MAT-INF1100 H16: Obligatorisk oppgave 2

Ole K. Aamot
Enkeltemnestudent i matematikk
Universitetet i Oslo

9. november 2016

Oppgavene i denne obligatoriske oppgaven er løst i Python og skrevet i LaTeX av enkeltemnestudent Ole K. Aamot i emnet MAT-INF1100 ved Matematisk institutt, Universitetet i Oslo høsten 2016.

Obligatorisk oppgave 2

Oppgave 1

I denne oppgaven har vi målt farten v til et objekt som beveger seg og målingene er gjort ved $N+1$ tidspunktet $(t_i)_{i=0}^N$, slik at resultatet er en følge av tall-par $(t_i, v_i)_{i=0}^N$ der v_i angir farten ved tidspunktet t_i.

a)

Finner en algoritme for å beregne en tilnærmning til objektets akselleresjon $a(t) = v'(t)$ ut fra de beregnede verdiene (t_i,v_i) av farten.

$$a(t) = v'(t) = \frac{\Delta v_i}{\Delta t_i} = \frac{(v_{i+1} - v_i)}{(t_{i+1} - t_i)} \tag{1}$$

```
1  from math import sqrt
2  import matplotlib.pyplot as plt
3
4  class oppgave1a(object):
5      """Oppgave 1a) i obligatorisk oppgave 2, MAT-INF1100"""
6      def run(self):
7          c = 0
8          for i in range(0,c):
9              a.append((v[i]-v[i-1])/(t[i]-t[i-1]))
10             print t[i],a[i]
11 if __name__ == '__main__':
12     oppgave1a().run()
```

b)

Finner en algoritme for å beregne en tilnærming til objektets avstand $s(t)$ fra startpunktet ut fra de beregnede verdiene når $v(t) = s'(t)$ og $s(t_0) = 0$.

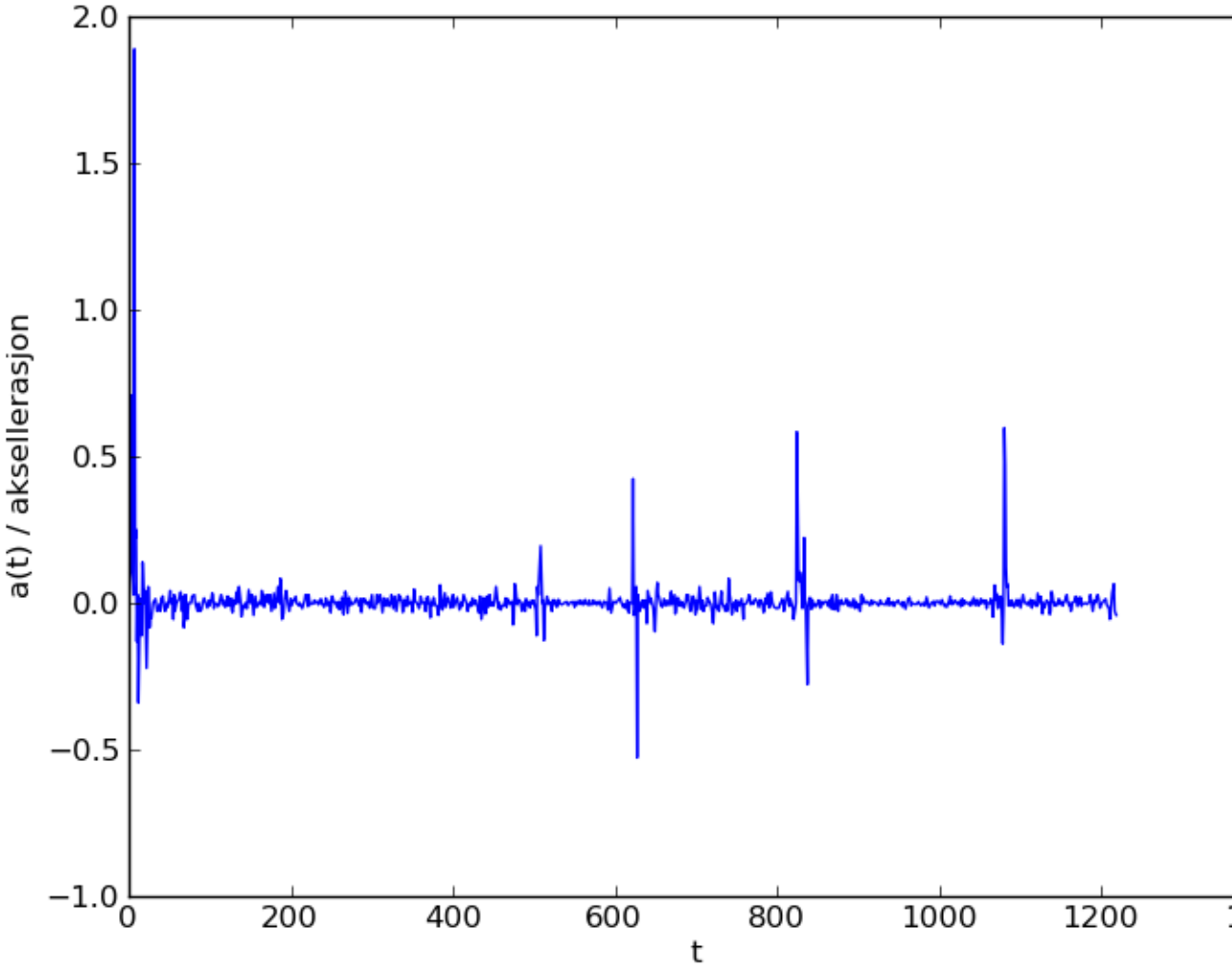
$$v(t) = s'(t) = \frac{\Delta s_i}{\Delta t_i} = \frac{(s_{i+1} - s_i)}{(t_{i+1} - t_i)} \tag{2}$$

```
1  from math import sqrt
2  import matplotlib.pyplot as plt
3
4  class oppgave1b(object):
5      """Oppgave 1b) i obligatorisk oppgave 2, MAT-INF1100"""
6      def run(self):
7          c = 0
8          for i in range(0,c):
9              v.append((s[i]-s[i-1])/(t[i]-t[i-1]))
10             print t[i],v[i]
11 if __name__ == '__main__':
12     oppgave1b().run()
```

c)

Implementerer lesing av loggfil i to vektorer t og v fra http://www.uio.no/studier/emner/matnat/math/MAT-INF1100/h16/obliger/running.txt hvor hver linje er kommaseparerte tid/-fart verdier og plotter objektets akselerasjon mot tid og objektets avstand fra startpunktet mot tid.

```
1
2  from math import sqrt
3  import matplotlib.pyplot as plt
4
5  class oppgave1c(object):
6      """Oppgave 1c) i obligatorisk oppgave 2, MAT-INF1100"""
7      def run(self):
8          t = []
9          v = []
10         a = []
11         x = []
12         y = []
13         s = []
14         c = 0
15         infile = open('running.txt','r')
16         for line in infile:
17             tnext, vnext = line.strip().split(',')
18             t.append(float(tnext))
19             v.append(float(vnext))
20             c = c + 1
21         infile.close()
22
23         for i in range(0,c):
24             print i,v[i]*t[i]
25             a.append((v[i]-v[i-1])/(t[i]-t[i-1]))
26             s.append((v[i]*t[i]))
27         plt.plot(a)
28         plt.xlabel("t")
29         plt.ylabel("a(t) / akselerasjon")
30         plt.show()
31 if __name__ == '__main__':
32     oppgave1c().run()
```



