Oppgavesett uke 8

"Test deg selv"-oppgaver

Disse oppgavene er fine å gjøre hjemme før du kommer på gruppetimen. Om du har spørsmål til disse oppgavene, ta dem gjerne opp på snublegruppa.

Vi minner om definisjonen av den deriverte er gitt ved

$$\frac{df(x)}{dx} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x} \tag{1}$$

Når vi deriverer numerisk må vi velge en endelig (og fonuftig) størrelse på Δx. Vi er også begrenset til å regne ut den deriverte for bestemte verdier av x. Noen ganger har vi heller ikke et gitt matematisk uttrykk for funksjonen $f(x)$. Dette kan for eksempel skje dersom vi måler posisjonen til en gjenstand ved bestemte tidspunkter og vil finne farta. I slike tilfeller har vi i praksis lister med samsvarende verdier x_i og $f_i = f(x_i)$. Når man deriverer numerisk må man derfor omskrive likning (1) til en av de to formene

$$\frac{df(x)}{dx} \to \frac{\Delta f}{\Delta x} = \frac{f(x+\Delta x) - f(x)}{\Delta x} = \frac{f_{n+1} - f_n}{x_{n+1} - x_n}. \tag{2}$$

TDS 8.1 Numerisk integrasjon

I denne oppgaven skal vi bruke venstre trappesum for å finne integralet av noen funksjoner numerisk.

For en funksjon $f(x)$ og et startpunkt x_0 definerer vi det bestemte integralet

$$F(x) = \int_{x_0}^{x} f(x)dx.$$

Venstre trappesum tilnærmer $F(x)$ som

$$F(x+\Delta x) = F(x) + f(x)\Delta x$$

hvor Δx er et lite intervall på x-aksen.

a) Skriv en funksjon i Python som har input en array med koordinater x_i og tilsvarende funksjonsverdier $f(x_i)$ og returnerer en array $F(x_i)$.

Løsning:

```
1  import numpy as np
2
3  def FE(f,x,C = 0):
4      F = np.zeros(len(f))
5      for i in range(len(f)-1):
6          dx = x[i+1]-x[i]
```

```
7           F[i+1] = F[i] + f[i]*dx
8       return F, x
```

b) Numerisk integrer følgende funksjoner med steglengde $dx = 0.0001$ og på intervallet $x \in [\frac{\pi}{16}, \frac{7\pi}{16}]$. Visualiser integralet.

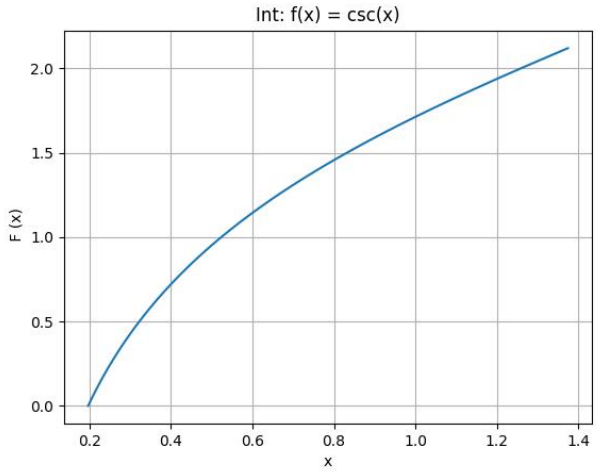
Løsning:

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4
5  x_0 = np.pi/16
6  x_n = 7*np.pi/16
7
8  xlen = x_n-x_0
9
10 dx = 0.0001
11 n = int(xlen/dx)
12
13 x = np.linspace(x_0,x_n,n)
14
15 def FE(f,x,C = 0):
16     F = np.zeros(len(f))
17     for i in range(len(f)-1):
18         dx = x[i+1]-x[i]
19         F[i+1] = F[i] + f[i]*dx
20     return F, x
21
22 f1 = lambda x: 1/np.sin(x)
23
24 F1,x = FE(f1(x),x)
25 plt.plot(x,F1)
26 plt.grid()
27 plt.xlabel("x")
28 plt.ylabel(" F (x)")
29 plt.title("Int: f(x) = csc(x)")
30 plt.savefig("TDS08task3ba.jpg")
31 plt.show()
32
33 f2 = lambda x: np.sin(x)/x
34
35 F2,x = FE(f2(x),x)
36 plt.plot(x,F2)
37 plt.title("Int: f(x) = sin(x)/x")
38 plt.grid()
39 plt.xlabel("x")
40 plt.ylabel(" F (x)")
41 plt.savefig("TDS08task3bb.jpg")
```

