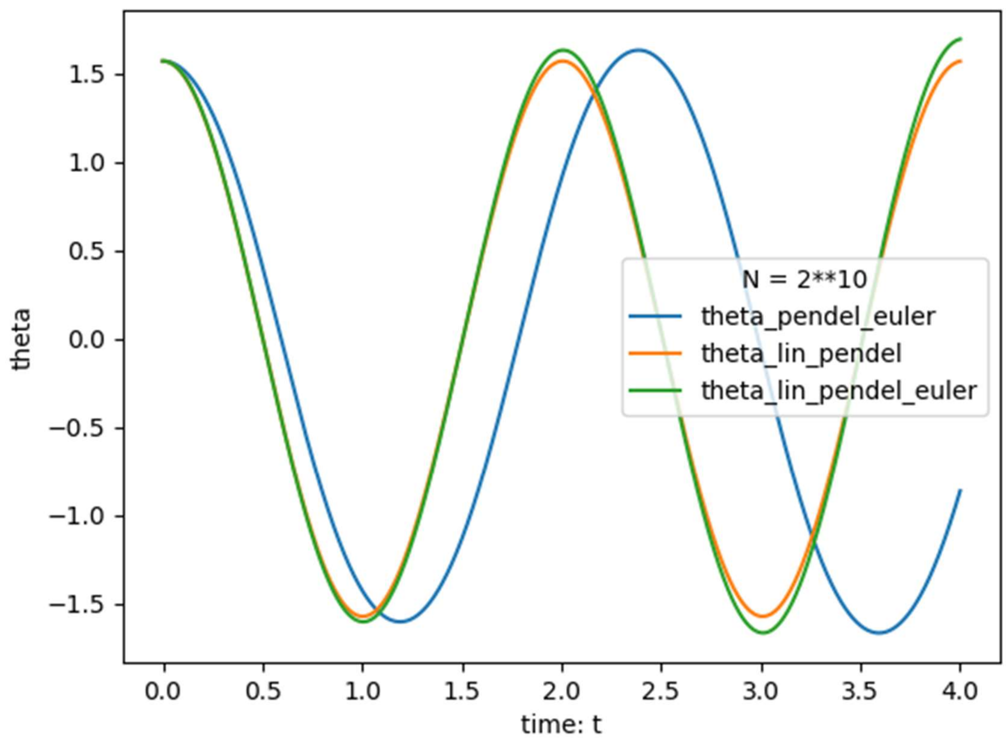
Vi får da ut tabellen:

```
run Oppg2f
-----------------------------------
|    N       h     e(p)     p   |
| 1.60e+01  0.2500 27.3939  1.3287|
| 3.20e+01  0.1250 10.9065  1.6135|
| 6.40e+01  0.0625  3.5644  1.4406|
| 1.28e+02  0.0312  1.3131  1.2259|
| 2.56e+02  0.0156  0.5614  1.1119|
| 5.12e+02  0.0078  0.2598  1.0555|
| 1.02e+03  0.0039  0.1250  1.0276|
```

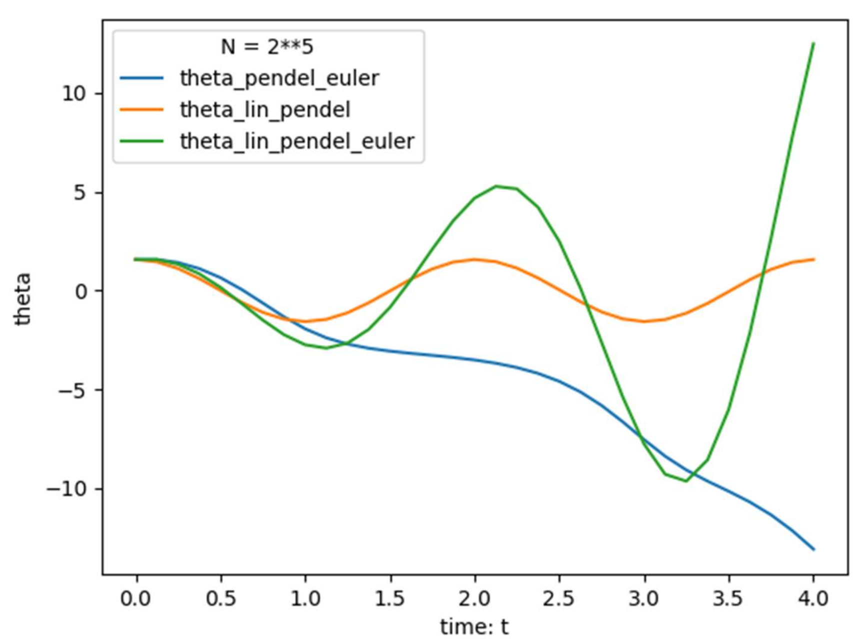
Ser ut til at p går mot 1.

