Læringsmål?

Eksempeloppgaver med programmering i Fysikk

Dette er et sett oppgaver som er ment å antyde aktuelle problemstillinger for eksamensoppgaver med programmering i Fysikk. Alle oppgavene er i flervalgsformat, da dette er formatet som vil bli brukt for slike oppgaver på eksamen (dvs. man vil **ikke** bli bedt om å skrive egen kode).

Oppgaver med likningsløsning

SciPy-funksjonen `fsolve` løser en likningen $f(x) = 0$, som illustrert i eksemplet under.

[ ]:
```
#Eksempel: løs likingen a*x**3=-b, der a og b er parametre

from scipy.optimize import fsolve
import numpy as np

#Definerer funksjonen f(x) slik at likningen kan skrives på formen f(x)=0.
#Her er x en inputvariabel, mens a og b inputparametre til funksjonen.
def f(x,a,b):
    return a*x**3+b

#Finn løsningen x1 av likninga tilsvarende en starverdi x = -1 og
#gitte verdier av parametrene a og b
a=1
b=2
x1=fsolve(f,-1,(a,b))
```

Oppgave 1

En ball skytes uten luftmotstand fra et utgangspunkt som ligger over den horisontale bakken. Ballen skytes ut med startfart v_0 og startvinkel α. Idet ballen treffer bakken, er posisjonon y, og $y < 0$ hvis vi velger positiv retning nedover. Se figuren under.

Vi skal lage et Python-skript som først finner falltiden t, og deretter beregner den horisontale rekkevidden $x = v_0 \cos \alpha t$.

a) Bestem hva som må stå i linje 35, indikert med ? i kodecellen, for at funksjonen `horisontal_rekkevidde(alfa,v0,y)` skal returnere størrelsen x på figuren over, for gitte verdier av α, v_0 og y.

[ ]:
```
#Oppgave 1a)
from scipy.optimize import fsolve
import numpy as np

#Definerer verdi for g som en global variabel
g=9.81

def f(t,alfa,v0,y):
```

```
    """Funksjonen definerer likningen f(t)=0 for å finne tiden før legemet␣
↪treffer bakken.
    Input:
    t: Tid, avhengig variabel [s]
    Parametre:
    alfa: Utskytingsvinkel med horisontalen [radianer]
    v0: Startfart [m/s]
    y: Posisjon når legemet treffer bakken [m] (NB: y < 0 pga. positiv retning␣
↪oppover)
    """
    return -0.5*g*t**2+v0*np.sin(alfa)*t-y

def horisontal_rekkevidde(alfa,v0,y):
    """Funksjonen beregner den horisontale rekkevidden [m], for et gitt sett av␣
↪input.
    Input:
    alfa: Utskytingsvinkel med horisontalen [grader]
    v0: Startfart [m/s]
    y: Starthøyde over bakken [m] (NB: y <= 0 pga. positiv retning oppover)

    Output:
    Horisontal rekkevidde, dvs. horisontal avstand utgangspunkt-nedslagspunkt␣
↪[m]
    """
    alfa=np.radians(alfa) #Konverterer vinkel til radianer
    t_start=1 #Startverdi/gjetning for t i fsolve
    #Beregner falltiden t. Vi leter etter løsninger for t, slik at alfa,v0,y er
    #parametre/gitte verdier til funksjonen f. Syntaksen (alfa,v0,y) angir at
    #f skal betraktes som en funksjon av t; ikke av alfa,v0,y.
    t=fsolve(f,t_start,(alfa,v0,y)) #Falltiden t
    return ?  #Returnerer horisontal rekkevidde

#Definerer verdier for kastet
alfa=30 #Utgangsvinkel i grader
v0=7 #Startfart i m/s
y=0 #Starthøyde i m over bakkenivå. NB! y <= 0 pga. positiv retning oppover

print(horisontal_rekkevidde(alfa,v0,y))
```

A. `v0*np.cos(alfa)*t`

B. `v0*np.sin(alfa)*t`

C. `v0*np.tan(alfa)*t`

D. `v0*np.cos(np.radians(alfa))*t`

E. `v0*np.sin(np.radians(alfa))*t`

b) I kodecellen over gjør funksjonen `horisontal_rekkevidde` et kall til `fsolve` med en fast startverdi `t_start`, som `fsolve` leter etter løsninger i nærheten av (i eksempelkoden er startverdien $t = 1,0$ s). Vi skal etter hvert utvide koden til å beregne vinkelen som gir **maksimal** horisontal rekkevidde og da må vi bruke "bedre" startverdier/gjetninger for t, som også tar høyde for spesialtilfellene $\alpha = 0$ ("horisontalt kast") og/eller $y = 0$ (skyter ut fra bakkenivå).