```
42  plt.show()
```

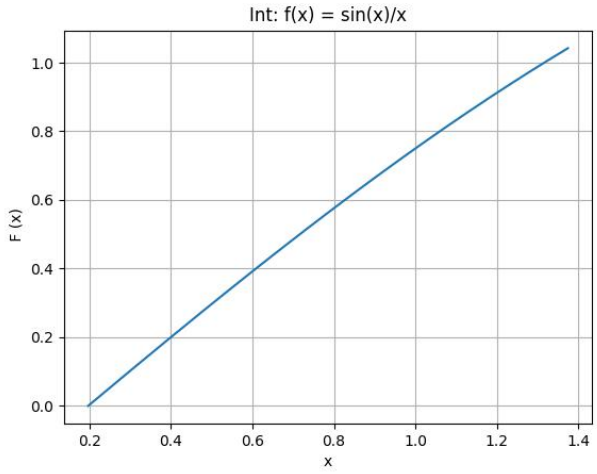
(a) $f(x) = \csc(x)$

Løsning:



(b) $f(x) = \frac{\sin(x)}{x}$

Løsning:



(c) Integralet

$$I = \int_0^{\pi} \csc(x) dx$$

er udefinert. Hva vil skje dersom man prøver å beregne integralet numerisk? Kan den numeriske algoritmen gi deg et svar og vil dette eventuelle svaret være riktig? Prøv å gjennomføre integralet numerisk og se hva som skjer. *Hint: Plott* $f(x)$.

Løsning: Det vil rent teknisk sett være mulig å integrere $f(x)$ ettersom man jobber med et sett diskrete punkter, som ikke nødvendigvis inneholder punktene som gir singulariteter. Dersom settet med diskrete punkter inneholder en verdi som gir en singularitet vil det ikke være "mulig" å gjøre det numerisk. Et singulært punkt er feks $x = 0$ som gir $f(0) = \infty$ Det at integralet "kan" regnes ut numerisk betyr derimot ikke at integralet er riktig, ettersom det ikke er veldefinert for $x = 0$ og $x = \pi$. Derfor konkluderer vi med at integralet ikke vil være mulig å regne ut riktig numerisk.

TDS 8.2 Baseball

En baseball har masse $m = 0{,}145$ kg. Ballen kastes med $v_0 = 45$ m/s, og etter slaget beveger ballen seg med $v_1 = 55$ m/s i motsatt retning.

a) Hvor stor er impulsen fra balltreet?

Løsning: $J_x = \Delta p_x = m(v_1 - v_0) = 14{,}5$ kgm/s.

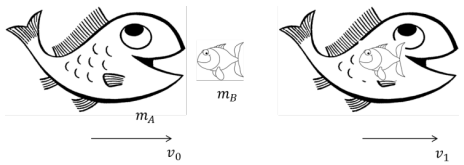
b) Hvis ballen forblir i kontakt med balltreet i $\Delta t = 2{,}0$ ms, hvor stor er den gjennomsnittlige kraften fra balltreet på ballen?

Løsning:

$$F_{av,x} = \frac{J_x}{\Delta t} = \frac{14{,}5 \text{ kgm/s}}{2{,}0 \text{ ms}} = 7250 \text{ N} \approx 7{,}3 \text{ kN}.$$

TDS 8.3 To fisker

En fisk på 20 kg svømmer i havet med $v = 1{,}0$ m/s og spiser opp en mindre fisk på 5,0 kg som står helt stille. Du trenger ikke ta hensyn til motstandskraften i vannet.



a) Med hvilken hastighet svømmer den store fisken videre etter den har spist den mindre?

Løsning: Spisingen kan anses som en fullstendig uelastisk kollisjon hvor driv er bevart:

$$m_A v_0 = (m_A + m_B) v_1$$
$$v_1 = \frac{m_a v_0}{m_A + m_B} = 0,80\text{m/s}.$$

b) Hvor mye mekanisk energi ble tapt under middagen?

Løsning:

$$\Delta E = \frac{1}{2} m_A v_0^2 - \frac{1}{2}(m_A + m_B) v_1^2 = 2,0\text{ J}$$

TDS 8.4 Ukas difflikning

Finn løsninga til differensiallikninga

$$\ddot{x} + 4x + \dot{x} = 0$$

med initialbetingelsene

$$x(0) = 1 \quad \dot{x}(0) = 0.$$

Løsning: Vi tipper $x(t) = e^{\lambda t}$ og får

$$\lambda^2 + \lambda + 4 = 0$$

som gir

$$\lambda_\pm = \frac{-1 \pm \sqrt{1-16}}{2} = -\frac{1}{2} \pm \frac{i}{2}\sqrt{15}.$$

Den generelle løsninga er da

$$x(t) = Ae^{\lambda_+ t} + Be^{\lambda_- t} = e^{-t/2}(C\cos\omega t + D\sin\omega t)$$

der $\omega = \sqrt{15}/2$. Initialbetingelsene gir

$$x(0) = C = 1$$
$$\dot{x}(0) = -\frac{1}{2} + D\omega = 0, \qquad D = \frac{1}{\sqrt{15}}$$

Dermed er løsinga

$$x(t) = e^{-t/2}(\cos \omega t + \frac{1}{\sqrt{15}} \sin \omega t).$$

Øvingsoppgaver

Disse oppgavene kommer vi til å jobbe med i øvingstimene. Men det er nok ofte mer å gjøre enn du rekker i timen, så det kan være lurt å forberede seg ved å se på dem på forhånd slik at du kan spørre om det som du ikke får til.

G 8.1 Trapesmetoden

I denne oppgaven skal vi sammenlikne to metoder for numerisk integrasjon: Venstre trappesum og trapesmetoden.

For venstre trappesum er integralet tilnærmet med

$$I = \sum_{i=0}^{n-1} f(x_i)\Delta x \tag{3}$$

mens for trapesmetoden bruker vi

$$I = \sum_{i=}^{n-1} \frac{f(x_i) + f(x_{i+1})}{2}\Delta x \tag{4}$$

Vi vil studere funksjonen

$$f(x) = \frac{1}{\sqrt[3]{2x^4 - x^3 + \pi}},$$

og integrere den på intervallet $x \in [1, 10]$.

a) Bruk venstre trappesum og trapesmetoden for å numerisk finne arealet under grafen til $f(x)$. Sammenlign verdiene du får med de to metodene. Husk å bruke samme steglengde på begge metodene samt samme intervall for x.

Løsning: Fra terminalen får vi at arealet funnet ved trapesmetoden ble $I \approx 1.2884$, mens arealet funnet ved venstre trappesum ble $I \approx 1.2886$, så vi ser at de er relativt like for $n = 10000$ datapunkter. Vi kan da anta at dette er nære den virkelige verdien til integralet.

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  n = 10000
5
6
7  x = np.linspace(1,10,n)
8
9
10 f3 = lambda x: 1/(2*x**4 -x**3 + np.pi)**(1/3)
11
12 I_F = 0
13
```

```
for i in range(n - 1):
    I_F = I_F + f3(x[i]) * (x[i+1]-x[i])



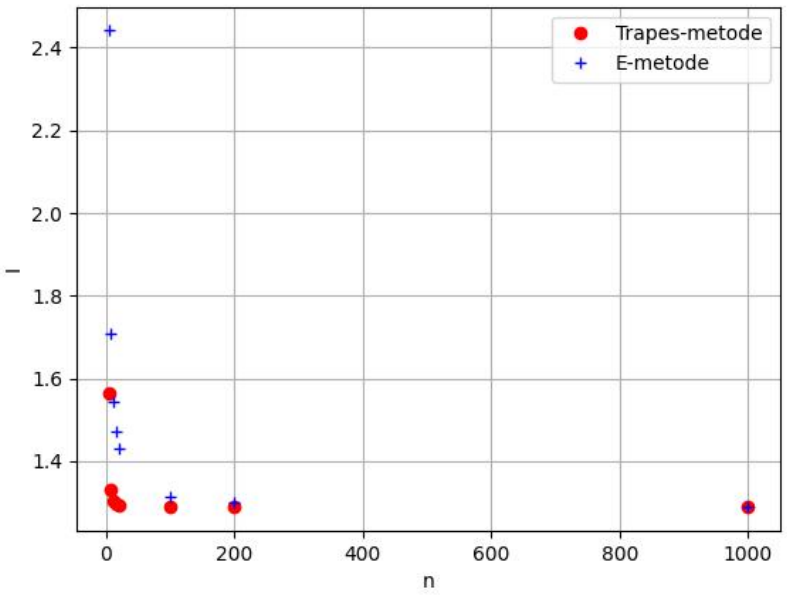
def trapes(f,x):
    I = 0
    for i in range(len(f)-1):
        I += (x[i+1]-x[i])* (f[i+1]+f[i])/2
    return I

I_T = trapes(f3(x),x)

print(f"Arealet funnet ved trapesmetoden I = {I_T}, arealet funnet↩
    ved Eulers metode I = {I_F}")
```