Figur 1: Plott av a(t) (akselleraSjon mot tid)

```
1
2  from math import sqrt
3  import matplotlib.pyplot as plt
4
5  class oppgave1c(object):
6      """Oppgave 1c) i obligatorisk oppgave 2, MAT-INF1100"""
7      def run(self):
8          t = []
9          v = []
10         a = []
11         x = []
12         y = []
13         s = []
14         c = 0
15         infile = open('running.txt','r')
16         for line in infile:
17             tnext, vnext = line.strip().split(',')
18             t.append(float(tnext))
19             v.append(float(vnext))
20             c = c + 1
21         infile.close()
22
23         for i in range(0,c):
24             print i,v[i]*t[i]
25             a.append((v[i]-v[i-1])/(t[i]-t[i-1]))
26             s.append((v[i]*t[i]))
27         plt.plot(s)
28         plt.xlabel("t")
29         plt.ylabel("s(t) / avstand")
30         plt.show()
31 if __name__ == '__main__':
32     oppgave1c().run()
```

Oppgave 2

a)

$$x' + x^2 = 1, x(0) = 0 \tag{3}$$

Ligningen er separabel og vi kan skrive den som

$$x' = -(x-1)(x+1) \tag{4}$$

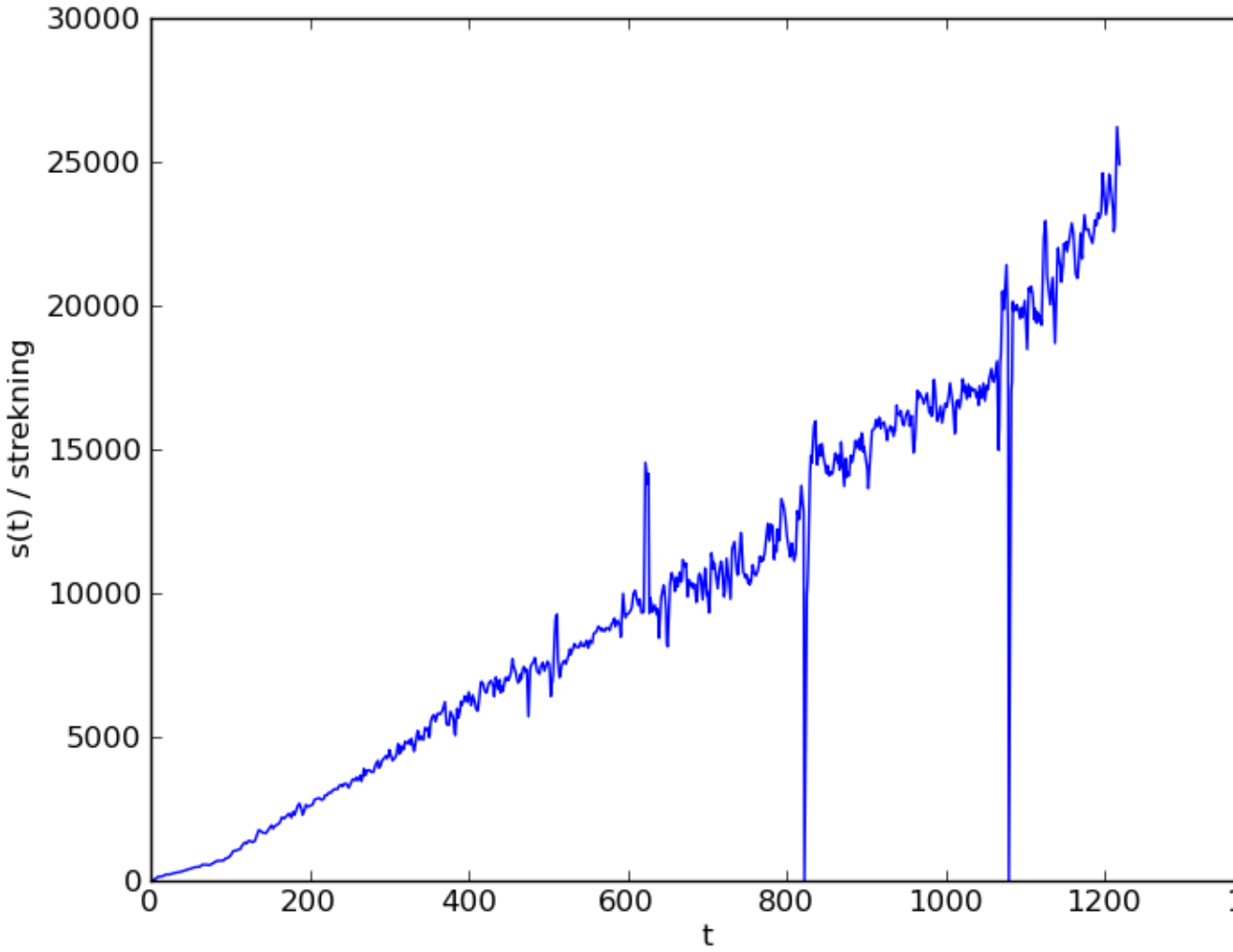
hvor $x(t)$ er ukjent. Vi regner ut løsningen $x(t)$ analytisk ved å integrere $x'(t)$.

$$x(t) = \int -(x-1)(x+1)d(t) = \frac{1}{3}x^3 + x + C \tag{5}$$

b)

Løser numerisk på intervallet [0,2] i 5 steg med Eulers metode og plotter den numeriske løsningen med den eksakte løsningen.

```
1  from numpy import *
2  from math import sqrt
3  import matplotlib.pyplot as plt
4
5  class oppgave2b(object):
6      """Oppgave 2b) i obligatorisk oppgave 2, MAT-INF1100"""
```



Figur 2: Plott av s(t) (avstand fra startpunkt mot tid)

```
7
8       def run(self):
9
10          f = lambda x,t: x+(1/3)*x**3
11          n = 5;
12          a = 0.001;
13          b = 2;
14          x = zeros(n+1)
15          t = linspace(a, b, n+1)
16          x[0] = 0;
17          h = float((b-a)/n);
18          for k in range(n):
19              x[k+1] = x[k]+h*f(x[k],t[k])
20          plt.plot(t,x,linewidth=3)
21          y = linspace(a,b,5);
22          plt.xlabel('t');
23          plt.ylabel('x(t)');
24          plt.show()
25  if __name__ == '__main__':
26      oppgave2b().run()
```

c)

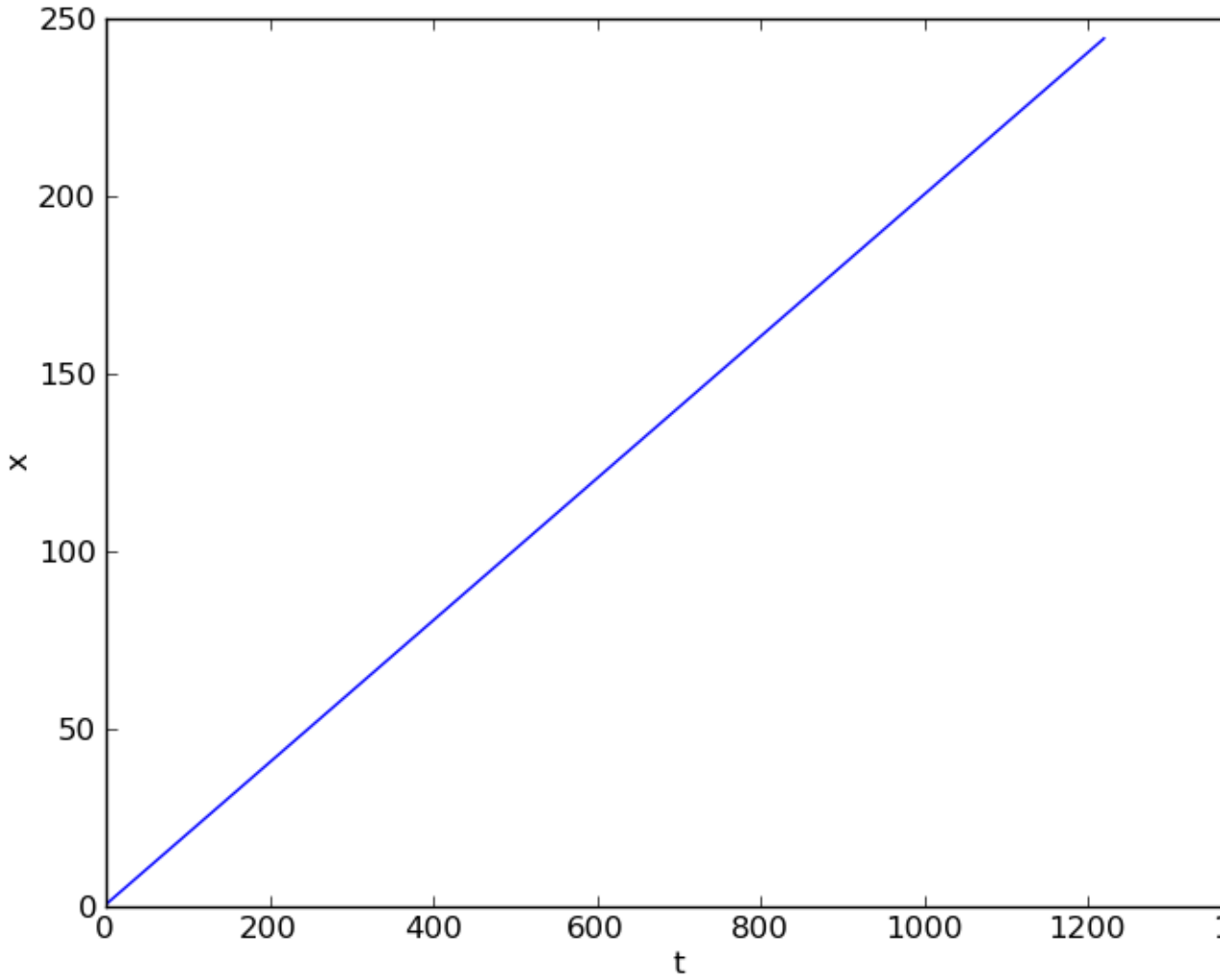
Gjentar **b)** og benytter Eulers midtpunktmetode, dvs. bestemmer Euler fra a til $(a+\frac{h}{2})$ og regner ut ny derivert i $(a+\frac{h}{2})$. Trekker denne tilbake til a og følger den til $(a+h)$.

Formel:

