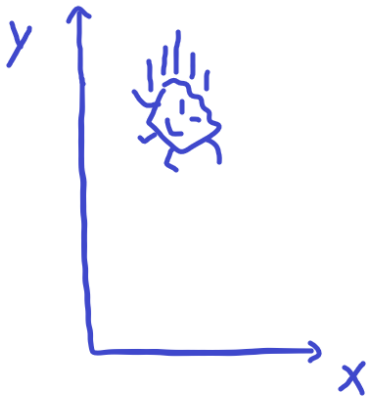
Seminaroppgaver 8

S 8.1 Fallende muffinsform



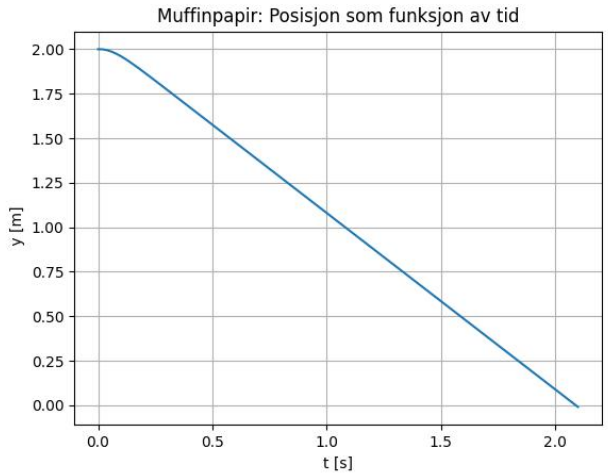
I følgende oppgave skal vi se på posisjonen til en muffinsform av papir som blir sluppet fra 2 meters høyde. Den blir påvirket av både tyngdekraft og luftmotstand.

a) Les inn posisjon- og tidsdatapunktene fra filen y_axis_muffinpaper.txt, og plott posisjonen som en funksjon av tid. Beskriv grafen.

Løsning:

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4
5  y = []
6  t = []
7  with open("y_axis_muffinpaper.txt", "r") as infile:
8      infile.readline()
9      for line in infile:
10         list = line.split(sep=", ")
11         y.append(float(list[0]))
12         t.append(float(list[1]))
13
14 y = np.array(y)
15 t = np.array(t)
16
17 if __name__ == "__main__":
18
19     plt.plot(t,y)
```

```
20      plt.grid()
21      plt.xlabel("t [s]")
22      plt.ylabel(" y [m]")
23      plt.title("Muffinpapir: Posisjon som funksjon av tid")
24      plt.savefig("S2a.jpg")
25      plt.show()
```



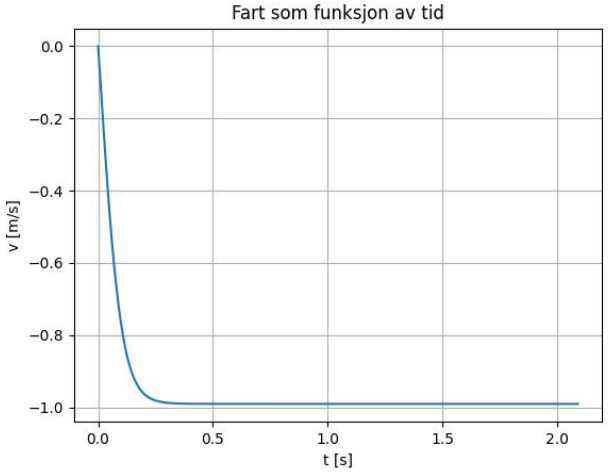
Som vi ser så er posisjonen lineær mesteparten av tiden etter $t \approx 0.1$ s. Den beveger seg fra 2 m til 0 m, noe som er forventet gitt at det er det totale intervallet som muffinsformen tilbakelegger.

b) Finn numerisk farten som en funksjon av tid, og plott den. Hva ser vi at skjer med farten ettersom tiden går? Hvorfor får farten denne formen? Forklar. Prøv videre fra farten å anslå kvalitativt hvordan akselerasjonen vil se ut.

Løsning:

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  from S2a import t,y
4
5
6  def num_dev(y,t):
7      dydt = []
8      for i in range(len(y)-1):
9          dt = t[i+1]-t[i]
10         deriv = (y[i+1]-y[i])/dt
11         dydt.append(deriv)
12     return np.array(dydt),t[0:len(dydt)]
13
14 vy,tv = num_dev(y,t)
15
```

```
16 if __name__=="__main__":
17
18     plt.plot(tv,vy)
19     plt.grid()
20     plt.ylabel("v [m/s]")
21     plt.xlabel("t [s]")
22     plt.title("Fart som funksjon av tid")
23     plt.savefig("S2b.jpg")
24     plt.show()
```



Ettersom tiden går så blir hastigheten større og konvergerer mot $|v| = 1\text{m/s}$. Denne farten kalles terminalhastighet og den oppnås fordi luftmotstanden, (som er proporsjonal med farten) og gravitasjonskraften blir like store, men motsatt rettet. Akselerasjonen blir derfor 0.