b) Forsøk deretter å sammenlikne de to metodene for ulike steglengder. For eksempel kan det være passe å $n = \{4, 8, 12, 16, 20, 100, 200, 1000\}$ datapunkter. Plott arealet I under $f(x)$ for de to metodene som en funksjon av antall datapunkter. Hvilken av metodene konvergerer først? Hva vil det si at en numerisk metode konvergerer før en annen?

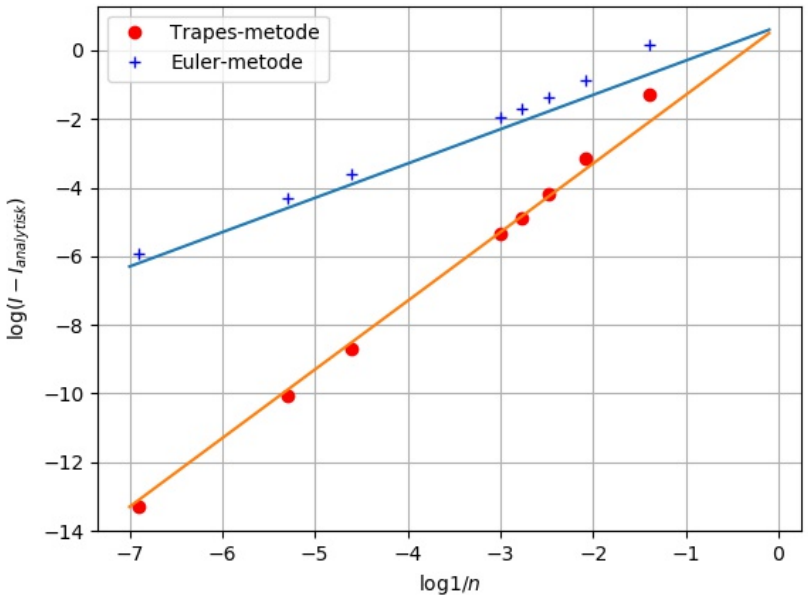
Løsning:



Som vi ser fra figuren over så konvergerer trapesmetoden vesentlig raskere enn venstre trappesum. Vi kan da trekke slutningen at trapesmetoden er en bedre metode for store verdier av Δx og færre

antall datapunkter. Det at en av metodene konvergerer raskere mot en verdi betyr at metoden er bedre egnet og som oftest mer nøyaktig når det kommer til numerisk integrasjon.

Vi ser også at figuren ikke er lett å lese fordi all punktene kommer langs aksene, og det er vanskelig å se tydelig forskjellene på de to metodene. Dette kommer av at vi har verdier av n som er ganske forskjellige slik at de små verdiene kommer langt inn på aksen for at vi skal få med den største ($n = 1000$). Tilsvarende er det noen store feil for små n som gjør at det er vanskelig å se forskjell på de små feilene for store n. En vanlig metode for å hjelpe på dette er å plotte med logaritmiske akser. Figuren under viser $\log(I - I_{analytisk})$ som funksjon av $\log(1/n)$, og da ser vi mye tydeligere forskjellen i konvergens for de to metodene.

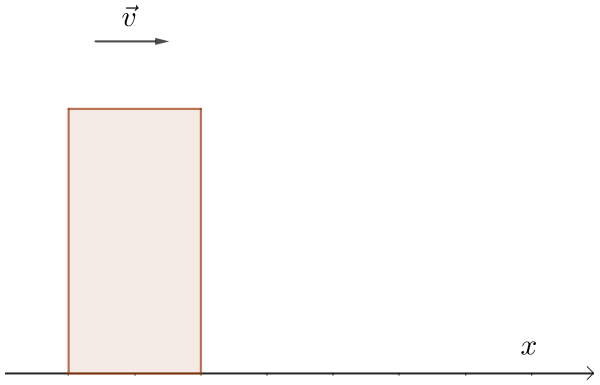



```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  nlist = [4,8,12,16,20,100,200,1000]
5  for n in nlist:
6
7      x = np.linspace(1,10,n)
8
9
10     f3 = lambda x: 1/(2*x**4 -x**3 + np.pi)**(1/3)
11
12     I_F = 0
13
14     for i in range(n - 1):
15         I_F = I_F + f3(x[i]) * (x[i+1]-x[i])
16
17
18
```

```
19      def trapes(f,x):
20          I = 0
21          for i in range(len(f)-1):
22              I += (x[i+1]-x[i])* (f[i+1]+f[i])/2
23          return I
24
25      I_T = trapes(f3(x),x)
26
27      plt.plot(n,I_T,"or")
28      plt.plot(n,I_F,"b+")
29
30
31      print(f" n={n} Trapesmetoden gir at arealet under grafen er I ↩
            = {I_T}, Eulers metode derimot gir I = {I_F}")
32  plt.ylabel("I")
33  plt.xlabel("n")
34  plt.legend(["Trapes-metode","E-metode"])
35  plt.grid()
36  plt.savefig("Sammenligning.jpg")
37  plt.show()
```