Vi vil foreta følgende gjetninger:

1. For $\alpha > 0$: Startverdi settes lik falltiden for tilfellet der $y = 0$, som lett lar seg beregne.
2. For $\alpha = 0$ og $y < 0$ (horisontalt kast fra et nivå over bakkeplan): Startverdi settes lik falltiden for et vertikalt fall fra samme høyde, dvs. tilsvarende at legemet slippes med null vertikal startfart fra samme starthøyde.
3. For alle andre tilfeller: Startverdi settes til `t=1`.

Kodecellen under viser den modifiserte koden i `horisontal_rekkevidde` som beregner `t_start` før funksjonskallet til `fsolve`(resten av koden er uendret).

Bestem hva som skal stå i kodelinje 5, indikert med ? i kodecellen.

[ ]:
```
#Oppgave 1b)
#Beregner startverdi/gjetning for t avhengig av v0 og y
if (alfa>0):
    t_start=2*v0*np.sin(alfa)/g #Startverdi for t lik falltid for y = 0
elif (alfa==0 and y<0):
    t_start=? #Startverdi for t lik falltid for vertikalt fall fra samme␣
 ↪starthøyde
else:
    t_start=1
```

A. `t_start=(2*(-y)/g)**0.5`

B. `t_start=(2*y/g)**0.5`

C. `t_start=(2*y/g)`

D. `t_start=(-2*y/g)`

E. `t_start=(-y/g)`

c) Vi skal nå modifisere koden for å bestemme den vinkelen $\alpha_{\max}$ som gir den **maksimale** horisontale rekkevidden. Algoritmen:

1. Definer et NumPy-array `vinkler` med alle vinkler i intervallet $[0, 90°]$
2. Definer en vektorisert utgave `horisontal_rekkevidde_vektorisert` av funksjonen `horisontal_rekkevidde`, slik at funksjonen beregner rekkevidden for hver vinkel i arrayet `vinkler`
3. Kall `horisontal_rekkevidde_vektorisert` for hver vinkel og lagre rekkeviddene i et NumPy-array `rekkevidde`
4. Finn indeks for den maksimale verdien lagret i `rekkevidde`
5. Hent ut vinkelen som gir maksimal rekkevidde fra `vinkler`

Bestem hva som skal stå i kodelinje 5, indikert med ? i kodecellen.

[ ]:
```
def alfa_maks(v0,y):
    vinkler=np.linspace(0,90,1000) #array med vinkler
    horisontal_rekkevidde_vektorisert=np.vectorize(horisontal_rekkevidde)␣
↪#vektoriserer funksjonen
    rekkevidde=horisontal_rekkevidde_vektorisert(vinkler,v0,y) #beregner␣
↪rekkevidde for alle vinkler i array
    indeks=? #henter ut indeks for maksimal verdi i array
    alfa_maks=vinkler[indeks] #henter ut vinkel for maksimal rekkevidde
    return alfa_maks

#Beregner maksimal rekkevidde for gitte verdier av v0 og y
v0=7
y=-9
print(alfa_maks(v0,y))
```

A. `rekkevidde[0]`

B. `np.argmax(vinkler)`

C. `np.argmax(rekkevidde)`

D. `np.argmin(rekkevidde)`

E. `np.argmin(vinkler)`

Oppgave 2

En ball skytes uten luftmotstand mot en blink som har posisjon (x, y) i forhold til utskytingspunktet, med $y < 0$ (positiv retning oppover). Ballen skytes ut med startfart v_0 og startvinkel α, som vist på figuren under.

Bestem hva som må stå i linja indikert med ? i kodecellen under for at funksjonen $f(\alpha)$ gjør at `fsolve` beregner verdier av startvinkelen α som gjør at ballen treffer blinken, for gitte verdier av x_0, y_0 og v_0.