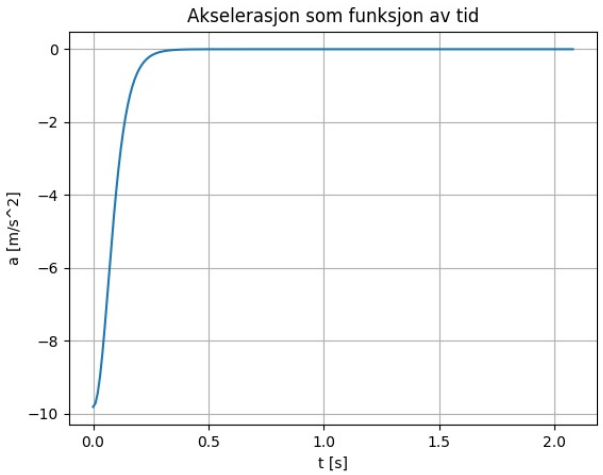
For å beskrive akselerasjonen kvalitativt så kan vi anta at før vi når terminalhastigheten så vil akselerasjonen være mindre enn null. Akselerasjonen vil deretter konvergere mot 0 ettersom tiden går. Dette ser vi fordi hastigheten blir konstant.

c) Finn numerisk akselerasjonen som en funksjon av tid og plott den. Stemmer den med den kvalitative beskrivelsen dere ga i forrige oppgave?

Løsning:

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from S2b import num_dev,vy,tv
4
5
6 ay,ta = num_dev(vy,tv)
7
8 plt.plot(ta,ay)
```

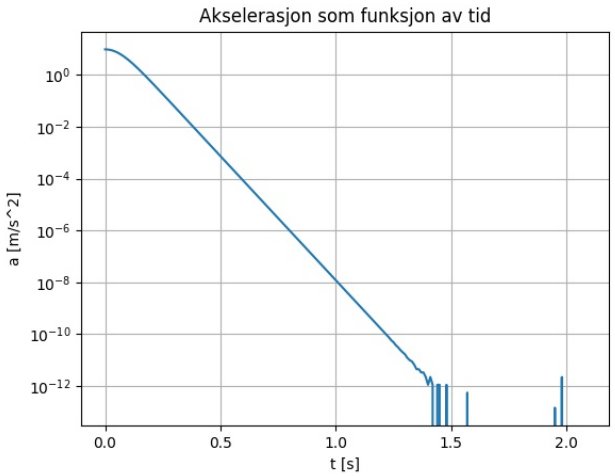
```
plt.grid()
plt.ylabel("a [m/s^2]")
plt.xlabel("t [s]")
plt.title("Akselerasjon som funksjon av tid")
plt.savefig("S2c.jpg")
plt.show()
```



Dette stemmer greit med den kvalitative beskrivelsen av akselerasjon som vi hadde. Akselerasjonen startet med å være mindre enn null, som vi antok og gikk mot null ettersom tiden ble større.

d) Ved hvilken tid blir luftmotstanden like stor som gravitasjonskraften? Bruk den numeriske akselerasjonsgrafen for å finne den omtrentlige tiden når dette skjer. Kan man se dette kun fra akselerasjonsgrafen eller kunne man ha funnet ut dette ved å se på de andre figurene?

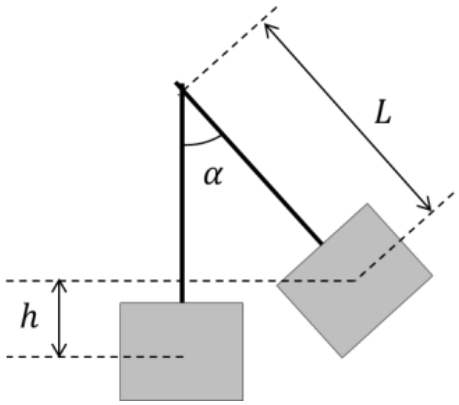
Løsning: Fra grafen kan det se ut som akselerasjonen blir null ved omlag $t \approx 0.25$ s, som vil si at det er her luftmotstanden og tyngdekraften blir like store. Men dette er bare fordi vi ikke klarer å se at akselerasjonen er forskjellig fra null med den oppløsningen vi har i plottet. teoretisk vil akselerasjonen nærme seg null, men aldri bli helt null. En måte som ofte er nyttig for å se små størrelser i samme plott som store er å plotte logaritmen til verdien (bruke logaritmiske akser). Siden akselerasjonen er negativ tar vi absoluttverdien.



Som vi ser skjer det ikke noe spesielt vet t=0,25 s, og akselerasjonen fortsetter å synke ned til verdien er så liten at numerisk feil av en eller annen type forstyrrer. Hvis vi hadde hatt reelle målte verdier for posisjonen som utgangspunkt ville de hatt en usikkerhet, og akselerasjonen ville på et tidspunkt blitt mindre enn vi kunne skille fra null med det eksperimentelle oppsettet.

S 8.2 En kule skytes i en trekubbe

En trekubbe på 5,0 kg henger i en 1,0 m lang snor som har neglisjerbar masse. Du skyter en pistolkule på 10 g horisontalt inn i kubben med hastighet $v = 400$ m/s. Kulen stanser og kubben svinger opp. Vi ser bort fra luftmotstand.



a) Hvor mye kinetisk energi har kulen før kollisjonen?

Løsning: Hvis kulas masse er m er energien

$$K = \frac{1}{2}mv^2 = 800 \text{ J} \approx 0,80 \text{ kJ}.$$

b) Hvor mye kinetisk energi har kubben etter kollisjonen?

Løsning: Kollisjonen er fullstendig uelastisk, så vi kan ikke bruke bevaring av mekanisk energi. Siden det er ingen ytre krefter i horisontal retning er drivet bevart under kollisjonen, $mv = (m+M)V$, hvor M og V er henholdsvis massen og hastighet til trekubben. Vi finner farta

$$V = \frac{m}{M+m}v = 0,80 \text{ m/s}.$$

og den kinetiske energien

$$K_1 = \frac{1}{2}(m+M)V^2 = 1,6 \text{ J}.$$

c) Til hvilken vinkel svinger kubben opp?

Løsning: Etter kollisjonen svinger kubben opp. Hvis vi ser bort fra luftmotstanden er energi bevart siden gravitasjon er en konservativ kraft, og snordraget står vinkelrett på bevegelsesretningen og gjør ikke noe arbeid. Energibevaring gir $K_1 = (m+M)gh$, og dermed er høyden

$$h = \frac{K_1}{(m+M)g} = \frac{V^2}{2g} = 0,033\text{m}.$$

Vi har at $h = L(1-\cos\alpha)$ slik at vinkelen blir

$$\alpha = \arccos(1-h/L)) = 15^\circ.$$

S 8.3 En pendel med stort utslag

Bevegelsesligningen til vinkelen θ mellom snora til en pendel og vertikalen er gitt ved

$$\frac{d^2\theta}{dt^2} + \frac{g}{l}\sin(\theta) = 0, \tag{1}$$

der g er tyngdeakselerasjonen og l er pendelsnorens lengde. Dersom pendelen slippes fra en liten vinkel θ_0, kan man bruke Taylor-rekkeutvikling til å tilnærme $\sin(\theta)$ som θ, og dermed forenkle ligningen til

$$\frac{d^2\theta}{dt^2} + \frac{g}{l}\theta = 0. \tag{2}$$

Dersom pendelen slippes fra ro, blir løsningen på denne differensialligningen $\theta(t) = \theta_0\cos\left(\sqrt{\frac{g}{l}}t\right)$.

a) Skriv et program som beregner vinkelen som funksjon av tid uten å bruke Taylor-rekkeutviklingen, for en gitt initialhastighet og startvinkel. Med andre ord, skriv et program som løser likning (1) numerisk. Bruk Euler-Cromer-metoden.

Løsning: Se løsningen til neste deloppgave.

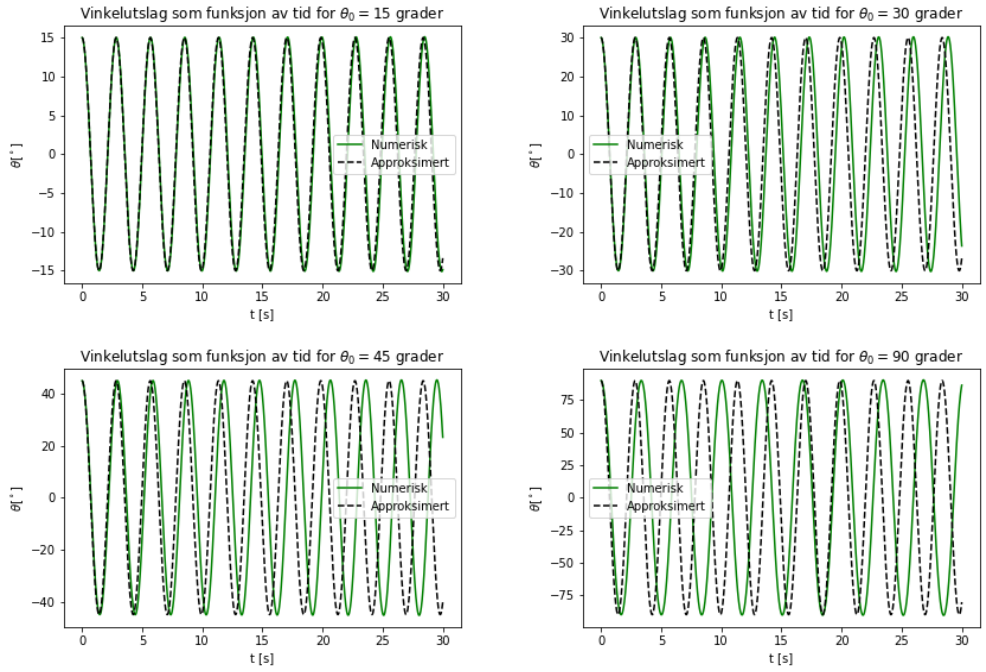
I de neste deloppgavene skal vi se på forskjellen på den numeriske løsningen og den analytiske løsningen vi får når vi bruker Taylor-rekkeutviklingen. Bruk $l = 2.0$ m, og anta at pendelen slippes fra ro. Vi lar koden beregne bevegelsen for en tid fra 0 s til 30 s.

b) Lag fire plott der dere sammenlikner den numeriske løsninga av likning (1) med den analytiske løsninga av likning (2). Bruk $\theta_0 \in \{15°, 30°, 45°, 90°\}$ og se på de første 30 s av bevegelsen. Diskuter hvor god den tilnærmede løsningen er for de forskjellige utslagsvinklene. Svarer resultatene til forventningene deres?

Løsning:

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  # Deloppgave b
5
6  g = 9.81
7  l = 2 #[m]
8  thetadot0 = 0
9
10 def theta_approx(t, theta0):
11     return theta0 * np.cos(np.sqrt(g / l) * t)
12
13 T = 30 #[s]
14 dt = 0.01
15 N = int(T / dt) # Antall tidssteg
16
17 t = np.zeros(N, float)
18 theta = np.zeros(N, float)
19 thetadot = np.zeros(N, float)
20 thetadot[0] = thetadot0
21
22 for theta0 in [15, 30, 45, 90]:
23     theta[0] = theta0 * np.pi / 180
24     for i in range(N - 1):
25         thetadotdot = - g * np.sin(theta[i]) / l
26         thetadot[i + 1] = thetadot[i] + thetadotdot * dt
27         theta[i + 1] = theta[i] + thetadot[i + 1] * dt
28         t[i + 1] = t[i] + dt
29
30     plt.plot(t, theta * 180 / np.pi, label = 'Numerisk', color = '↩
       green')
```

```
31      plt.plot(t, theta_approx(t, theta0), label = 'Approksimert',
32               color = 'black', linestyle = 'dashed')
33      plt.xlabel('t [s]')
34      plt.ylabel(r'$\theta[^\circ]$')
35      plt.legend()
36      title = r'Vinkelutslag som funksjon av tid for $\theta_0=$'
37      plt.title(title + f'{theta0} grader')
38  #     plt.savefig(f'theta_{theta0}.png')
39      plt.show()
```