G 8.2 Eskild Eskensen

I denne oppgaven skal vi se på farten til en person som har vært på joggetur.



Figuren over viser en noe forenkelt person, Eskild Eskensen, på joggetur. Vi skal numerisk finne akselerasjonen til Eskild, strekningen han har tilbakelagt, samt analysere og visualisere resultatet. Senere skal dere gjøre samme, men med et datasett med små mengder støy for å se hvordan det endrer de numeriske resultatene.

a) Skriv et skript i Python som leser inn farten og korresponderende tidspunkter fra filen velocity.txt og plott farten som en funksjon av tid. Kan dere se fra farten når Eskild akselererer negativt og positivt? Hvor lang tid varte joggeturen? Husk aksenavn!

Løsning:

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

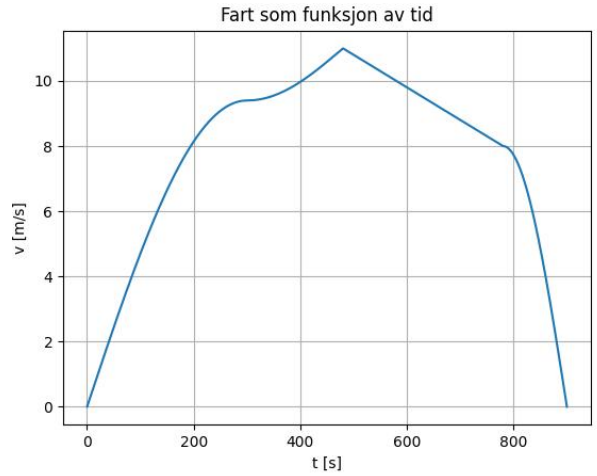
v = []
t = []
with open("velocity.txt", "r") as infile:
    f = infile.readline()
    f = infile.readline()
    for line in infile:
        a = line.split(sep=", ")
        v.append(float(a[0]))

        t.append(float(a[1]))

v = np.array(v)
t = np.array(t)




if __name__ == "__main__":
    plt.grid()
    plt.plot(t,v)
    plt.ylabel("v [m/s]")
    plt.xlabel("t [s]")
    plt.title("Fart som funksjon av tid")
    plt.show()
    print(f"Joggeturen varte i {t[-1] / 60:g} minutter")
```



Vi kan se fra figuren at i begynnelsen så er akselerasjonen til Eskild positiv siden farten blir større.

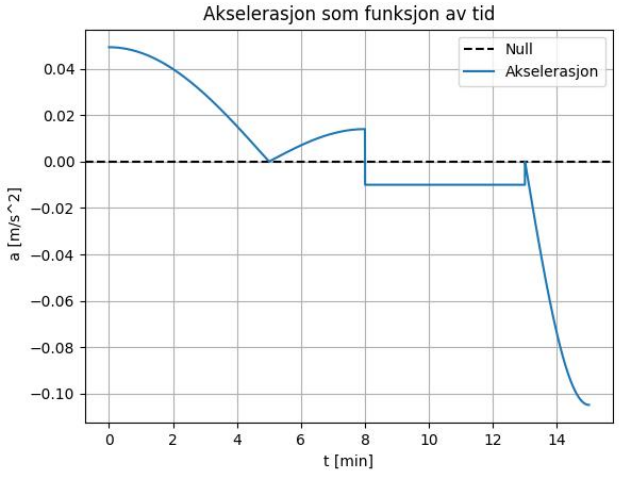
Men akselerasjonen avtar fram til ca 300 s siden farta øker saktere. Ved omlag 300 sekunder øker akselerasjonen igjen. Deretter er akselerasjonen negativ fra ca 500 sekunder og ut joggeturen.

Fra koden så ser vi at joggeturen til Eskild varte i 15 minutter.

b) Numerisk deriver farten til Eskild og plott akselerasjonen som en funksjon av tid. Stemmer akselerasjonen med det dere kom frem til i forrige deloppgave?

Løsning:

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  from S1a import v,t
4
5  def num_deriv(x,t):
6      dxdt = []
7      for i in range(len(x)-1):
8          deriv = (x[i+1]-x[i])/(t[i+1]-t[i])
9          dxdt.append(deriv)
10     return np.array(dxdt), t[0:len(dxdt)]
11
12 a,t_a = num_deriv(v,t)
13 if __name__ == "__main__":
14     plt.axhline(y=0,linestyle = "--",color = "k", label = "Null")
15     plt.plot(t_a/60,a, label = "Akselrasjon")
16     plt.legend()
17     plt.grid()
18     plt.xlabel("t [min]")
19     plt.ylabel(" a [m/s^2]")
20     plt.title("Akselerasjon som funksjon av tid")
21     plt.show()
```



Vi kan se at akselerasjonen stemmer godt med det vi antok fra kun å studere fartsgrafen.

c) Numerisk integrer farten for å finne strekningen tilbakelagt per sekund. Visualiser resultatet. Hva var den totale distansen tilbakelagt på joggeturen til Eskild?