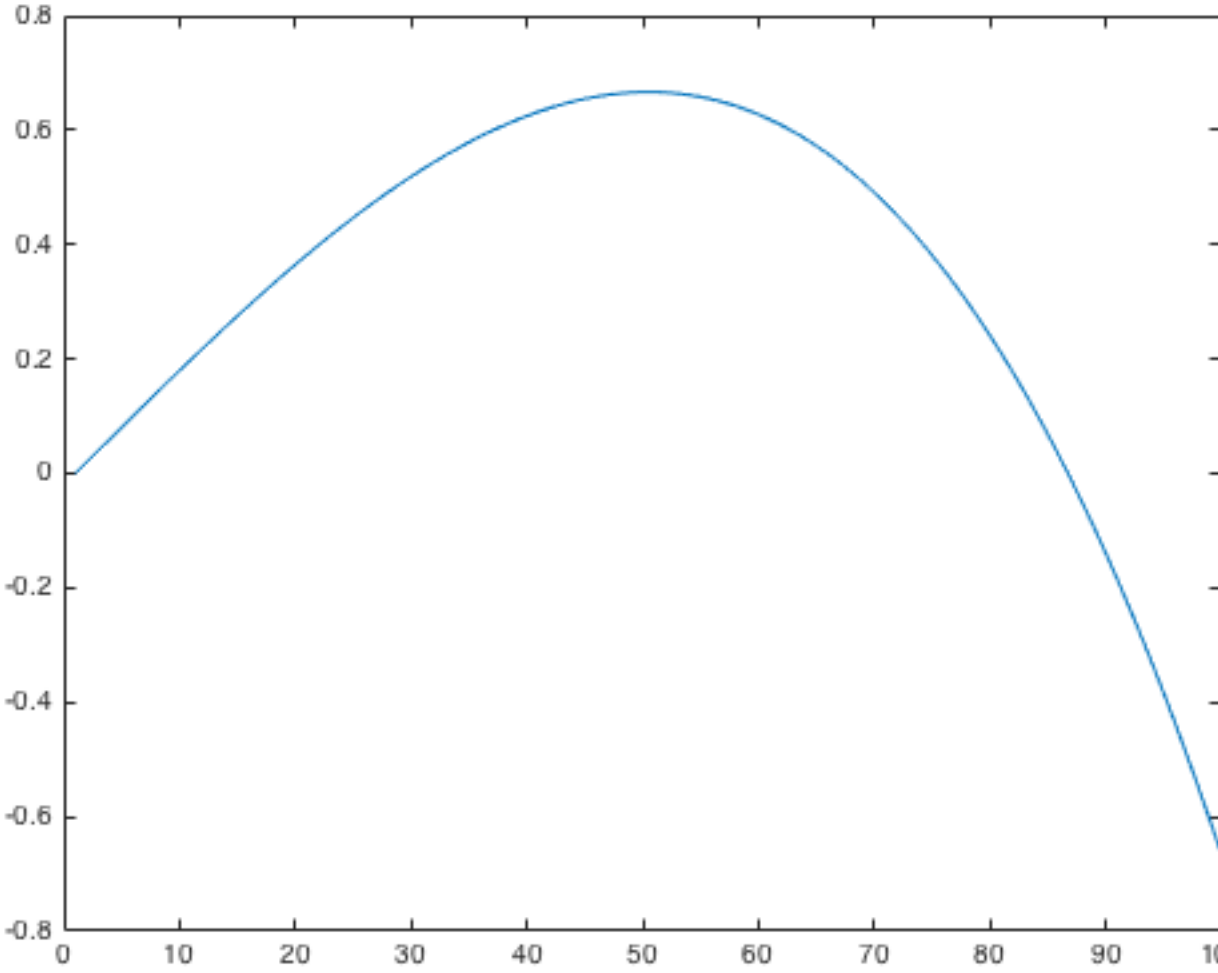
$$x_{k+\frac{1}{2}} = x_k + \frac{h}{2} f(t_n, x_n) \tag{6}$$