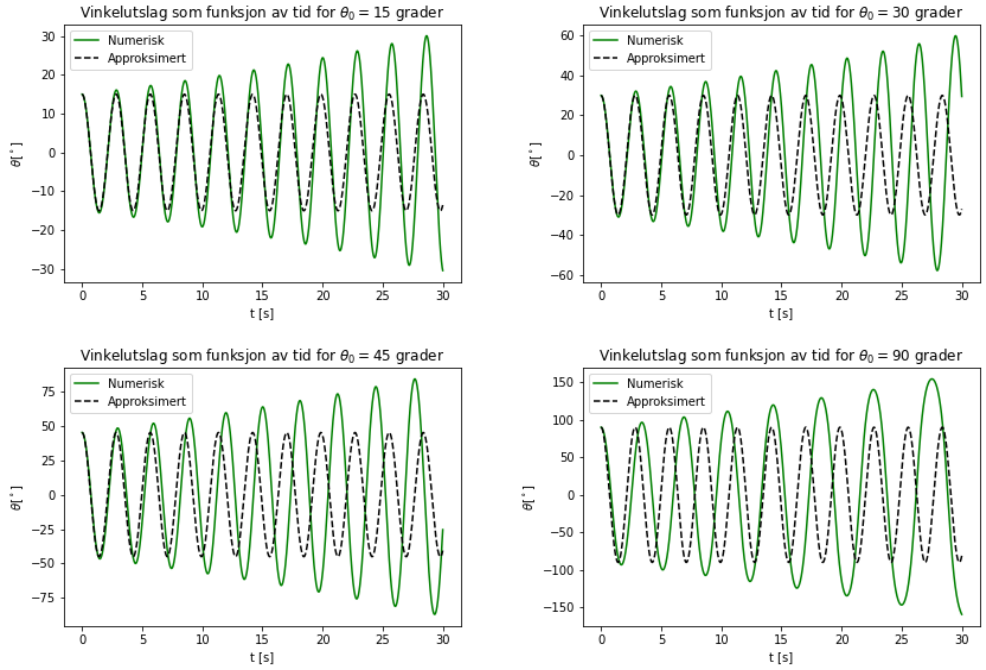


For $\theta_0 = 15°$ kan vi se at den numeriske løsningen følger tilnærmingen svært godt for hele tidsintervallet. Taylor-tilnærmingen baserer seg jo på at man har en svingning rundt en liten utslagsvinkel, så da gir det mening at løsningene stemmer overens. Jo større θ_0 blir, jo dårligere blir overensstemmelsen. Med $\theta_0 = 30°$ er det greit samsvar for omtrent 10 s, men så blir den tilnærmede løsningen gradvis dårligere. For $\theta_0 = 45°$ tar det bare rundt 5 s før løsningene blir vesentlig forskjellige, og for $\theta_0 = 90°$ overlapper de to løsningene såvidt en halv svingning før det går veldig galt. Vi kan derfor se at det må, som forventet, være små vinkler for at man skal kunne bruke den tilnærmede løsningen uten å få store feil i resultatene.

c) I forrige deloppgave er det (som alltid med numeriske løsninger!) viktig å finne en passende steglengde. Prøv forskjellige steglengder og kommenter hva som skjer hvis de er for store. Prøv her både Euler-Cromer-metoden som tidligere, og sammenlign dette med Euler-metoden.

Løsning: I løsningen til forrige deloppgave ble det brukt en steglengde på $dt = 0.0001$ s. Figurene under har blitt laget med $dt = 0.01$ s for å gjøre det ekstra tydelig hvor viktig steglengde er, men

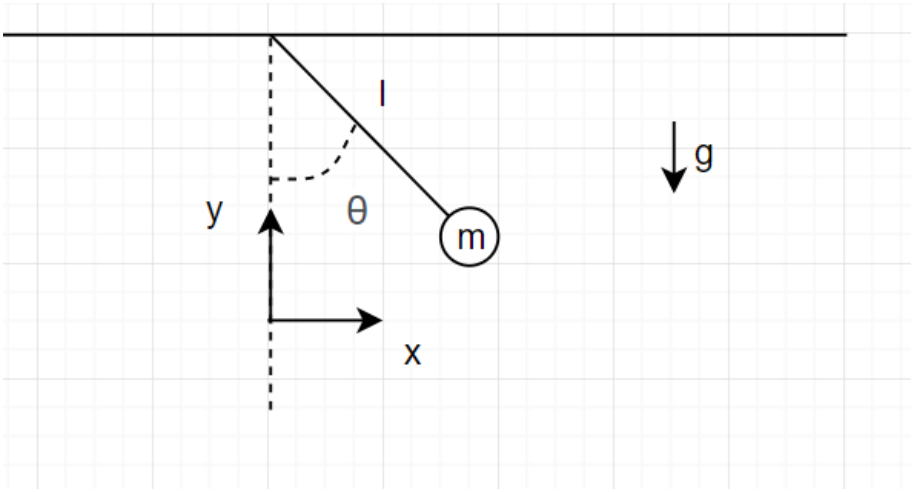
man vil få en tilsvarende effekt (bare ikke så stor) med $dt = 0.001$ s også. Figurene viser resultatet med Euler-algorimen.