2g: Skriv en Python-funksjon pendel_euler. Plott θ som en funksjon av tid. Inkluder de to løsningene fra b) og c) i plottet. Kommenter forskjellene.

```
from Oppg2f import *

import matplotlib.pyplot as plt
import numpy as np

def pendel_euler(v0, theta0, g, L, N, h):
    v = np.zeros(N+1)
    theta = np.zeros(N+1)
    v[0], theta[0] = v0, theta0
    for k in range(N):
        v[k+1] = v[k] - g*np.sin(theta[k])*h
        theta[k+1] = theta[k] + h* (1/L * v[k])
    return v, theta


g, L, v0, theta0 = 9.81, 1, 0, np.pi/2
T = 4
N = 2**10
h = T/N
t = np.linspace(0,T,N+1)

v_pendel_euler, theta_pendel_euler = pendel_euler(v0, theta0, g, L, N, h)
theta_lin_pendel = lin_pendel(t, v0, theta0, g, L)
v_lin_pendel_euler, theta_lin_pendel_euler = lin_pendel_euler(v0, theta0, g, L, N, h)

plt.plot(t,theta_pendel_euler, label='theta_pendel_euler')
plt.plot(t,theta_lin_pendel, label='theta_lin_pendel')
plt.plot(t,theta_lin_pendel_euler, label='theta_lin_pendel_euler')
plt.xlabel('time: t')
plt.ylabel('theta')
plt.legend(title ='N = 2**10')
plt.show()
```



VI ser at theta_pendel_euler raskt får en større feil en det theta_lin_pendel har. Hvis vi ser på en mindre N så får vi.



Der vi ser at theta_lin_pendel_euler raskt blir feil men føler fortsatt theta_lin pendel. Mens theta_pendel_euler er mindre feil men går i feil retning. Dermed mener jeg at theta_lin_pendel_euløer er det beste estimatet.

2 $$\frac{d^2\theta}{dt^2}(t) = -\frac{g}{L}\sin(\theta(t))$$

Vi har at $v(t) = L\frac{d\theta}{dt}(t)$

Vil finne et system av ligningen

$$\frac{d^2\theta}{dt}t = -\frac{g}{L}\sin(\theta(t)) \quad / \cdot L$$

$$L\frac{d^2\theta}{dt^2}(t) = -g\sin(\theta(t))$$

$L\frac{d^2\theta}{dt^2}(t) = v'(t)$, dermed kan vi skrive

$$\underline{v'(t) = -g\sin(\theta(t))}$$

For å finne $\frac{d\theta}{dt}$, bruker jeg uttrykket for $v(t)$

$$V(t) = L\frac{d\theta}{dt}(t) \quad /:L$$

$$\underline{\frac{V(t)}{L} = \frac{d\theta}{dt}}$$

#

For å finne løsningen av (1) når vi vet (2) finner jeg sådan:

Vi har

$$\frac{d\theta}{dt} = \frac{v}{L}$$

$$\Rightarrow \frac{d^2\theta}{dt^2} = \frac{1}{L}\frac{dv}{dt} \quad | \cdot L$$

$$L\frac{d^2\theta}{dt^2} = \frac{dv}{dt} \qquad (*)$$

Setter $(*)$ inn i ligningen $\frac{dv}{dt} = -g\sin(\theta(t))$. dvs.

$$L\frac{d^2\theta}{dt^2} = -g\sin(\theta(t)),$$ som er ligning (1)

b) Ligningssystemet

$$(3) \quad \frac{dv}{dt} = -g\theta \quad , \quad \frac{d\theta}{dt} = \frac{v}{L} \quad (4)$$

Startverdier

$$\theta(0) = \theta_0 \quad , \quad v(0) = v_0$$

Deriverer (3)

$$\frac{d^2v}{dt^2} = -g\frac{d\theta}{dt}$$, ser at da har vi mulighet til

å utrykke (4) med (3)

$$\frac{d^2v}{dt^2}\frac{1}{-g} = \frac{d\theta}{dt}$$, Setter inn i (4)

$$\frac{d^2v}{dt^2}\frac{1}{-g} = \frac{v}{L} \quad | \cdot g$$

$$\frac{d^2v}{dt^2} = -\frac{gv}{L} \quad | + \frac{gv}{L}$$

$$0 = \frac{d^2v}{dt^2} + \frac{gv}{L}$$

Setter opp det karakteristiske polynomet.

$$r^2 + 0r + \frac{g}{L} = 0$$

Løser for r

$$\frac{0 - \pm\sqrt{0^2 - 4 \cdot g/L}}{2} = r$$

$$\frac{\pm 2\sqrt{g/L}\, i}{2} = r$$

$$r_1 = i\sqrt{g/L} \quad , \quad r_2 = -i\sqrt{g/L}$$

Generelt for 2 komplekse røtter
$r_1 = a + ib$, $r_2 = a - ib$, gir
$$y = e^{ax}(C\cos(bx) + D\sin(bx))$$

Dermed har vi $a = 0$, $b = i\sqrt{g/L}$
dvs.

$$\theta(t) = e^0 \left(C\cos\left(i\sqrt{g/L}\, x\right) + D\sin\left(i\sqrt{g/L}\, x\right) \right)$$
$$= \left(C\cos\left(i\sqrt{g/L}\, x\right) + D\sin\left(i\sqrt{g/L}\, x\right) \right)$$

Setter da inn startverdien. $\theta(0) = \theta_0$

$$\theta(0) = \theta_0 = C \underbrace{\cos(0)}_{1} + D \underbrace{\sin(0)}_{0}$$

$$\theta_0 = C$$

Vi har at $v'(t) = L \cdot \theta'(t)$

$$L \cdot \theta(0) = v'(0) = L\sqrt{\frac{g}{L}} \left(-\theta_0 \cdot \overbrace{\sin(0)}^{0} + D \overbrace{\cos(0)}^{1} \right)$$

$$v_0 = \sqrt{gL}\, D \quad \Big/ \frac{1}{\sqrt{gL}}$$

$$D = \frac{v_0}{\sqrt{gL}}$$

Dermed har vi løsningen

$$\theta(t) = \theta_0 \cos\left(\sqrt{\frac{g}{L}}\, t\right) + \frac{v_0}{\sqrt{gL}} \cdot \sin\left(\sqrt{\frac{g}{L}}\, t\right)$$