$$x_{k+1} = x_k + hf(t_n + \frac{h}{2}, x_n + \frac{1}{2}) \tag{7}$$

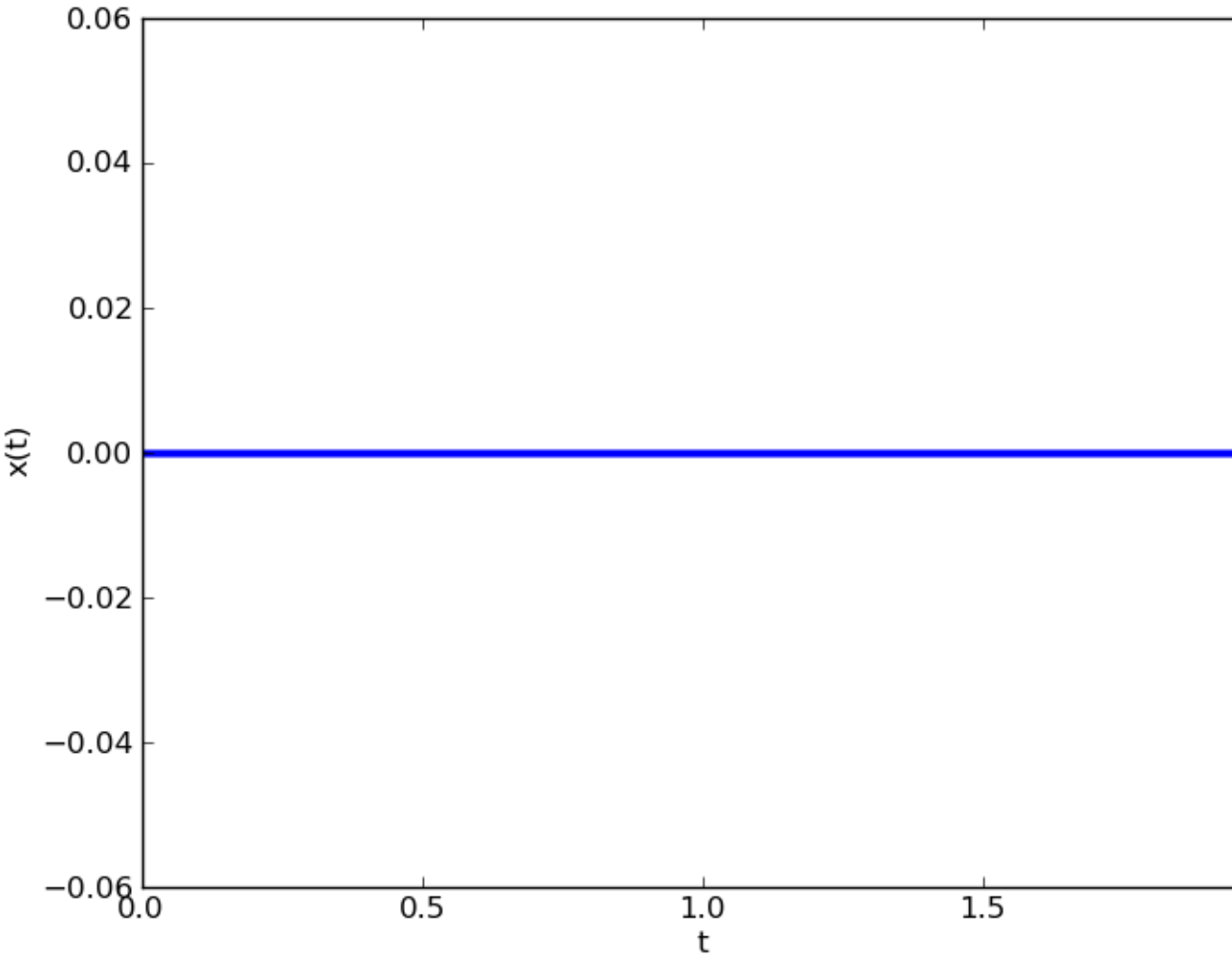
```
1   from numpy import *
2   from math import sqrt
3   import matplotlib.pyplot as plt
4
5   f = lambda x,t: x+(1/3)*x**3
6
7   class oppgave2c(object):
8       """Oppgave 2c) i obligatorisk oppgave 2, MAT-INF1100"""
9
10      def run(self):
11          t = []
12          v = []
13          a = []
14          x = []
15          y = []
16          s = []
17          c = 0
18
19          infile = open('running.txt','r')
20
21          for line in infile:
22              tnext, vnext = line.strip().split(',')
23              t.append(float(tnext))
24              v.append(float(vnext))
25              c = c + 1
26          infile.close()
27
28          n = 5;
29          a = 0.001;
30          b = 2;
```



Figur 3: Plott av numerisk Euler-tilpasning av funksjonen x(t)



Figur 4: Plott av eksakt funksjon x(t)