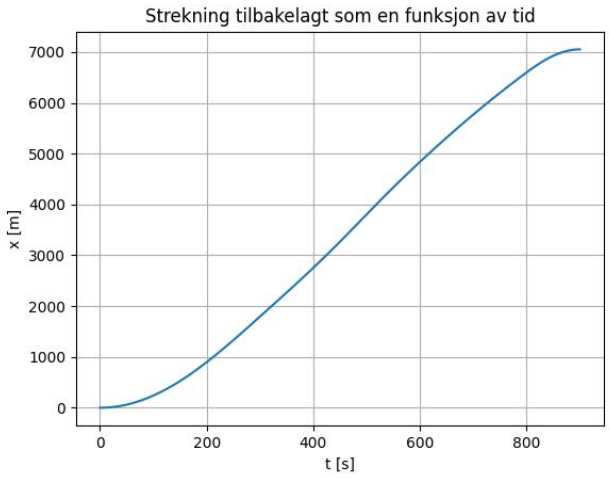
Løsning:

```
import numpy as np
import matplotlib.pyplot as plt
from S1a import v,t



def num_int(v,t):
    x = np.zeros(len(v))
    for i in range(len(v)-1):
        dt = t[i+1]-t[i]
        x[i+1] = x[i]+v[i]*dt
    return x
x = num_int(v,t)
if __name__ == "__main__":
    plt.plot(t,x)
    plt.grid()
    plt.xlabel("t [s]")
    plt.ylabel("x [m]")
    plt.title("Strekning tilbakelagt som en funksjon av tid")
    plt.show()

    print(f"Den totale strekningen tilbakelagt var {x[-1]:g}m")
```



Fra koden ser vi at den totale distansen tilbakelagt var 7053.64m.

d) Gjør deloppgave a, b og c igjen, men nå med filen velocitynoise.txt. Sammenlign resultatene med og uten støy ved å plotte dem sammen. Hvor stor er den typiske størrelsen av støyen i farta?

Ser dere en forskjell i de numeriske beregningene når man legger til en liten mengde støy? Hva er absoluttverdien av differansen mellom tilbakelagt strekning med og uten støy? Kommenter resultatene.

Løsning:

```
import numpy as np
import matplotlib.pyplot as plt
from S1a import v,t
from S1b import a,t_a,num_deriv
from S1c import num_int,x


# Importerer datapunktene med støy
vs = []
ts = []
with open("velocitynoise.txt", "r") as infile:
    f = infile.readline()
    f = infile.readline()
    for line in infile:
        linesegments = line.split(sep=", ")
        vs.append(float(linesegments[0]))

        ts.append(float(linesegments[1]))

vs = np.array(vs)
ts = np.array(ts)



a_noise,t_noise =num_deriv(vs,ts)

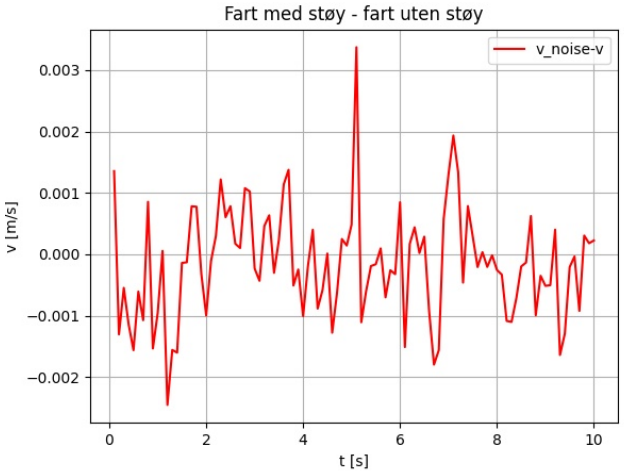
#Plotter de to hastighetene
plt.plot(t,v,"r",label = "v")
plt.plot(ts,vs,"--",label = "v_n")
plt.grid()
plt.ylabel("v [m/s]")
plt.xlabel("t [s]")
plt.title("Fart med og uten støy")
plt.legend()
plt.show()

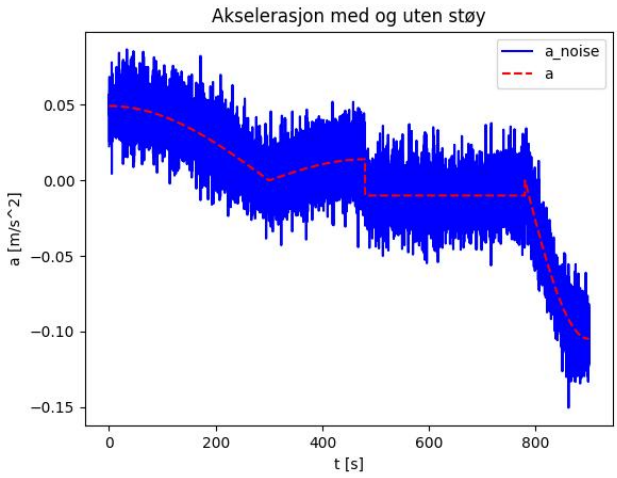


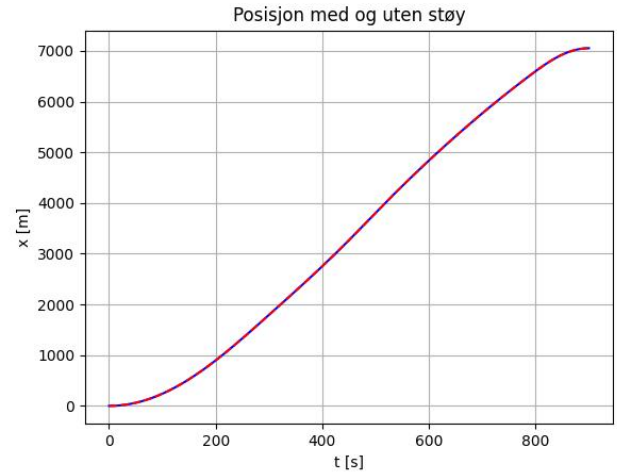
#Plotter de to akselerasjonene
plt.plot(t_noise,a_noise,"b",label = "a_noise")
plt.plot(t_a,a,"--r",label = "a")
plt.axis()
plt.legend()
plt.ylabel("a [m/s^2]")
```

```
45 plt.xlabel("t [s]")
46 plt.title("Akselerasjon med og uten støy")
47 plt.show()
48
49 #Plotter posisjonen med og uten støy
50 x_noise = num_int(vs,ts)
51 plt.plot(t,x_noise,"b",label = "x_noise")
52 plt.plot(t,x,"--r",label = "x")
53 plt.xlabel("t [s]")
54 plt.ylabel("x [m]")
55 plt.grid()
56 plt.title("Posisjon med og uten støy")
57 plt.show()
58
59 print(f"Differansen mellom tilbakelagt strekning med og uten støy ↩
    er {np.abs(x[-1]-x_noise[-1])}m")
```