[ ]:
```
from scipy.optimize import fsolve
import numpy as np

g=9.81

def f(alfa,x,y,v0):
    """Funksjonen definerer likningen f(alfa)=0 som bestemmer vinkel alfa som␣
↪gir treff i (x,y).
    Input:
    alfa: Startvinkel, avhengig variabel [grader]
    Parametre:
    x: x-koordinat til blinken
    y: y-koordinat til blinken [m]. NB! y < 0 for valgt positiv retning
    v0: Startfart [m/s]
    """
```

```
    alfa=np.radians(alfa) #konverterer x til radianer
    return y+0.5*g*x**2/(v0**2*(np.cos(alfa))**2) -x*np.tan(alfa)

#Definerer startverdier

x=1.5
y=-0.4
v0=4.0
alfa0=60

#Leter etter løsninger for alle startvinkler i intervallet [0 grader, 90 grader]
#for alfa in range(0,90):
#vinkler=np.linspace(0,90,1000)
#vfunk=np.vectorize(f)
#vinkel=fsolve(f,alfa,(x,y,v0))
losning=fsolve(f,alfa0,(x,y,v0))
print(losning)
```

```
[62.9733725]
```

A.
B.
C.
D.
E.

Oppgave 2

Oppgaver med integrasjon

Oppgave 1

En bil kjører rettlinjet på horisontalt underlag, og påvirkes av luftmotstand med absoluttverdi $F_D = kv^2$ som virker motsatt av fartsretningen. Bilen er utstyrt med en datalogger som gir bilens fart v som funksjon av tiden t, dvs. $v(t)$.

Vi skal skrive en funksjon som beregner arbeidet W utført av luftmotstanden fra $t = t_0$ til $t = t_1$, ved å bruke definisjonen av effekt og arbeid: $P = \frac{dW}{dt} \Rightarrow W = \int_{t_0}^{t_1} P(t)dt$.

[ ]:
```
import numpy as np
import scipy.integrate as integrate

def v(t):
    #Gir fart v(t) [m/s] basert på data fra datalogger
    return 0.5*t**2-3.0*t

def P(t):
    k=2.0 #Verdi for k [Ns^2/m^2]
    return k*v(t)**3
```

```
def W(t0,t1):
    W=integrate.quad(P,t0,t1)[0]
    return W

print(W(0,10))
```

```
12142.857142857147
```

[ ]:
```
import numpy as np
import scipy.integrate as integrate
t=np.arange(0,20,0.1)
v=0.5*0.2*t**1.44
I=np.trapz(v**3)
print(I)
```

```
15275.932727818745
```

Oppgave 2

[ ]:
```
import numpy as np
t=np.arange(0,20,0.1)
v=0.5*0.2*t**1.44
print(v)
```

```
[0.00000000e+00 3.63078055e-03 9.85106521e-03 1.76625989e-02
 2.67279954e-02 3.68567304e-02 4.79223162e-02 5.98330550e-02
 7.25186284e-02 8.59229568e-02 1.00000000e-01 1.14711106e-01
 1.30023243e-01 1.45907772e-01 1.62339563e-01 1.79296335e-01
 1.96758170e-01 2.14707120e-01 2.33126910e-01 2.52002695e-01
 2.71320865e-01 2.91068886e-01 3.11235165e-01 3.31808941e-01
 3.52780189e-01 3.74139544e-01 3.95878231e-01 4.17988006e-01
 4.40461106e-01 4.63290204e-01 4.86468369e-01 5.09989036e-01
 5.33845970e-01 5.58033246e-01 5.82545217e-01 6.07376499e-01
 6.32521950e-01 6.57976651e-01 6.83735893e-01 7.09795161e-01
 7.36150120e-01 7.62796609e-01 7.89730622e-01 8.16948305e-01
 8.44445944e-01 8.72219958e-01 9.00266889e-01 9.28583400e-01
 9.57166261e-01 9.86012351e-01 1.01511865e+00 1.04448223e+00
 1.07410024e+00 1.10396995e+00 1.13408868e+00 1.16445383e+00
 1.19506289e+00 1.22591340e+00 1.25700299e+00 1.28832934e+00
 1.31989019e+00 1.35168334e+00 1.38370667e+00 1.41595806e+00
 1.44843551e+00 1.48113701e+00 1.51406063e+00 1.54720449e+00
 1.58056672e+00 1.61414554e+00 1.64793917e+00 1.68194590e+00
 1.71616403e+00 1.75059192e+00 1.78522794e+00 1.82007054e+00
 1.85511814e+00 1.89036925e+00 1.92582237e+00 1.96147606e+00
 1.99732888e+00 2.03337944e+00 2.06962636e+00 2.10606831e+00
 2.14270396e+00 2.17953202e+00 2.21655121e+00 2.25376030e+00
 2.29115804e+00 2.32874325e+00 2.36651474e+00 2.40447134e+00
```

```
2.44261192e+00 2.48093534e+00 2.51944052e+00 2.55812635e+00
2.59699178e+00 2.63603576e+00 2.67525725e+00 2.71465523e+00
2.75422870e+00 2.79397669e+00 2.83389821e+00 2.87399232e+00
2.91425808e+00 2.95469455e+00 2.99530082e+00 3.03607601e+00
3.07701921e+00 3.11812956e+00 3.15940620e+00 3.20084828e+00
3.24245496e+00 3.28422542e+00 3.32615885e+00 3.36825443e+00
3.41051139e+00 3.45292894e+00 3.49550631e+00 3.53824274e+00
3.58113748e+00 3.62418980e+00 3.66739895e+00 3.71076422e+00
3.75428490e+00 3.79796028e+00 3.84178967e+00 3.88577239e+00
3.92990775e+00 3.97419509e+00 4.01863375e+00 4.06322307e+00
4.10796241e+00 4.15285113e+00 4.19788860e+00 4.24307420e+00
4.28840731e+00 4.33388733e+00 4.37951365e+00 4.42528568e+00
4.47120283e+00 4.51726452e+00 4.56347017e+00 4.60981922e+00
4.65631110e+00 4.70294525e+00 4.74972114e+00 4.79663820e+00
4.84369591e+00 4.89089373e+00 4.93823113e+00 4.98570759e+00
5.03332260e+00 5.08107564e+00 5.12896621e+00 5.17699381e+00
5.22515793e+00 5.27345810e+00 5.32189382e+00 5.37046461e+00
5.41917000e+00 5.46800951e+00 5.51698269e+00 5.56608905e+00
5.61532816e+00 5.66469954e+00 5.71420276e+00 5.76383737e+00
5.81360292e+00 5.86349899e+00 5.91352513e+00 5.96368092e+00
6.01396593e+00 6.06437974e+00 6.11492194e+00 6.16559211e+00
6.21638984e+00 6.26731472e+00 6.31836635e+00 6.36954433e+00
6.42084827e+00 6.47227777e+00 6.52383245e+00 6.57551192e+00
6.62731579e+00 6.67924369e+00 6.73129524e+00 6.78347007e+00
6.83576781e+00 6.88818810e+00 6.94073056e+00 6.99339484e+00
7.04618058e+00 7.09908743e+00 7.15211504e+00 7.20526304e+00
7.25853111e+00 7.31191890e+00 7.36542606e+00 7.41905226e+00]
```

