o4

February 16, 2024

1 Øving 4

Aslak Øvereng Iveland, Sevat Lappegard Skurtveit, Olav A. Samuelsen Telneset

1.1 Oppgave 1

Kula i et flipperspill skytes ut fra en fjærbelastet avtrekker. Spilleren trekker i fjæra slik at den presses sammen en avstand $x = 7,0$ cm. Massen til kula er 80 g, og flipperspillet har en helningsvinkel $\theta = 15°$. Kula sklir friksjonsfritt mot underlaget. Se figuren under.

Hvor stor må fjærkonstanten k til fjæra være dersom kula akkurat skal nå toppen av flipperspillet, som ligger en avstand $s = 78$ cm fra punktet der kula forlater fjæra (i punktet der fjæra er slapp)?

Svar: Bruker arbeid-energi-setningen:

$W = \Delta U = \Delta K$

som gir:

$\frac{1}{2}v_1^2 + \frac{1}{2}mgh_1 + \frac{1}{2}kx_1^2 = \frac{1}{2}v_0^2 + \frac{1}{2}mgh_0 + \frac{1}{2}kx_0^2$

som videre gir:

$\frac{1}{2}mgh_1 = \frac{1}{2}v_0^2 + \frac{1}{2}kx_0^2$

deretter får vi uttrykket:

$k = \frac{2mgh}{x^2} \implies \frac{2mg \cdot \sin\theta \cdot s}{x^2}$

$\frac{2 \cdot (0.08kg) \cdot (9.81m/s^2) \sin 15° \cdot 0.78}{(0.07)^2}$

$= \underline{\underline{64.68N/m}}$

1.2 Oppgave 2

En vogn i en berg-og-dalbane starter i en viss høyde h over det laveste punktet A i en sirkulær loop med radius r. To andre punkter i loopen er markerte: B er midtveis oppe, og C er det høyeste punktet. Se figuren under.

I denne oppgaven skal vi se bort fra friksjon og luftmotstand.

a) Fra hvilken høyde h over punkt A må vogna slippes for at den skal kunne fullføre en hel loop uten å miste kontakten med underlaget?

I de to neste oppgavene slippes vogna fra en starthøyde $h = 3r$. Bestem normalkrafta på vogna, uttrykt ved vognas tyngde G, i

b) Punkt C (toppen)

c) Punkt B (midtveis til toppen)

Oppgave 2

a)

Energi før: mgh

hc = 2r

Energi i C = mg2r + 1/2mv^2

sentripetalakselerasjonene i C må være større enn g:

v^2/r >= g

==> v^2 = gr

mgh = mg2r + 1/2mgr

h = 2.5r

b)

h = 3r

mg3r = mg2r + 1/2mv^2

==> v^2 = 2gr

G + N = mv^2/r

G + N = m2gr/r

==> N = G

c)

mg3r = mgr + 1/2mv^2

==> v^2 = 4gr

N = mv^2/r

N = m4gr/r

==> N = 4G

1.3 Oppgave 3

To curlingsteiner A og B med identisk masse $m = 19$ kg kolliderer i et rett, sentralt støt. Før støtet har stein A fart $v_{\mathrm{A}} = 3,0$ m/s mot høyre og stein B fart $v_{\mathrm{B}} = 5,0$ m/s mot venstre, og etter støtet har stein A fart $u_{\mathrm{A}} = 4,5$ m/s mot venstre. Se figuren under.

a) Finn farten til stein B etter støtet.

b) Var støtet elastisk?

1.3.1 a

positiv retning mot høyre $P_f = P_e \rightarrow V_a m + V_b m = u_a m + u_b m \rightarrow u_b = \frac{V_a m + V_b m - u_a m}{m} = 3,5 m/s$

1.3.2 b

vi ser fra oppgave a at den samlede farten før og etter er lik. når massen for alle klossene er den samme, og bevart, vil dette bety at den kinetiske energien er bevart, som konstituerer ett elastisk støt

1.4 Oppgave 4

To fallskjermhoppere A og B med identiske masser $m = 70$ kg faller vertikalt fra samme starthøyde med null startfart. A faller med hodet først og har frontareal $A = 0,17$ m^2 og "drag-koeffisient" $C_d = 0,70$; B faller liggende og har $A = 1,0$ m^2 og $C_d = 1,0$. Vi forutsetter kvadratisk luftmotstand $F_D = \frac{1}{2}\rho A C_d v^2$ med $\rho = 1,29$ kg/m^3, og tyngdeakselerasjonen g kan antas konstant over fallhøyden.

a) Finn ved regning terminalhastighetene for A og B.

I de to siste oppgavene skal vi bruke Python til å løse bevegelseslikningene numerisk.

b) Gitt følgende Python-funksjon som beregner luftmotstanden (drag) som funksjon av frontareal A, drag-koeffisient C_d og fart v, for en gitt (konstant) verdi av luftas massetetthet ρ:

```
def drag(A,C,v):
    """Input:
    A: Frontareal [m^2]
    C: Drag-koeffisient []
    v: Fart [m/s]
    """
    rho=1.29
    k=0.5*rho*A*C
    ?
```

Hvilken Python-kode skal stå i linja markert med ? for at funksjonen skal gi luftmotstanden i newton som funksjon av inndataene?

A. `return k*v**2` B. `return k*(v/3.6)**2` C. `return k*(v/3,6)**2` D. `return k*(v*3.6)**2` E. `return k*(v*3,6)**2`

c) Ta utgangspunkt i den ferdige rutinen for Eulers metode med tidssteg $\Delta t = 0,10$ s til å beregne tiden t det tar det før hopperne har nådd 98 % av sine terminalhastigheter, og den vertikale høyden h hopperen har falt på dette tidspunktet (målt fra startpunktet).

[Hint: Numpy-funksjonen `argmax` kan brukes til å finne den laveste/første indeksen der elementene i en NumPy-array oppfyller en viss betingelse. Eksempel: for `a=np.array([0.1,2.4,4.0,9.4])` vil `np.argmax(a>3)` returnere `2` , dvs. `a[2]=4.0` er det første elementet som oppfyller a>3. I denne oppgaven: Dersom terminalfarten er `v_terminal`, og `v_liste` er en array med fartsverdiene, dvs. $v(t)$, og man vil finne ut hvilken t-verdi som tilsvarer 98 % av terminalfarten, blir syntaksen `np.argmax(v>0.98*v_terminal)` ("første/laveste indeks som oppfyller at farten er **større** enn 98 % av terminalfarten"). Hvis denne returnerer f.eks. 199 (verdi av indeks), kan man hente ut t-verdien fra tids-arrayet med `t_liste[199]`.

Svar:

a) Fra newtons 2. lov:

$\sum F = 0$

gir $G - f_d = 0$

deretter får vi:

$v_{max} = \sqrt{\frac{mg}{k}} \implies \sqrt{\frac{2mg}{A\rho C_d}}$

og setter inn for verdiene vi har oppgitt:

for A: $v_{max} = \sqrt{\frac{2(70{,}0kg)\cdot(9{,}81m/s^2)}{0{,}17m^2\cdot 1{,}29kg/m^3\cdot 0{,}70}} = \underline{\underline{94,58m/s}}$

og B: $v_{max} = \sqrt{\frac{2(70{,}0kg)\cdot(9{,}81m/s^2)}{1{,}0m^2\cdot 1{,}29kg/m^3\cdot 1{,}0}} = \underline{\underline{32,63m/s}}$

b, c)

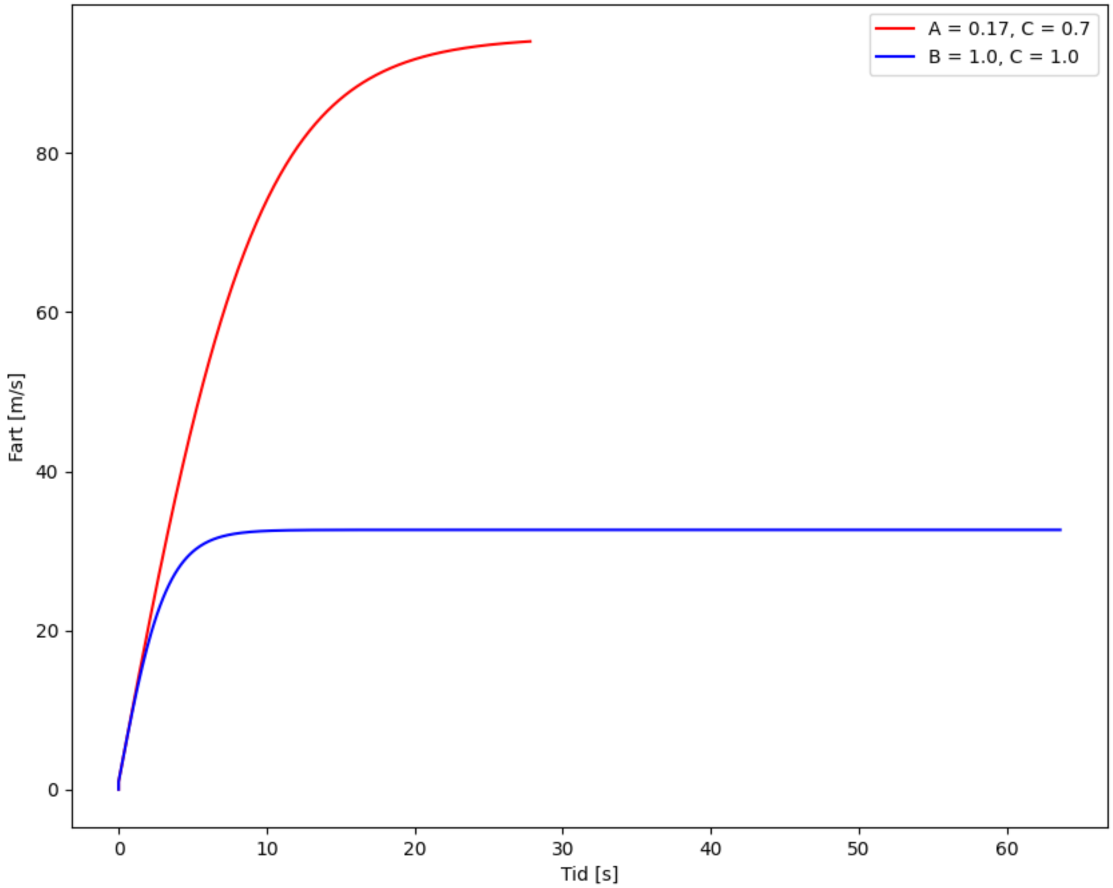
[ ]:
```python
import numpy as np
import matplotlib.pyplot as plt

# Globale konstanter
m=70 #Legemets masse i kg
g=9.81 #Tyngdeakselerasjonen i m/s^2

def drag(A,C,v):
    rho=1.29
    k=0.5*rho*A*C
    #? Her må du legge inn riktig kode for at funksjonen skal returnere
    #luftmotstanden (drag) i newton
    return k*v**2


def dXdt(X, A, C):
    """Funksjonen dXdt beregner høyresiden f(X) i differensiallikningssystemet;␣
 ↪dX/dt=f(X).
    Input:
    X:  X=[y,v] en vektor som inneholder posisjon y og (vertikal) fart v. Med␣
 ↪positiv retning
    nedover er v > 0 og y < 0    for et legeme som faller vertikalt mot bakken,␣
 ↪der y = 0.

    Output:
```

```
    [dydt,dvdt]: Array med nye verdier for hastighet (dydt) og akselerasjon␣
 ↪(dvdt)
    """
    y , v =X         #Koordinater y og v hentes fra inndatavektor X
    f=drag(A,C,v)    #Luftmotstand i N
    dydt=v           #Sammenhengen mellom y og v er at v = dy/dt
    dvdt=-f/m+g      #Akselerasjonen a=dv/dt, fra Newtons 2. lov
    return np.array([dydt,dvdt])

def euler(t0,y0,v0,dt, A, C):
    """Funksjon som bruker Eulers metode til å løse et system av␣
 ↪differensiallikninger,
    dX/dt = f(X), der X =[y,v] er
    en vektor som inneholder posisjons- og hastighetsvariable.
    Input:
    t0: Starttid [s]
    y0: Startverdi for y [m]
    v0: Vertikal startfart [m/s]
    dt: Tidssteg [s]

    Output:
    t_liste: array med t-verdier,[t0,...tn]
    y_liste: array med y-verdier, [y0,...,yn]
    v_liste: array med v-verdier, [v0,...,vn]
    """

    X0=np.array([y0,v0]) #X0 er en vektor med posisjon og fart ved t=t0
    t_liste=[0.0]# Liste med t-verdier
    y_liste=[y0]# liste med y-verdier
    v_liste=[v0] # liste med v-verdier
    X=X0  # initierer loop
    t=t0
    y=y0
    while y<=0: #Loop kjøres inntil legemet treffer bakken; pos. retning␣
 ↪nedover gir y0 < 0
        Xn=X+dt*dXdt(X, A, C) #Beregner neste steg Xn i Euler-metoden
        y=Xn[0] #Henter ut y-koordinat fra array
        v=Xn[1] #Henter ut fart v fra array
        t_liste.append(t) # t-verdi legges til liste
        y_liste.append(y)# y-verdi legges til liste
        v_liste.append(v)# v-verdi legges til liste
        t=t+dt #Ny tidsverdi
        X=Xn #Ny verdi for X
    return t_liste,y_liste,v_liste
```