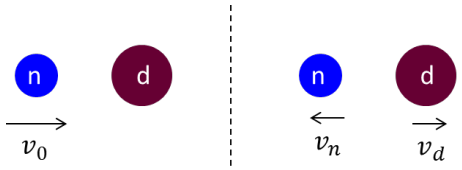




Støyen er relativt liten i forhold til størrelsen på farten, noe som gjør at vi ikke ser forskjellen når vi plotter fart med og uten støy sammen. For å se hvor stor støy det er plotter vi differansen mellom fart med og uten støy, og sitter igjen med bare støyen. Da ser vi at den er like mye positiv og negativ, og med typiske verdier rundt 0,001 m/s. Dette gjenspeiler seg også i posisjonsberegningene, der hvor støyen ikke gir et særlig stort bidrag. Dette er fordi støyen har like stor sannsynlighet for å være positiv som negativ, og når vi integrerer får vi derfor noen ganger en litt for stor verdi for funksjonen og noen ganger en litt for liten. Når vi summerer sammen disse bidragene kansellerer de positive og negative støverdiene hverandre og absoluttverdien av differansen mellom den tilbakelagte strekningen med og uten støy er omlag $|\Delta x| = 0{,}00575042$ m. Akselerasjonen derimot den deriverte av farta, og selv om støyen er liten i verdi, deles hastighetsdifferansen med det lille tidsintervallet mellom hvert datapunkt, og da kan feilen bli stor.

G 8.3 Kjernereaksjon

I tungtvannsmodererte kjernereaktorer kolliderer nøytroner med masse m_n med deuteroner med masse m_d. I en slik reaktor kolliderer et nøytron med hastighetv_0 elastisk med et deuteron som er i ro. Kollisjonen er kortvarig og ytre krefter som gravitasjon er neglisjerbare i forhold til de indre kreftene mellom nøytronet og deuteronet.



a) Hva er hastigheten til nøytronet, uttrykt som en brøkdel av den opprinnelige hastigheten, etter kollisjonen? **Løsning:** Siden kollisjonen er kortvarig og ytre krefter som gravitasjon er neglisjerbare

i forhold til de indre kreftene mellom nøytronet og deuteronet, er drivet under kollisjonen bevart, $m_n v_0 = m_n v_n + m_d v_d$, som gir at

$$v_d = \frac{m_n}{m_d}(v_0 - v_n). \tag{5}$$

Kollisjonen er elastisk og vi bruker bevaring av energi

$$\frac{1}{2}m_n v_0^2 = \frac{1}{2}m_n v_n^2 + \frac{1}{2}m_d v_d^2$$

eller

$$v_d^2 = \frac{m_n}{m_d}(v_0^2 - v_n^2) = \frac{m_n}{m_d}(v_0 - v_n)(v_0 + v_n). \tag{6}$$

Hvis vi deler (6) på (5) får vi

$$v_d = v_0 + v_n$$

som vi kan sette inn i (5) og få

$$v_0 + v_n = \frac{m_n}{m_d}(v_0 - v_n).$$

Vi løser for

$$v_n = -\frac{m_d - m_n}{m_d + m_n}v_0.$$

b) Hvor mye kinetisk energi E_n, uttrykt som en brøkdel av den opprinnelige kinetiske energien E_0, har nøytronet igjen etter kollisjonen? **Løsning:**

$$E_n = \frac{1}{2}m_n v_n^2 = \left(\frac{m_d - m_n}{m_d + m_n}\right)^2 E_0.$$

c) Hvor mye kinetisk energi, uttrykt som en brøkdel av den opprinnelige kinetiske energien, har nøytronet igjen etter fem slike kollisjoner? **Løsning:** I hver kollisjon minker energien med en faktor $\left(\frac{m_d - m_n}{m_d + m_n}\right)^2 \approx \frac{1}{9}$. Etter 5 kollisjoner er

$$E_n \approx \left(\frac{1}{9}\right)^5 = \frac{E_0}{59049}.$$

Ukas utfordring

Denne oppgava krever mer enn vanlig kompliserte resonnement eller utregninger. Det er en utfordring for studenter som føler at de mestrer stoffet og trenger noe ekstra å jobbe med.

G 8.4 Falle fra månen

Hvor lang tid tar det å falle fra månen til jorda? Vi antar at vi starter i ro i fohold til jorda i samme avtand som månen (men langt fra månen, så vi ikke trenger å ta hensyn til tyngdekrafta fra den), og vil finne tida det tar før vi når jordoverflata. Vi kaller avstanden fra jordsenteret til startposisjonen R_0 og til jordoverflata R.

a) Bruk dimensjonsanalyse til å finne et uttrykk for tida T det tar. Uttrykket kommer til å inneholde en ubestemt funksjon som du ikke kan finne med dimensjonsanalyse.

Løsning: Vi har disse størrelsene som inngår

Gravitasjonskonstanten	G	$[G] = \frac{\mathrm{m}^3}{\mathrm{kg\ s}^2}$
Jordmassen	M	$[M] = \mathrm{kg}$
Avstanden til startpunktet	R_0	$[R_0] = \mathrm{m}$
Avstanden til sluttpunktet	R	$[R] = \mathrm{m}$
Tida for fallet	T	$[T] = \mathrm{s}$

(7)

Vi finner da at uttrykket for tida må ha formen

$$T = \sqrt{\frac{R_0^3}{GM}} F\left(\frac{R}{R_0}\right).$$

b) Finn et uttrykk for farta i avstanden r.

Løsning: Bevaring av energi gir (m er massen til det fallende legemet)

$$-G\frac{Mm}{R_0} = -G\frac{Mm}{r} + \frac{1}{2}mv^2$$

som vi løser for v og finner

$$v = -\sqrt{2GM}\sqrt{\frac{1}{r} - \frac{1}{R_0}}.$$

c) Finn tida T som trengs for å nå R. Her kan du trenge integralet

$$\int_x^1 \sqrt{\frac{u}{1-u}}du = \sqrt{x(1-x)} + \arccos\sqrt{x}.$$

Løsning: Vi har

$$v = \frac{dr}{dt} = -\sqrt{2GM}\sqrt{\frac{1}{r} - \frac{1}{R_0}}.$$

Vi separerer variablene og får

$$-\int_{R_0}^{R} \frac{dr}{\sqrt{\frac{1}{r} - \frac{1}{R_0}}} = \sqrt{2GM}\int_0^T dt = \sqrt{2GM}T.$$

Hvis vi innfører $u = r/R_0$ og $x = R/R_0$ får vi

$$-\int_{R_0}^{R} \frac{dr}{\sqrt{\frac{1}{r} - \frac{1}{R_0}}} = R_0^{3/2}\int_x^1 \sqrt{\frac{u}{1-u}}du = \sqrt{x(1-x)} + \arccos\sqrt{x}.$$

Dermed er

$$T = \sqrt{\frac{R_0^3}{2GM}}\left(\sqrt{x(1-x)} + \arccos\sqrt{x}\right).$$

d) Vi vil sjekke uttrykket for et kjent tilfelle. Hvis $h = R_0 - R \ll R_0$ (som ikke er tilfelle om vi faller helt fra månen il jorda, bare en mye kortere strekning) har vi tilnærmet fritt fall med konstant akselerasjon. Finn tida det tar å falle strekningen h med kosntant akselerasjon $g_0 = \frac{GM}{R_0^2}$.

Løsning: Vi har $h = \frac{1}{2} g_0 T^2$ og dermed er tida

$$T = \sqrt{\frac{2h}{g_0}}.$$

e) Prøv å rekkeutvikle uttrykket fra c) i ei Taylorrekke for $y = 1 - x \ll 1$. Forklar hvorfor det ikke går an.

Løsning: Vi har at

$$T = \sqrt{\frac{R_0^3}{2GM}} \left(\sqrt{y(1-y)} + \arccos \sqrt{1-y} \right).$$

Vi definerer funksjonene

$$f(y) = \sqrt{y(1-y)}, \qquad g(y) = \arccos \sqrt{1-y}.$$

Hvis vi prøver å Taylorutvikle $f(y)$ om $y = 0$ finner vi at $f(0) = 0$

$$f'(y) = \frac{1 - y/2}{2\sqrt{y(1-y)}}, \qquad f'(0) = \infty.$$

$f(y)$ har altså uendelig derivert i $y = 0$, og har dermed ikke noen Taylorrekke i det punktet. Det samme finner vi for $g(y)$ når vi bruker

$$\frac{d}{dx} \arccos x = -\frac{1}{\sqrt{1-x^2}}.$$

f) Finn ut hvordan du allikevel kan vise at det tilnærmede uttrykket i d) kommer fram av det eksakte uttrykket i c) i grensa $y \ll 1$ og finn laveste ordens korreksjon til uttrykket i d).

Løsning: Vi definerer

$$\bar{f}(y) = \frac{f(y)}{\sqrt{y}}, \qquad \bar{g}(y) = \frac{g(y)}{\sqrt{y}}$$

og finner Taylorrekkene

$$\bar{f}(y) = 1 - \frac{1}{2} y + \cdots$$
$$\bar{g}(y) = 1 + \frac{1}{6} y + \cdots.$$

Dermed er

$$\frac{T}{\sqrt{y}} = \sqrt{\frac{R_0^3}{2GM}}(2 - \frac{1}{3}y + \cdots)$$

og da blir

$$T = \sqrt{\frac{2h}{g_0}}\left(1 - \frac{1}{6}\frac{h}{R_0} + \cdots\right).$$

NukTek

Ekstraoppgave spesielt tilpasset NukTek-studenter. Men også fin for andre

G 8.5 Radioaktivt henfall

^{18}F er en av de mest brukte radioaktive nuklidene i medisin. Den kan brukes i positron emisjons tomografi, også kjent som PET-scan. Dette er en diagnostisk undersøkelse som brukes til å ta bilder av kroppen, og kan blant annet brukes til å finne kreftsvulster.

^{18}F henfaller ved å sende ut positroner og har en halveringstid $\mathrm{T}_{1/2} = 109.77$ min. Det betyr at etter 109.77 min ble halvparten av 18F dannet om tilbake til 18O. Relasjonen mellom halveringstid $\mathrm{T}_{1/2}$ og henfallskonstanten λ er gitt ved

$$\lambda = \frac{\ln 2}{\mathrm{T}_{1/2}}. \tag{8}$$

Henfallskonstanten gir raten som henfallsprosesser per sekund.

a) Regn ut henfallskonstanen til ^{18}F.