Figur 5: Plott av numerisk løsning av x(t) for [a,b] med Eulers metode i 5 steg

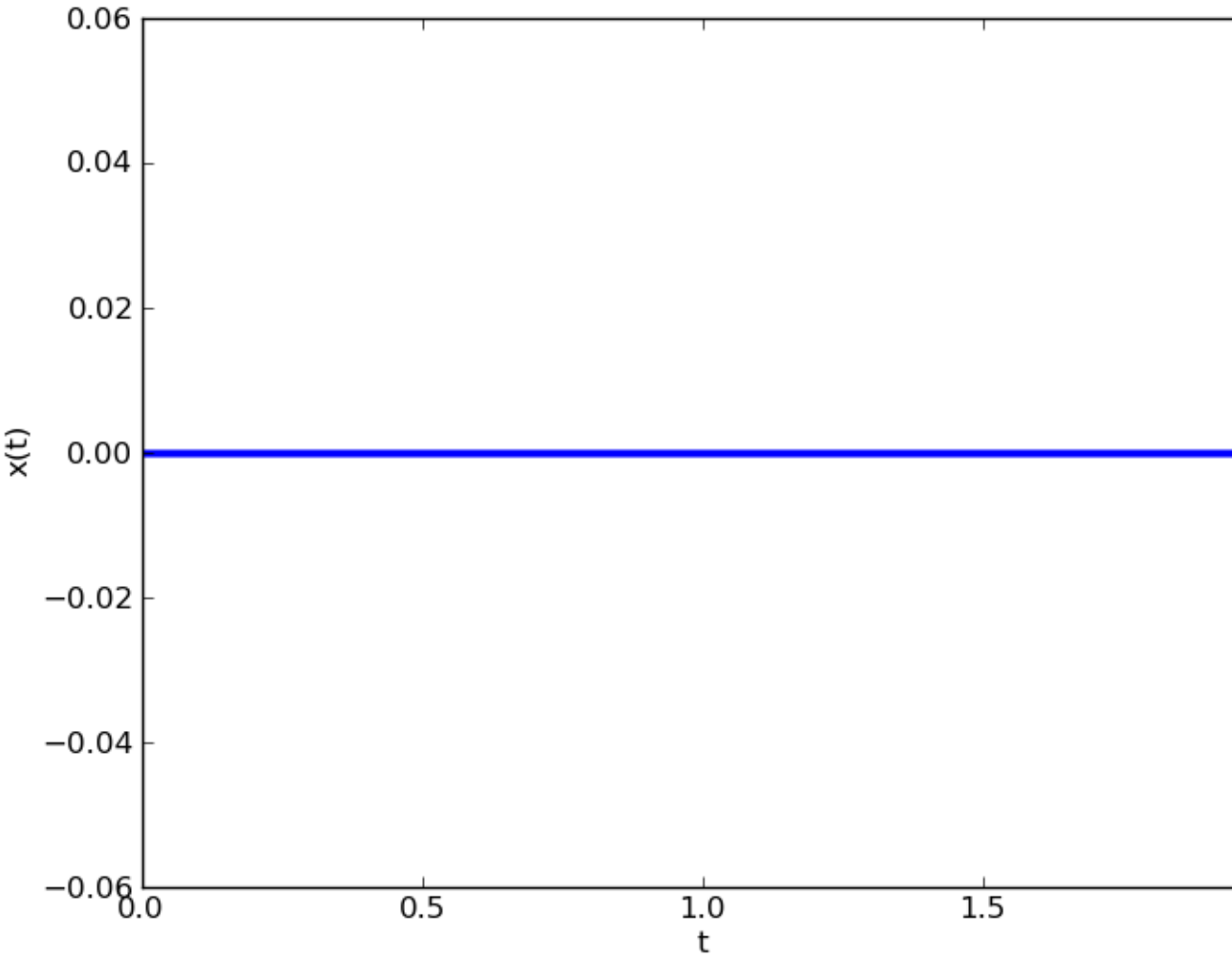
```
31          x = zeros(n+1)
32          t = linspace(a, b, n+1)
33          x[0] = 0;
34          h = float((b-a)/n);
35          for k in range(n):
36              x[k+1] = x[k]+(h/2)*f(x[k]+h/2,t[k]+1/2)
37          plt.plot(t,x,linewidth=3)
38          y = linspace(a,b,5);
39          plt.xlabel('t');
40          plt.ylabel('x(t)');
41          plt.show()
42  if __name__ == '__main__':
43      oppgave2c().run()
```

Vi ser at Eulers midtpunktmetode ikke gir riktig tilnærming til grafen i 5 steg sammenlignet med Figur 4 (eksakt x(t) funksjon), trolig pga. for korte intervaller (eller programmeringsfeil).

d)

Antar at $0 \leq x(t*) < 1$ for en verdi $t*$.

Om tangenten, dvs. den deriverte for uttrykket i punktet $t*$ er voksende, så vil $x(t)$ være voksende i punktet $t = t*$.



Figur 6: Plott av numerisk løsning av x(t) for [a,b] med Eulers midtpunktmetode i 5 steg