Vi kan se at for stor steglengde får det numeriske systemet til å tilsynelatende svinge med større og større utslag, som tilsvarer økt energi i systemet. Siden det kun er tyngdekraft som virker, skal total mekanisk energi være bevart. Gal steglengde gjør derfor systemet ufysikalsk. Med Euler-Cromer-metoden får vi ikke dette problemet før ved ekstremt store steglengder.

S 8.4 Pendel med luftmotstand

I denne oppgaven skal vi undersøke en pendels bevegelse med luftmotstand numerisk, finne farten i bunnpunktet og sammenlikne denne med hva farten ville vært uten luftmotstand. Anta at pendelen er festet i taket med en masseløs, stram snor med lengde $l = 1.0$ m. Vi plasserer koordinatsystemet vårt slik at når pendelen henger helt stille i bunnpunktet, befinner den seg i origo, se figur. Pendelen har en masse på $m = 3.0$ kg, og før den slippes henger den med en vinkel på $\theta_0 = \frac{\pi}{4}$ i forhold til y-aksen.



a) Finn pendelens x- og y-posisjon som funksjon av vinkelen θ. Husk at $y(\theta = 0) = 0$ m.

Løsning: Siden vinkelen er utspent fra y-aksen, kan man bruke trigonometri for å relatere vinkelen til posisjonen. Vi får da at

$$x(\theta) = l\sin(\theta), \quad y(\theta) = l - l\cos(\theta)$$

og vi har at $y(\theta = 0) = 0$ m.

b) Bruk bevaring av mekanisk energi for å finne hastigheten til pendelen når den er i bunnpunktet i banen, dersom vi slipper den fra ro og ser bort fra luftmotstand.

Løsning: Vi begynner med å finne $y(\theta_0)$.

$$y(\theta_0) = l\left(1 - \frac{\sqrt{2}}{2}\right)$$

Siden pendelen slippes fra ro, har den kun potensiell energi gitt ved $U(y) = mgy$. Fra dette får vi at den totale mekaniske energien i det vi slipper pendelen er

$$U(y(\theta_0)) = U_0 = mgl\left(1 - \frac{\sqrt{2}}{2}\right)$$

I bunnen av banen har pendelen kun kinetisk energi. Bevaring av mekanisk energi gir da

$$mgl\left(1 - \frac{\sqrt{2}}{2}\right) = \frac{1}{2}mv^2$$

$$\Rightarrow v^2 = 2gl\left(1 - \frac{\sqrt{2}}{2}\right)$$

$$\Rightarrow v = \pm\sqrt{2gl\left(1-\frac{\sqrt{2}}{2}\right)}$$

Setter vi inn verdiene får vi at

$$v \approx \pm 2.4 \text{ m/s}$$

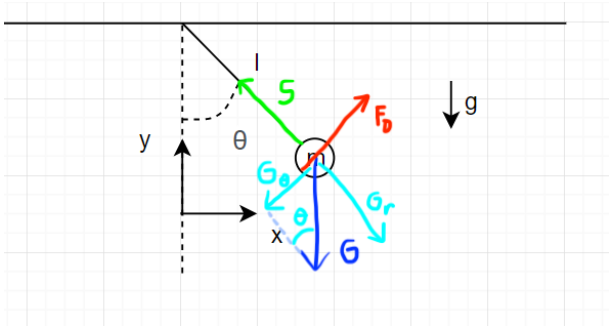
Siden pendelen beveger seg mot venstre i negativ x-retning i bunnpunktet (hastigheten har alltid retning tangentielt til banen), har vi

$$\vec{v} = -2.4 \text{ m/s} \cdot \hat{\mathbf{i}}$$

i bunnpunktet når den beveger seg mot venstre.

c) Tegn et frilegemediagram av pendelen litt etter at den slippes. Inkluder luftmotstand.

Løsning: Siden pendelen beveger seg og har fått en hastighet, vil luftmotstanden virke i motsatt retning som hastigheten. Kreftene som virker på pendelen er snordraget $\vec{S}$, tyngdekraft $\vec{G}$ og luftmotstand $\vec{F}_D$.



d) Skriv opp summen av kreftene som virker i tangentiell retning på pendelen. Finn også akselerasjonen i denne retningen ved Newtons 2. lov.

Løsning: Summen av kreftene i tangentiell retning er gitt ved

$$\sum F_t = -G_t + F_D$$

$$\Rightarrow ma_t = -G_t + F_D$$

$$\Rightarrow a_t = \frac{-G_t + F_D}{m}$$

e) Vis at bevegelsesligningen til pendelen er

$$\frac{d^2\theta}{dt^2} = -\frac{g}{l}\sin(\theta) - \frac{b}{ml}v$$

når luftmotstandskraften er gitt ved

$$\vec{F}_D = -b\vec{v}$$

Her er b en konstant, vi bruker $b = 0.50$ kg/s.

Løsning: Dekomponeringen av G i tangentiell retning er gitt ved $G_t = G\sin(\theta)$. Newtons andre lov gir at

$$a_t = -g\sin(\theta) - \frac{b}{m}v$$

Videre bruker vi at for sirkelbevegelser så kan vi uttrykke buelengden s som

$$s = l\theta$$

Vi dobbeltderiverer med hensyn på tid og får

$$a_t = l\frac{d^2\theta}{dt^2}$$

Setter vi dette inn og deler på l så får vi

$$\frac{d^2\theta}{dt^2} = -\frac{g}{l}\sin(\theta) - \frac{b}{ml}v$$

som er uttrykket vi skulle finne.

f) Bruk Euler-Cromer-metoden for å finne farten i banens bunnpunkt numerisk ved å bruke akselerasjonen som inkluderer luftmotstand. Kommenter svaret. *Hint: Vi kan bruke buelengde til å skrive at* $v_t = l \cdot \frac{d\theta}{dt}$.

Løsning:

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  l = 1 #m
5  m = 3 #kg
6  g = 9.81 # m/s**2
7  b = 0.5 #kg/s
8
9  def ddtheta(theta, v):
10     term1 = -g/l *np.sin(theta) # Gravitasjonsterm
11     term2 = -b*v/(l*m) # Luftmotstand
12     return term1 + term2
13
14 print(6*np.pi*eta*r)
15
16 theta0 = np.pi/4
17 dtheta = [0]
18 theta = [theta0]
```

```
x = [l*np.sin(theta0)]
y = [l-l*np.cos(theta0)]
t = [0]
dt = 0.001
i = 0
while x[-1]>=0:

    # Oppdaterer tid
    t.append(t[i]+dt)

    #Oppdaterer thetadotdot
    thetadotdot = ddtheta(theta[i],l*dtheta[-1])
    #Finner det første steget i Euler-Cromer
    dtheta.append(dtheta[i]+thetadotdot *dt)
    #Finner det andre steget i Euler_Cromer
    theta.append(theta[i] + dtheta[i+1]*dt)
    #Appender til x og y
    x.append(l*np.sin(theta[i]))
    y.append(l-l*np.cos(theta[i]))
    i+=1



print(l*dtheta[-1])
```

Fra terminalen fikk vi at

$$v_{num} = -2.30 \text{ m/s} \approx -2.3 \text{ m/s}.$$

Vi ser at farta vi finner her er mindre enn den vi fant da vi så bort fra luftmotstand. Forskjellen er ikke så stor, $\delta v \approx -0.1$ m/s, den bremses bare litt for hver periode.

g) Er forskjellen i fart med og uten luftmotstand større eller mindre dersom man begynner i $\theta_0 = \pi/2$? Forklar resultatet du finner.

Løsning:

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

l = 1 #m
m = 3 #kg
g = 9.81 # m/s**2
b = 0.5 #kg/s

def ddtheta(theta, v):
    term1 = -g/l *np.sin(theta) # Gravitasjonsterm
    term2 = -b*v/(l*m) # Drag
    return term1 + term2


```

```
15
16 theta0 = np.pi/2
17 dtheta = [0]
18 theta = [theta0]
19 x = [l*np.sin(theta0)]
20 y = [l-l*np.cos(theta0)]
21 t = [0]
22 dt = 0.001
23 i = 0
24 while x[-1]>=0:
25
26     # Oppdaterer tid
27     t.append(t[i]+dt)
28
29     #Oppdaterer thetadotdot
30     thetadotdot = ddtheta(theta[i],l*dtheta[-1])
31     #Finner det første steget i Euler-Cromer
32     dtheta.append(dtheta[i]+thetadotdot *dt)
33     #Finner det andre steget i Euler_Cromer
34     theta.append(theta[i] + dtheta[i+1]*dt)
35     #Appender til x og y
36     x.append(l*np.sin(theta[i]))
37     y.append(l-l*np.cos(theta[i]))
38     i+=1
39
40 print(l*dtheta[-1])
```

Fra terminalen leser vi av at $v_{num} \approx -4.23$ m/s. Med analytisk løsning uten luftmotstand får vi at $v \approx -4.43$ m/s. Dette gir oss en forskjell mellom analytisk og numerisk på $\delta v \approx -0.2$ m/s. Forskjellen blir større i dette tilfellet fordi luftmotstanden har hatt en lengre strekning å virke på, og fordi farta er større slik at luftmotstanden blir større.

h) Finn numerisk arbeidet luftmotstanden har gjort under bevegelsen fra starten til bunnpunktet. Finn fra dette den mekaniske energien i bunnpunktet og dermed farta. Stemmer denne med det du fant ved den direkte numeriske beregningen?

Løsning:

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  l = 1 #m
5  m = 3 #kg
6  g = 9.81 # m/s**2
7  b = 0.5 #kg/s
8
9  def ddtheta(theta, v):
10     term1 = -g/l *np.sin(theta) # Gravitasjonsterm
11     term2 = -b*v/(l*m) # Drag
12     return term1 + term2
```

```
13
14
15
16 theta0 = np.pi/2
17 U0  = m*g*l*(1-np.cos(theta0))
18 dtheta = [0]
19 theta = [theta0]
20 x = [l*np.sin(theta0)]
21 y = [l-l*np.cos(theta0)]
22 t = [0]
23 dt = 0.001
24 i = 0
25 w = 0
26 while x[-1]>=0:
27
28     # Oppdaterer tid
29     t.append(t[i]+dt)
30
31     #Oppdaterer thetadotdot
32     thetadotdot = ddtheta(theta[i],l*dtheta[-1])
33     #Finner det første steget i Euler-Cromer
34     dtheta.append(dtheta[i]+thetadotdot *dt)
35     #Finner det andre steget i Euler_Cromer
36     theta.append(theta[i] + dtheta[i+1]*dt)
37     #Appender til x og y
38     x.append(l*np.sin(theta[i]))
39     y.append(l-l*np.cos(theta[i]))
40     #Beregner arbeidet ved numerisk integrasjon med trapesmetoden
41     w = w + b*l*l*0.5*(dtheta[i]*dtheta[i]+dtheta[i+1]*dtheta[i↩
           +1])*dt
42     i+=1
43
44 print(f"Farta i bunnpunktet fra numerisk beregning er {l*dtheta↩
       [-1]} m/s")
45 print(f"Arbeidet gjort av lufmotstanden er {w} J")
46 print(f"Farta i bunnpunktet fra arbeid-energiteoremet er {np.sqrt↩
       (2*(U0-w)/m)} m/s")
```

Arbeidet luftmotstanden gjør er gitt ved integralet

$$W = \int_0^s F_D \cdot ds' = \int_0^t bv^2 dt' = bl^2 \int_0^t \omega^2 dt'.$$

Her har vi brukt at $F_D = bv$, $ds = vdt$, og $v = l\frac{d\theta}{dt} = l\omega$ hvor ω er vinkelfarten. Vi beregner arbeidet numerisk ved trapesmetoden:

$$W \approx \frac{1}{2} bl^2 \sum_i (\omega_i^2 + \omega_{i+1}^2).$$

Legg merke til at dette uttrykket er implementert i koden slik det er skrevet. Dette er for å gjøre det lett å lese, men er ikke effektivt siden vi må legge til samme størrelsen to ganger (unntatt endepunktene). For å effektivisere beregningen burde vi endre dette (evt. bruke funksjoner i Numpy for trapesmetoden).

Vi bruker arbeid-energiteoremet: $K_1 = U_0 - W$ hvor K_1 er kinetisk energi i bunnpunktet, U_0 potensiell energi i startpunktet, og luftmotstanden gjør et negativt arbeid W på pendelen. Dette gir farta i bunnpunktet:

$$v_1 = \sqrt{\frac{2K_1}{m}} = \sqrt{\frac{2(U_0 - W)}{m}}.$$