^{18}F produseres ved å akselerere protoner i en partikkelakselerator som deretter blir skutt på ^{18}O, som deretter sender ut et nøytron.

$$^{18}\mathrm{O} + p \rightarrow ^{18}\mathrm{F} + n$$

Siden ^{18}F selv er radioaktivt, vil en del av den produserte mengden ^{18}F henfalle mens bestrålingen med protoner foregår. Antallet ^{18}F nuklider som finnes, $N(t)$, som funksjon av bestrålingstiden, t, er derfor gitt ved

$$\frac{dN(t)}{dt} = R_p - \lambda N(t), \tag{9}$$

der R_p er produksjonsraten og λ er henfallskonstanten til ^{18}F. Anta at det produseres $44 \cdot 10^{10}$ ^{18}F nuklider per sekund, altså $R_p = 44 \cdot 10^{10}\mathrm{s}^{-1}$.

b) Løs differensialikningen over numerisk, slik at du beregner hvor mange ^{18}F nuklider som finnes som funksjon av bestrålingstid. Plot for rundt 20 timers bestrålingstid. Husk at vi starter med null produserte ^{18}F kjerner.

c) Hva er optimal bestrålingstid for å produsere mest mulig ^{18}F?

d) Løs differensiallikningen gitt i likning 9 analytisk.

Løsning:

a) Vi bruker likning 8, og setter inn $T_{1/2} = 109.77$ min:

$$\lambda = \frac{\ln 2}{109.77 \cdot 60\text{s}} = 1.052 \cdot 10^{-4}\text{s}^{-1} \tag{10}$$

b)

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


#Defining constants
Thalf = 109.77*60  # [s]
Rp = 44*10**10  # [s^-1]
N0 = 10**6


# a)
lamb = np.log(2)/Thalf  # [s^-1]

print(f"The decay constant for 18-F: lambda = {lamb} s^-1")



# b)
# Time interval
T = 20.0*3600  # [seconds]
N_t = 10000
dt = T/N_t


# Vector for storing number of 18F nuclides
N = np.zeros(N_t,float)
t = np.zeros(N_t, float)


# Loop to solve equation

for i in range(N_t-1):
    t[i+1] = t[i] + dt

```

```
35      dN = (Rp - lamb*N[i])*dt
36      N[i+1] = N[i] + dN
37
38  # Making sure we have hours on the x-axis
39  t = t/3600
40
41  plt.plot(t, N, label="N(t)")
42  plt.xlabel("Time [hours]")
43  plt.ylabel("Number of 18F nuclides")
44  plt.legend()
45  plt.show()
46
47  """
48  Terminal>> python3 uke8.py
49  The decay constant for 18-F: lambda = 0.00010524235227596267 s^-1
50  """
```

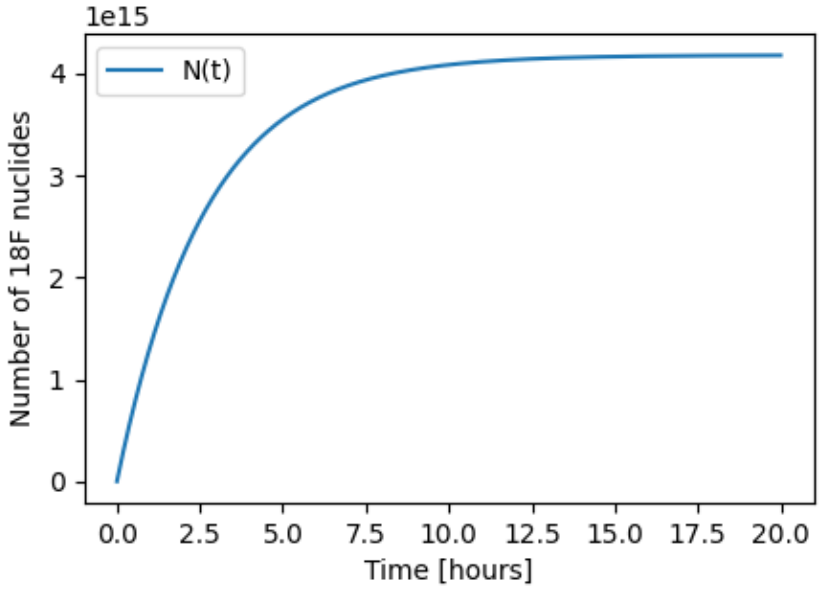


Figure 1: Antall ^{18}F nuklider som funksjon av bestrålingstid.

c) Fra figur 1, kan vi se at antallet ^{18}F nuklider når et platå rundt 12.5 timers bestrålingstid. Dette er fordi produksjonen har nådd et likevektspunkt der antallet nye ^{18}F-kjerner skapt per tid er lik antallet ^{18}F-kjerner som henfaller per tid. Det vil derfor ikke være lønnsomt å bestråle lenger enn denne tiden. I virkeligheten vil en enda kortere bestrålingstid bli brukt ettersom det er svært dyrt å drive en partikkelakselearator. Når man nærmer seg likevekt vil man få produsert få ^{18}F-kjerner i forhold til hvor mange som henfaller. Det vil derfor være mer lønnsomt å stoppe bestrålingen før man når likevekt.

d)

Vil løse likning 9 analytisk. Bruker metoden med separasjon av variabler og integrerer.

$$\frac{dN(t)}{dt} = R_p - \lambda N(t) \tag{11}$$

$$\int \frac{1}{R_p - \lambda N(t)} dN(t) = \int dt \tag{12}$$

Integrasjon ved substitusjon gir

$$-\frac{1}{\lambda}\int \frac{1}{x}dx = \int dt, \qquad x = R_p - \lambda N(t) \tag{13}$$

$$-\frac{1}{\lambda}\ln|x| = t + c. \tag{14}$$

Setter inn for x og løser med hensyn på $N(t)$.

$$\ln|R_p - \lambda N(t)| = -\lambda t - \lambda c \tag{15}$$

$$R_p - \lambda N(t) = e^{-\lambda t - \lambda c} \tag{16}$$

$$N(t) = \frac{1}{\lambda}\left(R_p - Ce^{-\lambda t}\right) \tag{17}$$