[ ]:
```
#Eksempel på bruk av rutinene
#Initialiserer variable
```

```
t0=0.0 #t = 0 i startpunktet
v0=0 #Startfart
y0=-2000 #Med positiv retning nedover, er y-verdier negative over bakken.
#Her starter fallet 2,0 km over bakken.
dt=0.1 #Tidssteg
A_A=0.17 #Frontareal
A_C=0.7 #Drag-koeffisient

B_A=1.0 #Frontareal
B_C=1.0 #Drag-koeffisient

#Bruker Eulers metode til å generere sammenhengende verdier for t, y og v
A_t_liste,A_y_liste,A_v_liste=euler(t0,y0,v0,dt,A_A,A_C)

B_t_liste,B_y_liste,B_v_liste=euler(t0,y0,v0,dt,B_A,B_C)


A_v_terminal=A_v_liste[np.argmax(A_v_liste>0.98*A_v_liste[-1])]

B_v_terminal=B_v_liste[np.argmax(B_v_liste>0.98*B_v_liste[-1])]


print(f'A: v_terminal: {A_v_terminal:.2f} m/s')
print(f'B: v_terminal: {B_v_terminal:.2f} m/s')


#Plotter fartsgraf v(t)
plt.figure(figsize = (10, 8))
#Diagrammet angir hvilke verdier for A og C som er tilhører grafen
plt.plot(A_t_liste,A_v_liste,color="red",label='A = '+str(A_A)+", C =␣
 ↪"+str(A_C))
plt.plot(B_t_liste,B_v_liste,color="blue",label='B = '+str(B_A)+", C =␣
 ↪"+str(B_C))

plt.xlabel("Tid [s]")
plt.ylabel("Fart [m/s]")
plt.legend()
plt.show()
```

```
A: v_terminal: 92.17 m/s
B: v_terminal: 31.98 m/s
```


A = 0.17, C = 0.7
B = 1.0, C = 1.0
Fart [m/s]
0
20
40
60
80
0
10
20
30
40
50
60
Tid [s]