[ ]:
```
print(v_array)
```

```
[0.  0.2 0.6 1.1 2.4 3.8 4.9]
```

Oppgave x

Et legeme med masse m som synker i vann, påvirkes av to krefter: tyngden $G = mg$ (antas konstant), samt en væskemotstand $F_D = kv$, der k er en positiv konstant.

Programmet under bruker Eulers metode til å bestemme legemets fart $v(t)$, fra $t = 0$ til en sluttid $t = t_1$, for gitte verdier av m, g og k.

Hvilken Pyton-kode skal stå i linja markert med ? for at programmet skal generere to lister: én liste `t_liste` med t-verdier fra $t0$ til t_1, og én liste `v_liste` som inneholder fartsverdiene $v(t)$?

[ ]:
```
import numpy as np

def dvdt(v):
    #Funksjonen beregner akselerasjonen a=dv/dt for legemet som funksjon av␣
 ↪farten v
    g=9.81 #Tyngdeakselerasjonen
```

```
    m=1.0E-3 #Legemets masse
    k=1.0E-3 #Verdien av konstanten k
    return ? #Returnerer akselerasjonen a(v)

#Tidssteg
dt=0.1

#Startverdier for t og v, legges i start på liste
t0=0
v0=0
t_liste=[t0]
v_liste=[v0]

#Initaliserer t og v(t)
t=t0
v=v0

while(t<t1):
    v=v+dvdt(v)*dt
    t=t+dt
    v_liste.append(v)
    t_liste.append(t)
```

[ ]:
```
#Fasit

import numpy as np

def dvdt(v):
    #Beregner akselerasjonen a=dv/dt for legemet som funksjon av farten v
    g=9.81 #Tyngdeakselerasjonen
    m=1.0E-3 #Legemets masse
    k=1.0E-3 #Verdien av konstanten k
    return g-(k/m)*v #Returnerer akselerasjonen a(v)

#Tidssteg
dt=0.1

#Startverdier for t og v, legges i start på liste
t0=0
v0=0
t_liste=[t0]
v_liste=[v0]

#Initaliserer t og v(t)
t=t0
v=v0
```

```
while(t<t1):
    v=v+dvdt(v)*dt
    t=t+dt
    v_liste.append(v)
    t_liste.append(t)
```

[ ]: