FYS1100 Mekanikk og modellering, vår 2025

Forelesning uke 8: Energiprinsippet. Numerisk derivasjon og integrasjon

(AMS 11.4, 6.2, 7.5)

Læringsmål: Vite hva **numerisk derivasjon** og **integrasjon** er og hvordan det brukes i praksis. Se på betydningen av størrelsen på ***tidsintervallet***.

Beskjeder:

✓ …

Mandag 10.03.2025

Midtveiseksamen ☺️

Skriftlig, digital eksamen med Inspera:
https://www.uio.no/studier/eksamen/inspera/index.html

1. Tid og sted: tirsdag 18. mars kl. 15-18, Silurveien. OBS: sjekk hvor du skal møte i Studentweb! ***Møt i god tid!***
2. Digital eksamen med 20 flervalgsoppgaver, PC-er er på plassene på Silurveien (NB: ikke lov å ha med egen PC til eksamen!)
3. Pensum: alt vi har gjennomgått så langt + denne ukas temaer: grovt sett fra og med kapittel 3 til og med kapittel 11 i læreboka, og notatet om diff-ligninger, og noen delkapitler fra Kalkulus-boka (se liste som legges ut i Canvas)
4. Tillatte hjelpemidler: godkjent kalkulator og Rottmann
5. Formelark er tilgjengelig i Inspera

«Trøsterunde» 🤗:
Om dere lurer på noe i selve oppgaveteksten, kommer jeg innom og går en runde ca. kl. 16 og ca kl. 17. Jeg kan *ikke* svare på om dere har gjort riktig! Kun avklare selve oppgaveteksten.

Formelark FYS1100

$\sum \vec{F} = m\vec{a} = \frac{d\vec{p}}{dt}$, hvor $\vec{p} = m\vec{v} = m\frac{d\vec{r}}{dt}$ og $\vec{a} = \frac{d\vec{v}}{dt} = \frac{d^2\vec{r}}{dt^2}$

Konstant $\vec{a}$: $\vec{v} = \vec{v}_0 + \vec{a}t$, $\vec{r} = \vec{r}_0 + \vec{v}_0 t + \frac{1}{2}\vec{a}t^2$, $v^2 - v_0^2 = 2\vec{a}\cdot(\vec{r} - \vec{r}_0)$

Konstant α: $\omega = \omega_0 + \alpha t$, $\theta = \theta_0 + \omega_0 t + \frac{1}{2}\alpha t^2$, $\omega^2 - \omega_0^2 = 2\alpha(\theta - \theta_0)$

Baneakselerasjon: $\vec{a} = \frac{d\vec{v}}{dt}\hat{u}_T + \frac{v^2}{r}\hat{u}_N$

Rotasjon: $\vec{v} = \vec{\omega} \times \vec{r}$, $\vec{a} = \vec{\alpha} \times \vec{r} + \vec{\omega} \times (\vec{\omega} \times \vec{r})$

Galilei-transformasjon, referansesystemer: $\vec{r} = \vec{R} + \vec{r}'$, $\vec{v} = \vec{V} + \vec{v}'$, $\vec{a} = \vec{a}'$

Fjærkraft: $F(x) = -k(x - x_0)$, luftmotstand: $\vec{F}_v = -k\vec{v}$ eller $\vec{F}_v = -D|\vec{v}|\vec{v}$

Statisk friksjon: $|\vec{F}_s| \leq \mu_s N$, dynamisk friksjon: $|\vec{F}_d| = \mu_d N$

Arbeid: $W_{AB} = \int_{t_A}^{t_B} \vec{F}\cdot\vec{v}\,dt = \int_A^B \vec{F}\cdot d\vec{r} = K_B - K_A$, kinetisk energi: $K = \frac{1}{2}mv^2$

Potensiell energi for gravitasjon: $U = mgy$, for fjærkraft: $U = \frac{1}{2}k(x - x_0)^2$

Effekt: $P = \lim_{\Delta t \to 0}\frac{\Delta W}{\Delta t} = \vec{F}\cdot\vec{v}$

Konservativ kraft: 1D: $F(x) = -\frac{dU(x)}{dx}$, 3D: $\vec{F} = -\vec{\nabla}U(\vec{r})$

Impuls: $\vec{J} = \int_{t_0}^{t_1}\vec{F}dt = \Delta\vec{p} = \vec{p}(t_1) - \vec{p}(t_0)$ Rakettligningen: $\vec{F}^{\text{ext}} + \vec{v}_{\text{rel}}\frac{dm}{dt} = m\vec{a}$

Massesenter: $\vec{R} = \frac{1}{M}\sum_i m_i\vec{r}_i = \frac{1}{M}\int_M \vec{r}dm$, $M = \sum_i m_i = \int_M dm$

Kraftmoment: $\vec{\tau} = \vec{r} \times \vec{F}$, spinn: $\vec{L} = \vec{r} \times \vec{p}$

Spinnsats: $\vec{\tau} = \frac{d\vec{L}}{dt}$ Stive legemer: $L_z = I_z\omega_z$, $\tau_z = I_z\alpha_z$

Kinetisk rotasjonsenergi: $K = \frac{1}{2}I\omega^2$ Treghetsmoment: $I = \sum_i m_i\rho_i^2 = \int_M \rho^2 dm$

Parallellakseteoremet: $I = I_{cm} + Md^2$ Rullebetingelse: $V = \omega R$

Gravitasjon: $\vec{F}(\vec{r}) = -G\frac{m_1 m_2}{r^2}\hat{u}_r$, $U(r) = -G\frac{m_1 m_2}{r}$; G er gravitasjonskonstanten

Lorentz-transformasjon: $x' = \gamma(x - ut)$, $y' = y$, $z' = z$, $t' = \gamma\left(t - \frac{u}{c^2}x\right)$, $\gamma = \frac{1}{\sqrt{1-\frac{u^2}{c^2}}}$

Lorentz-transformasjon for hastighet: $v' = \frac{v-u}{1-\frac{u}{c^2}v}$

Tidsdilatasjon: $\Delta t = \gamma\Delta t_0$; lengdekontraksjon: $l = \frac{l_0}{\gamma}$; relativistisk kinetisk energi: $K = m_0c^2(\gamma - 1)$, hvor m_0 er hvilemassen.

Taylor-rekke til en funksjon $f(x)$ om funksjonsargumentet $x = a$:

$T_n(f;a)(x) = f(a) + \frac{1}{1!}(x-a)\frac{df(x=a)}{dx} + \frac{1}{2!}(x-a)^2\frac{d^2f(x=a)}{dx^2} + \cdots + \frac{1}{n!}(x-a)^n\frac{d^nf(x=a)}{dx^n}$

Restledd: $R_n(f;a)(x) = \frac{(x-a)^{n+1}}{(n+1)!}\cdot\frac{d^{n+1}f(x=\zeta)}{dx^{n+1}}$ hvor $\zeta \in [x,a]$ er ukjent.

Difflikninger: Separabel 1.ordens lineær diff-likn.:
$q(x)\cdot\frac{dx}{dt} = p(t) \Rightarrow \int q(x)\,dx = \int p(t)\,dt \Rightarrow Q(x) = P(t) + C$, med $\frac{dQ}{dx} = q(x)$ og $\frac{dP}{dt} = p(t)$
2.ordens homogene lineære diff-likn. med konstante koeffisienter:
$\frac{d^2x(t)}{dt^2} + p\frac{dx(t)}{dt} + qx(t) = 0 \Rightarrow$ Karakteristisk likn. $\lambda^2 + p\lambda + q = 0$ med generelt to røtter $\lambda_1$og λ_2.
Generelle løsninger: To reelle røtter: $x(t) = C_1e^{\lambda_1 t} + C_2e^{\lambda_2 t}$ En reell rot: $x(t) = C_1e^{\lambda t} + C_2te^{\lambda t}$
To komplekse røtter, $\lambda_1 = a + ib, \lambda_2 = a - ib : x(t) = e^{at}(C_1\cos(bt) + C_2\sin(bt))$

Ikke-konservative krefter og energiprinsippet 💃

[✏️ lysbilde med skriblerier på iPad ✏️]

Vi kan dele opp nettokraften vår i

- Den konservative kraften (eller summen av dem): $\vec{F}$
- Den ikke-konservative kraften (eller summen av dem): $\vec{f}$

Vi vet at for en konservativ kraft,
kan vi finne et potensial slik at:

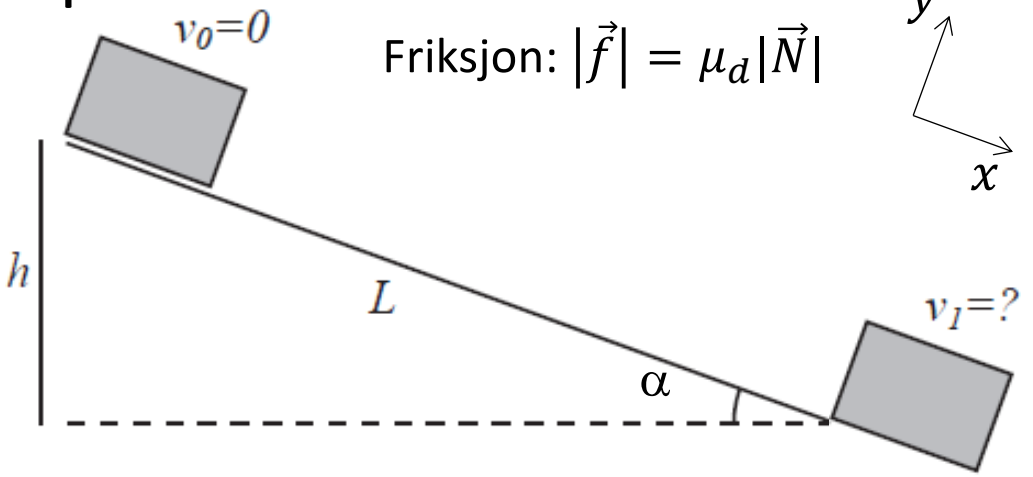
Arbeid-energi-teoremet:

Forandring i den mekaniske energien = arbeid gjort av *ikke*-konservative krefter 😀

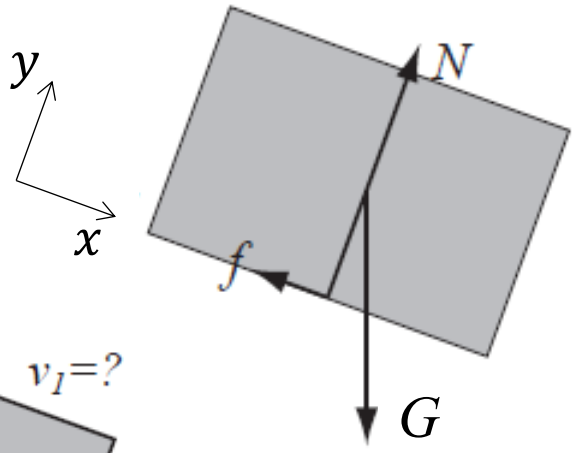
Eksempel: bok på et skråplan

[✏️ lysbilde med skriblerier på iPad ✏️]

Ei bok sklir på et skråplan med helnings-vinkel α og med friksjons-koeffisient μ_d. Vi ser bort fra luftmotstand. Hva er farten v_1 til boka når den har sklidd ned skråplanet?



Friksjon: $|\vec{f}| = \mu_d |\vec{N}|$



N2L i y-retning:

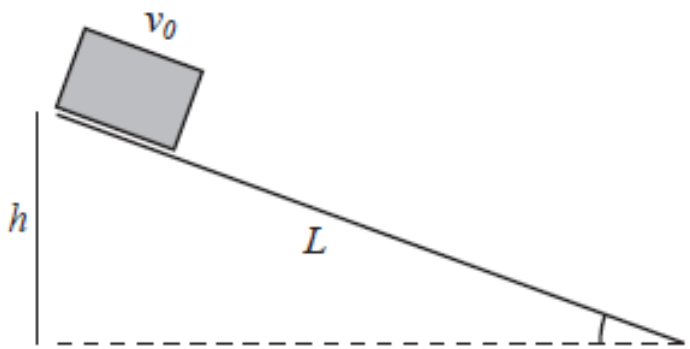
Potensiell energi:

Forandring i mekanisk energi:

Hvor ble det av energien ΔE? 🤔

Termisk energi ☕

[✏️ lysbilde med skriblerier på iPad ✏️]



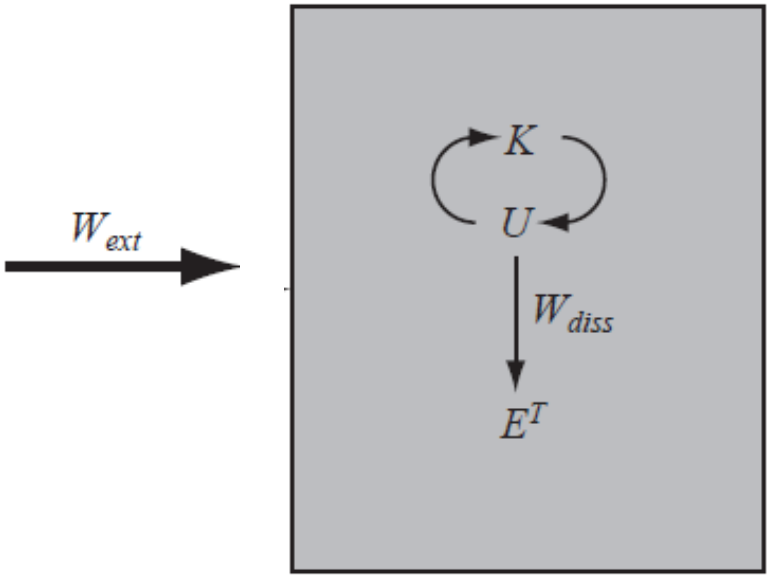
Friksjon ⇒ *atomære* vibrasjoner
kinetisk og potensiell energi på *mikroskopisk nivå*

Mikroskopiske bevegelser ⇒ varme

Friksjon *varmer* boka og planet
⇒ temperaturen i systemet «bok + skråplan» øker

Energien i hele systemet er bevart:

Arbeid fra en ytre kraft:
- jeg skyver boka opp
- jeg løfter boka opp



Lukket system:

- konservative krefter
  kinetisk ⇔ potensiell energi
- ikke–konservative krefter
  (dissipative krefter)
  mekanisk ⇒ termisk energi

Praktisk tips til ukas hovedtema ☺️

Hans Petter Langtangen: «A Primer on Scientific Programming with Python»

https://link.springer.com/book/10.1007/978-3-642-30293-0%20

3.1.7 Multiple Return Values

Python functions may return more than one value. Suppose we are interested in evaluating both $y(t)$ and $y'(t)$:

$$y(t) = v_0 t - \frac{1}{2} g t^2,$$
$$y'(t) = v_0 - g t\,.$$

To return both y and y' we simply separate their
a comma in the `return` statement:

```
def yfunc(t, v0):
    g = 9.81
    y = v0*t - 0.5*g*t**2
    dydt = v0 - g*t
    return y, dydt
```

Calling this latter `yfunc` function makes a need for two
of the assignment operator because the function returns

```
position, velocity = yfunc(0.6, 3)
```

il å oversette)

As an example, consider a function for computing the second-order derivative of a function $f(x)$ numerically:

$$f''(x) \approx \frac{f(x-h) - 2f(x) + f(x+h)}{h^2}, \tag{3.3}$$

where h is a small number. The approximation (3.3) becomes exact in the limit $h \to 0$. A Python function for computing (3.3) can be implemented as follows:

```
def diff2nd(f, x, h=1E-6):
    r = (f(x-h) - 2*f(x) + f(x+h))/float(h*h)
    return r
```

Kap. 3.1.12

Praktisk tips til ukas hovedtema ☺️

Joakim Sundnes: «Introduction to Scientific Programming with Python»

Gratis å laste ned fra UiO-nettet! 👍 NB! Python3 🥰

https://link.springer.com/book/10.1007/978-3-030-50356-7

Kapittel 4: Functions and Branchings



Numerisk derivasjon

[✏️ lysbilde med skriblerier på iPad ✏️]

Definisjonen av den deriverte: $$\frac{df(x)}{dx} = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

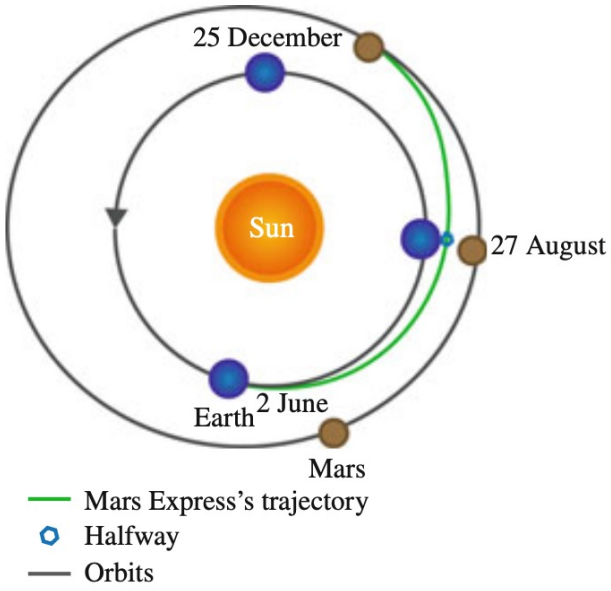
Numerisk derivasjon: diskretiserer f(x) og x

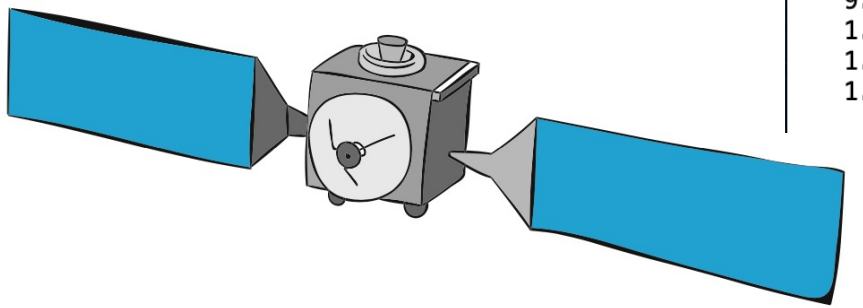
Newtons kvotient:

I praksis kan vi ikke regne ut for alle mulige verdier, vi må gjøre et (klokt) utvalg 🤓

Eksempel: Mars Express

[✏️ lysbilde med skriblerier på iPad ✏️]





Datafil ("low resolution", marsexpresslr.txt):

marsexpresslr.txt

Tid (dager)	x (km)	y (km)	z (km)
4.9532291e-02	-2.8034186e+07	-1.4922374e+08	-3.5527680e+04
3.0066101e+01	5.4163451e+07	-1.4711959e+08	-2.6206946e+05
6.0082670e+01	1.2452776e+08	-1.1167004e+08	-4.2939533e+05
9.0099239e+01	1.7188367e+08	-5.4904690e+07	-5.1677779e+05
1.2011581e+02	1.9351086e+08	1.0496006e+07	-5.2653357e+05
1.5013238e+02	1.9154329e+08	7.4880556e+07	-4.7213981e+05
1.8014894e+02	1.6995059e+08	1.3176488e+08	-3.7005167e+05

30-dagers intervaller

Eksempel: Mars Express: forflytning og endring i $\vec{v}$

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

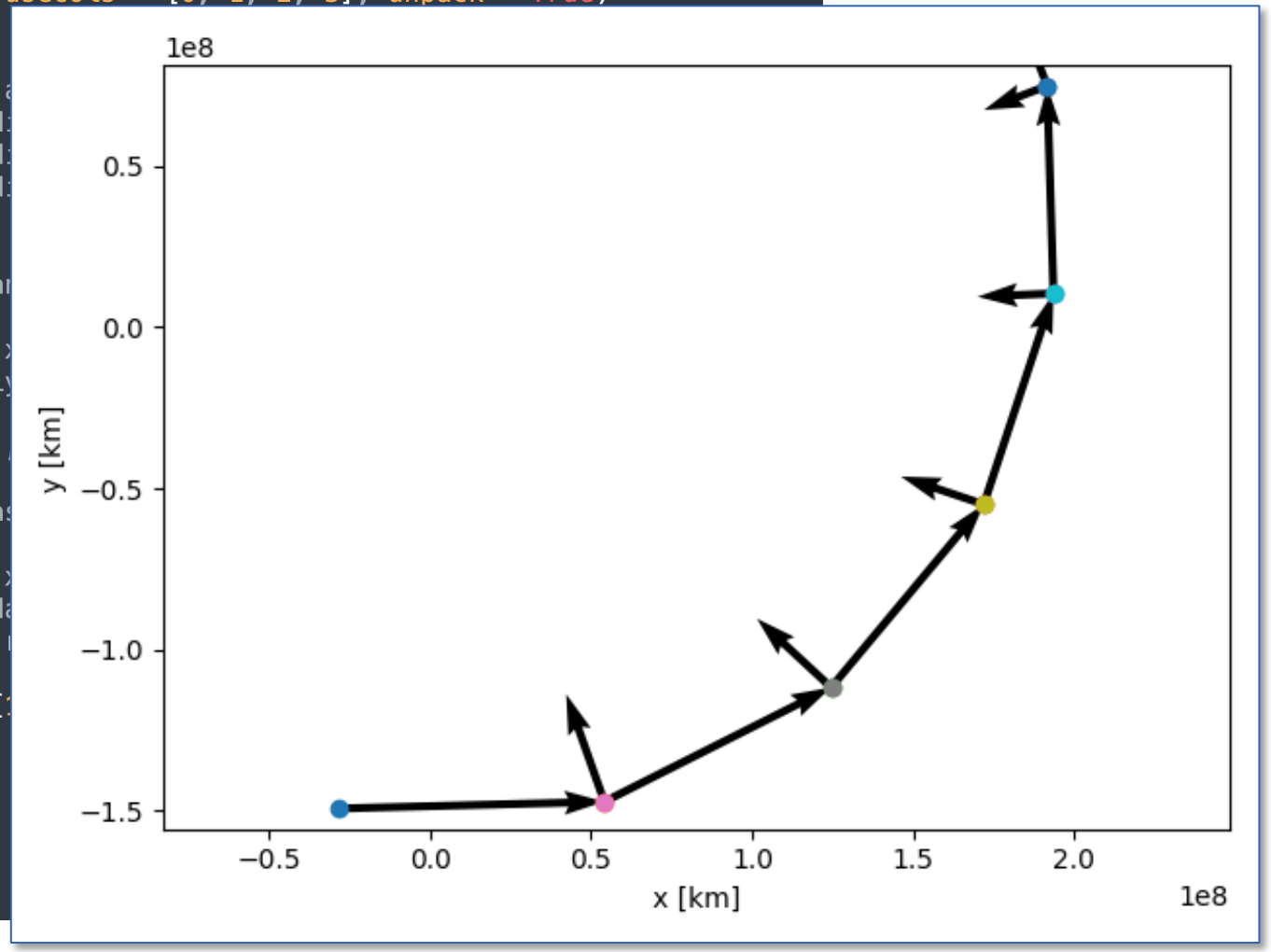
# Lese datafil
t, x, y, z = np.loadtxt('marsexpresslr.txt', usecols = [0, 1, 2, 3], unpack = True)
n = len(t)

r = np.zeros((n,3),float) # Posisjonsvektor, a
r[:, 0] = x # For alle rader fyller vi x-verdi
r[:, 1] = y # For alle rader fyller vi y-verdi
r[:, 2] = z # For alle rader fyller vi z-verdi


# Vi vil gjerne plotte forflytningen i xy-plan
for i in range(n-1):
    plt.plot(r[i, 0], r[i, 1], 'o') # Plotte x
    dr = r[i+1, :] - r[i, :] # Regne ut forfly
    # Vi plotter kun i xy-planet
    plt.quiver(r[i, 0], r[i, 1], dr[0], dr[1],

# Så vil vi også gjerne regne ut endring i has
for i in range(1,n-1):
    plt.plot(r[i, 0], r[i, 1], 'o') # Plotte x
    # Regne ut endring i hastighet for alle da
    dr2 = (r[i+1, :] - r[i, :]) - (r[i, :] - r
    # Igjen plotter vi kun i xy-planet
    plt.quiver(r[i, 0], r[i, 1], dr2[0], dr2[1

plt.xlabel('x [km]')
plt.ylabel('y [km]')
plt.axis('equal')
plt.tight_layout()
plt.show()
```



Python-script: marsexpress.py

Fellesoppgave: Mars Express: hastighet og akselerasjon

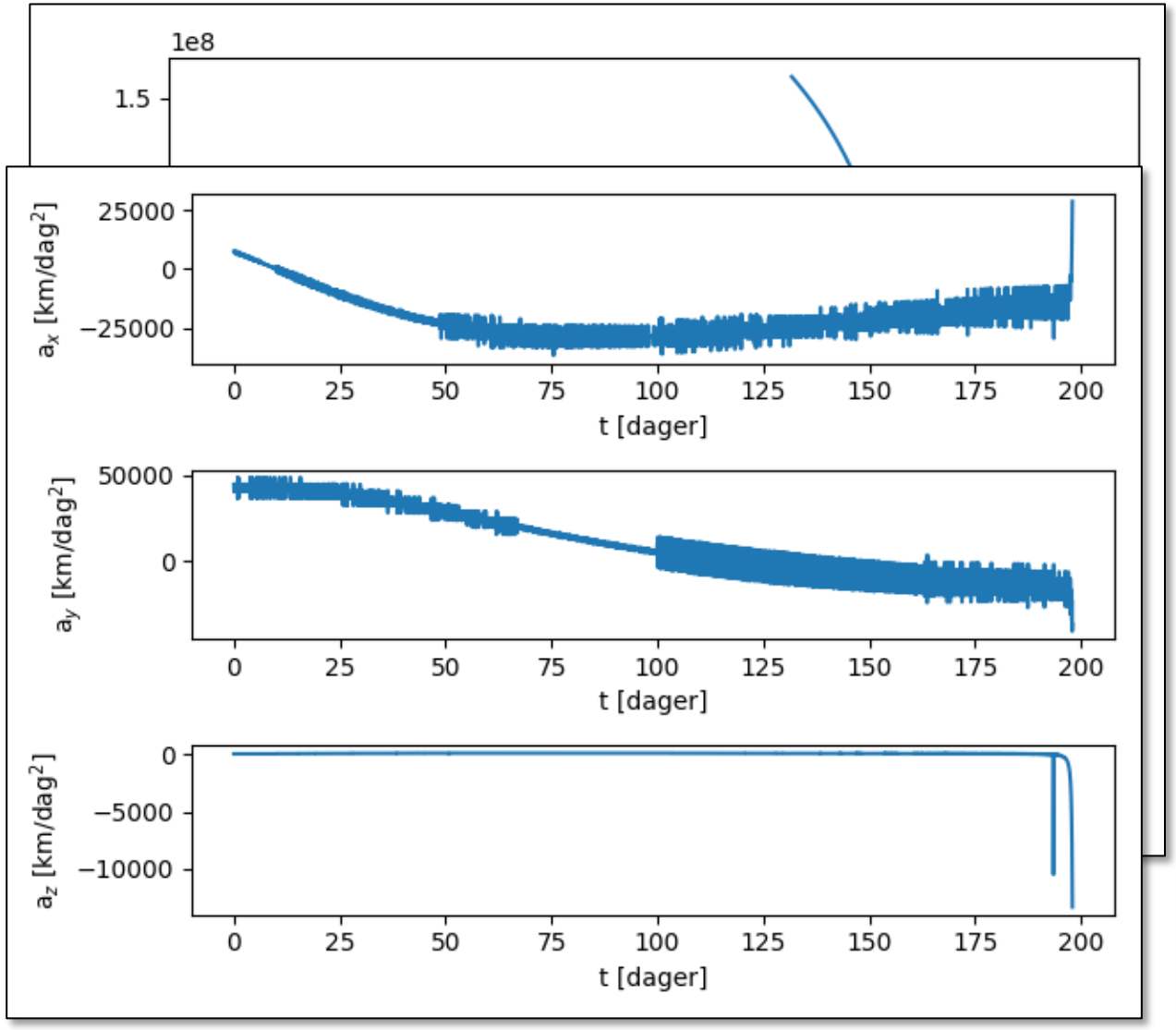
Datafil ("high resolution", marsexpresshr.txt) på semestersiden:

Tid (dager) x (km) y (km) z (km)

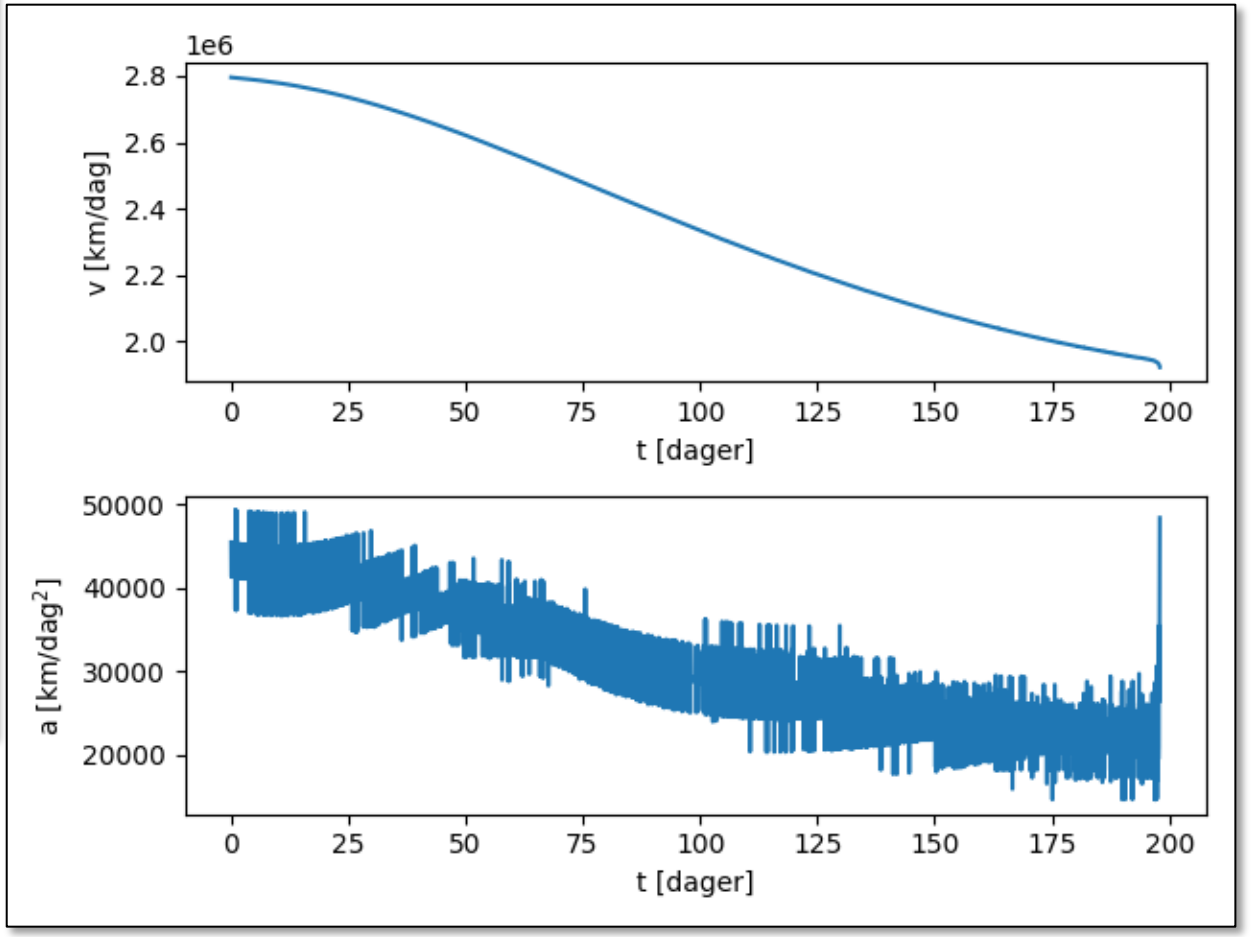
marsexpresshr.txt

```
  0.0000000e+00  -2.8169824e+07  -1.4919588e+08  -3.5126537e+04
  4.9532291e-02  -2.8034186e+07  -1.4922374e+08  -3.5527680e+04
  9.9064583e-02  -2.7898530e+07  -1.4925149e+08  -3.5928763e+04
  1.4859687e-01  -2.7762857e+07  -1.4927914e+08  -3.6329785e+04
  1.9812917e-01  -2.7627166e+07  -1.4930669e+08  -3.6730749e+04
  2.4766146e-01  -2.7491458e+07  -1.4933413e+08  -3.7131653e+04
  2.9719375e-01  -2.7355732e+07  -1.4936147e+08  -3.7532497e+04
  3.4672604e-01  -2.7219989e+07  -1.4938870e+08  -3.7933282e+04
  3.9625833e-01  -2.7084229e+07  -1.4941583e+08  -3.8334007e+04
  4.4579062e-01  -2.6948452e+07  -1.4944286e+08  -3.8734673e+04
  4.9532292e-01  -2.6812658e+07  -1.4946978e+08  -3.9135279e+04
  5.4485521e-01  -2.6676847e+07  -1.4949660e+08  -3.9535825e+04
  5.9438750e-01  -2.6541019e+07  -1.4952331e+08  -3.9936312e+04
  6.4391979e-01  -2.6405175e+07  -1.4954992e+08  -4.0336740e+04
  6.9345208e-01  -2.6269313e+07  -1.4957643e+08  -4.0737107e+04
```

Fellesoppgave: Mars Express: hastighet og akselerasjon



Hva skjer med akselerasjonen?? 🤯🤔



Python-script: marsexpress_v_and_a.py

Numerisk integrasjon

[✏️ lysbilde med skriblerier på iPad ✏️]

En funksjon $F(x)$ som er den integrerte av en annen funksjon $f(x)$: $f(x) = \frac{dF(x)}{dx}$

For en endelig endring i F, $dF \rightarrow \Delta F$: $\Longrightarrow f(x)dx = dF$

“Forward Euler”

Bedre metode: **Euler-Cromer** 👍

En bordtennisball blir kastet på skrå mot bakken 🏓

Hva skjer *etter* at ballen treffer bakken? 🤔



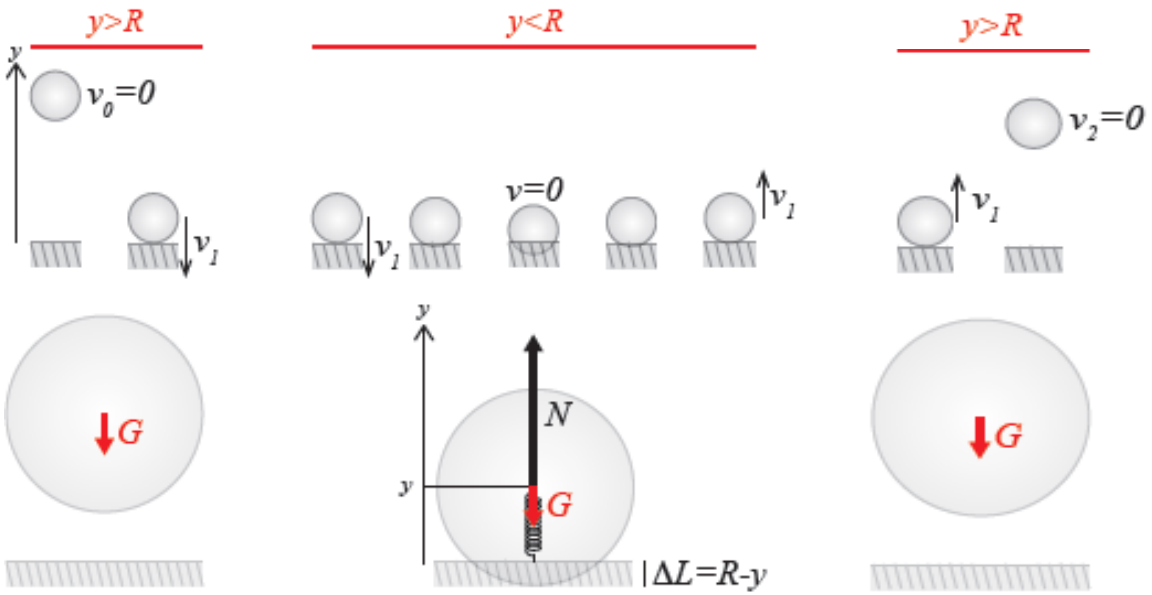
Hva skjer *etter* at ballen treffer bakken? 🤔

[✏️ lysbilde med skriblerier på iPad ✏️]

Ball som spretter fra gulvet i én dimensjon.

Normalkraften oppstår hvis:

Ballen deformeres
⇒ vi modellerer normalkraften som en fjærkraft:



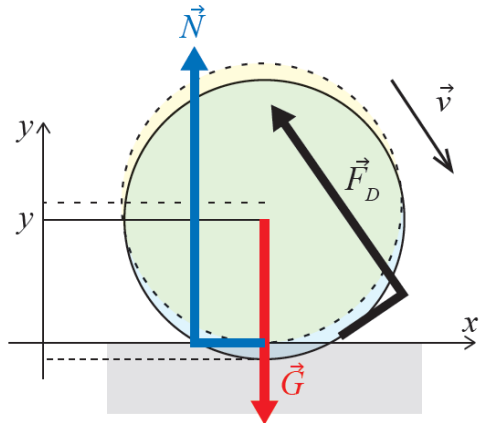
Normalkraften virker alltid vertikalt oppover *uansett* i hvilken retning ballen beveger seg.

Fellesoppgave: bordtennisball

Frilegemediagram:

Kontaktkrefter:
- luftmotstand
- normalkraft

Langtrekkende kraft:
- gravitasjon



Alle (ytre) krefter har årsak i omgivelsene og virker på objektet (=ballen)

Kraftmodeller:

luftmotstand: $\vec{F}_D = -D|\vec{v}|\vec{v}$ — fra luften på ballen, hastighetsavhengig, mot bevegelsesretning

normalkraft: $\vec{N} = \begin{cases} +k\big(R - y(t)\big)\hat{\jmath} & y < R \\ \vec{0} & y \geq R \end{cases}$ — fra gulvet på ballen, posisjonsavhengig, oppover (vinkelrett på gulvet)

gravitasjon: $\vec{G} = -mg\hat{\jmath}$ — fra jorden på ballen, konstant, nedover (mot jordens sentrum)

(Vi ser bort fra friksjon mellom ballen og gulvet.)

Fellesoppgave: bordtennisball 🏓 (numerisk)

Luftmotstand: $\vec{F}_D = -D|\vec{v}|\vec{v}$

```
F_D = -D*np.linalg.norm(v[i, :])*v[i, :]
```

Normalkraft: $\vec{N} = \begin{cases} +k\big(R - y(t)\big)\hat{\jmath} & y < R \\ \vec{0} & y \geq R \end{cases}$

```
if r[i, 1]<R:
    F_N = k*(R-r[i, 1])*np.array([0, 1])
else:
    F_N = [0, 0]
```

Gravitasjon: $\vec{G} = -mg\hat{\jmath}$

```
F_G = -m*g*np.array([0, 1])
```

N2L: $\vec{F}_{net} = \sum \vec{F}_{ext} = \vec{F}_D + \vec{N} + \vec{G} = m\vec{a}$

$$\vec{a} = \frac{\vec{F}_{net}}{m}$$

```
Fnet = F_N + F_D + F_G
a[i, :] = Fnet/m
```

Fellesoppgave: bordtennisball 🏓 (numerisk)

sprettende_ball.py

```python
# Importer nyttige Python-bibliotek
import numpy as np
import matplotlib.pyplot as plt

# Definerer konstanter og variabler vi vil trenge
# Jeg har stort sett brukt de konstantene fra pingpong.py,
# det samsvarer ikke helt med tallene vist i Matlab-koden fra forelesnings-sliden.
g = 9.81    # tyngdeakselerasjon i m/s^2
m = 0.0027  # masse til ballen i kg
v_T = 5.0   # terminalhastigheten i m/s.
D = m*g/(v_T**2)    # luftmotstandskoeffisienten i kg/m
R = 0.02    # radius til ballen i m (2 cm)
k = 100.0   # fjærkonstanten i N/m
Fnet = 0.   # netto kraft på ballen. Den skal vi regne ut seinere i ei løkke
F_D = 0.    # luftmotstanden på ballen. Den skal vi også regne ut seinere i ei løkke. Antar turbulente
F_N = 0.    # normalkraften på ballen. Den skal vi også regne ut seinere i ei løkke. Modelert som en f
F_G = 0.    # gravitasjonskraften på ballen. Den skal vi også regne ut seinere i ei løkke.
x0 = 0.     # initiell posisjon i x-retning, definert til å være i null m
y0 = 0.8    # initiell høyde (y-retning), kalt h på forelesning  i m. Vi kan teste forskjellige start-
r0 = np.array([x0, y0]) # initiell r-vektor, med sin x- og y-komponent
# Her lager vi start-hastigheten ved t0 = 0 s
v0 = np.array([1.5, -0.5])
time = 5.   # maximum tid vi ser på i sekunder
dt = 0.001  # tidssteg i sekunder - pass på at det er lite nok!

# Her bruker vi numpys ceil-funksjon for å bestemme antall elementer vi vil ha i vektorene og matrisen
# "The ceil of a scalar b is the smallest integer i such that i>= b"
# https://www.geeksforgeeks.org/numpy-ceil-python/
n = int(np.ceil(time/dt))
```


Python-script: sprettende_ball.py

Fellesoppgave: bordtennisball 🏓 (numerisk)

sprettende_ball.py

```
18  # Importer nyttige Python-bibliotek
19  import numpy as np
20  import matplotlib.pyplot as plt
21
48  # Nå definerer vi en vektor for tiden t,  og matriser for posisjonen r (x- og y-komponent ved tid t),
49  # hastigheten v (x- og y-komponent ved tid t), og akselerasjonen a (x- og y-komponent ved tid t)
50  # n rader, to kolonner:
51  # r = [x0, y0]
52  #     [x1, y1]
53  #       ...
54  #     [xn, yn]
55
56  t = np.zeros(n)
57  r = np.zeros((n,2))
58  # For å sjekke hvordan r ser ut:
59  #print(r)
60  v = np.zeros((n,2))
61  a = np.zeros((n,2))
62
63  # Nå bruker vi initialbetingelsene.
64  # Denne litt kryptiske notasjonen betyr:
65  # "0": rad 0 -> første rad
66  # ":": "slice notation", her betyr det bare at vi tar alle kolonner i rad 0
67  # som er x0, y0 for posisjonen of vx0, vy0 for hastigheten.
68  r[0, :] = r0 # Startposisjonen til r-vektoren
69  t[0] = 0    # Denne er forsåvidt satt allerede - vi har jo fylt hele vektoren med nuller da vi definert
70  v[0, :] = v0    # Initialhastighet
71  a[0, :] = 0 # initial-akselerasjon
```


Python-script: sprettende_ball.py

Fellesoppgave: bordtennisball 🏓 (numerisk)

```
# Nå er vi klare for å kjøre ei løkke og regne ut alt vi vil ha:
for i in range (0,n-1):
    # Først regner vi ut normalkraften. Husk at r[i,1] er y-komponenten til posisjonen ved rad ("tid") i
    if r[i, 1]<R:
        F_N = k*(R-r[i, 1])*np.array([0, 1]) # Enhetsvektoren i y-retning = np.array([0,1])
    else:
        F_N = [0, 0]
    F_D = -D*np.linalg.norm(v[i, :])*v[i, :]
    F_G = -m*g*np.array([0, 1]) # Enhetsvektoren i y-retning = np.array([0,1])
    Fnet = F_N + F_D + F_G  # Nettokraften. Fortegnene er satt i linjene over, der F_D pg F_G er regnet ut.
    a[i, :] = Fnet/m        # akselerasjonen
    v[i+1, :] = v[i, :] + a[i, :]*dt # hastigheten
    # Vi bruker Euler-Cromer-metoden for å beregne posisjonsvektoren
    r[i+1, :] = r[i, :] + v[i+1, :]*dt
    t[i+1] = t[i] + dt # her inkrementerer vi tiden
```

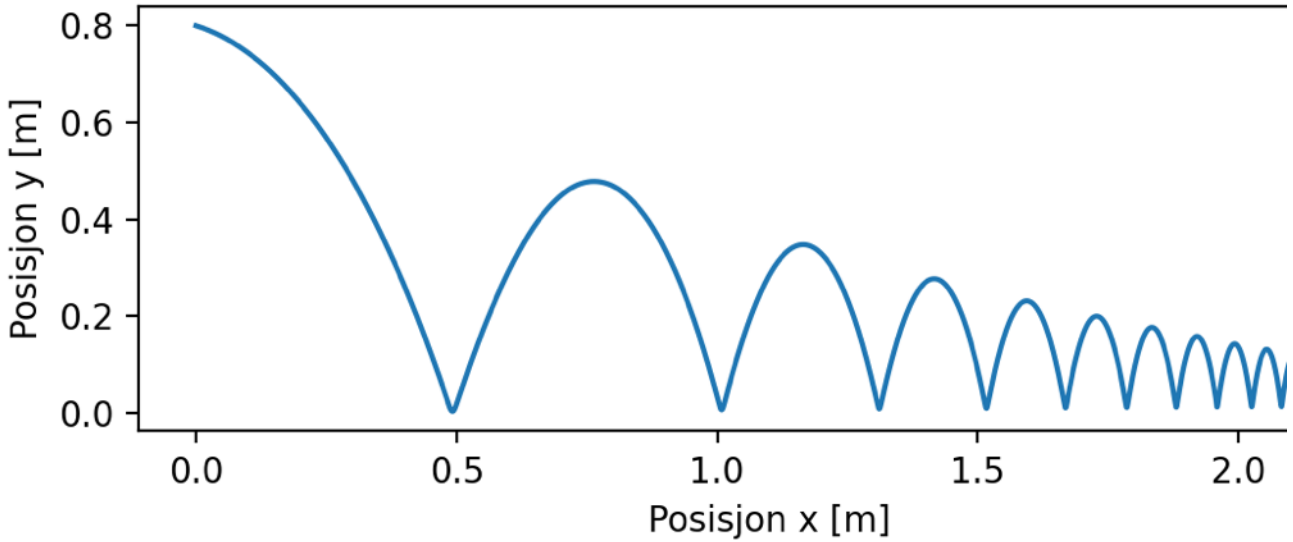
Fellesoppgave: bordtennisball 🏓 (numerisk)

```
# Nå er vi klare for å kjøre ei løkke og regne ut alt vi vil ha:
for i in ran
    # Først
    if r[i,
        F_N
    else:
        F_N
    F_D = -D
    F_G = -m
    Fnet = F
    a[i, :]
    v[i+1, :
    # Vi bru
    r[i+1, :
    t[i+1] =
```

```
# Funksjonen subplots er veldig kjekk!
# Se https://matplotlib.org/api/_as_gen/matplotlib.pyplot.subplots.html
fig, ax = plt.subplots(nrows=3, figsize=(6, 8))
# r[:i,0]: Første kolonne (0) som da er x-komponenten, for alle tider 0->i
# r[:i,1]: Andre kolonne (1) som da er y-komponenten, for alle tider 0->i
# Husk at vi har laget en matrise med n rader, en for hver tid,
# og 2 kolonner, en for x-komponenten og en for y-komponenten
ax[0].plot(r[:i, 0], r[:i, 1])
ax[0].set_xlabel("Posisjon x [m]")
ax[0].set_ylabel("Posisjon y [m]")
# Her plotter vi de to hastighetskomponentene for alle tider
ax[1].plot(t[:i], v[:i, 0])
ax[1].plot(t[:i], v[:i, 1])
ax[1].set_xlabel('Tid t [s]')
ax[1].set_ylabel('v$_x$, v$_y$ [m/s]')
#ax[1].set_title('Hastighetskomponenter v$_x$ og v$_y$')
# Putte på en boks med info om hva grafene er for noe
ax[1].legend(["v$_x$", "v$_y$"])

# Her plotter vi de to akselerasjonskomponentene for alle tider
ax[2].plot(t[:i], a[:i, 0])
ax[2].plot(t[:i], a[:i, 1])
ax[2].set_xlabel('Tid t [s]')
ax[2].set_ylabel('a$_x$, a$_y$ [m/s$^2$]')
#ax[2].set_title('Akselerasjonskomponenter a$_x$ og a$_y$')
# Putte på en boks med info om hva grafene er for noe
ax[2].legend(["a$_x$", "a$_y$"])
# tight_layout() sørger for at ting ikke blir skrevet oppå hverandre.
# tight_layout() funker kanskje ikke for eldre Python 3-versjoner - isåfall, kommenter den ut
fig.tight_layout()
plt.show()
```

Resultater, bordtennisball 🏓 (numerisk)





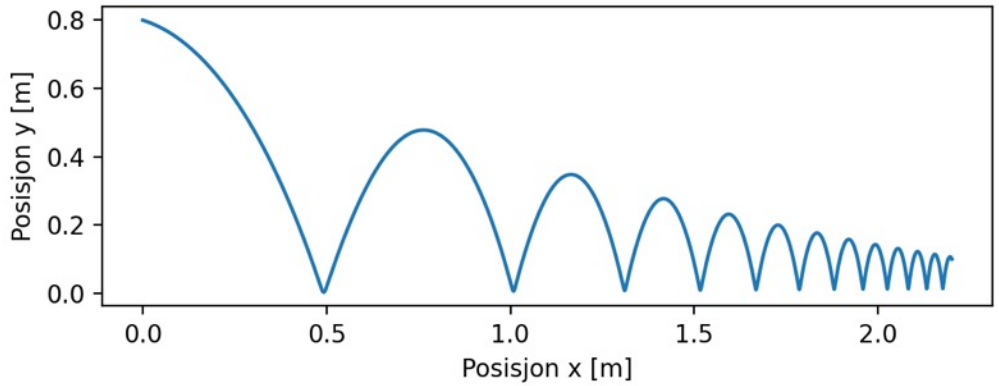
Ikke perfekt:

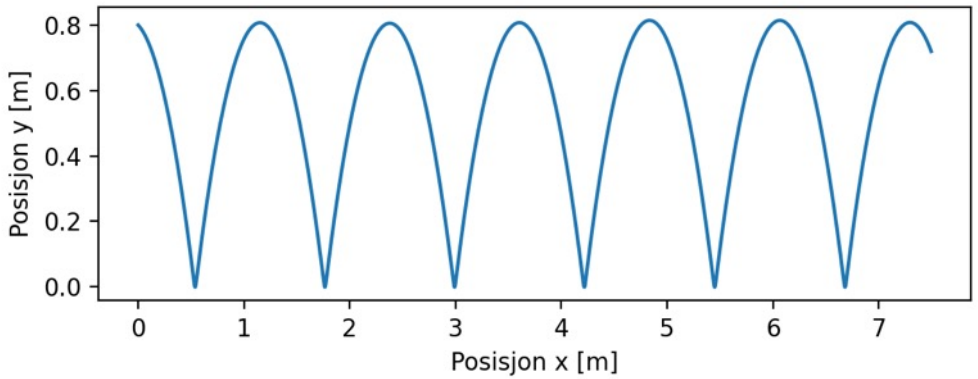
- Ingen dempning i fjærkraften
- Ingen friksjon
- (Ingen rotasjon)

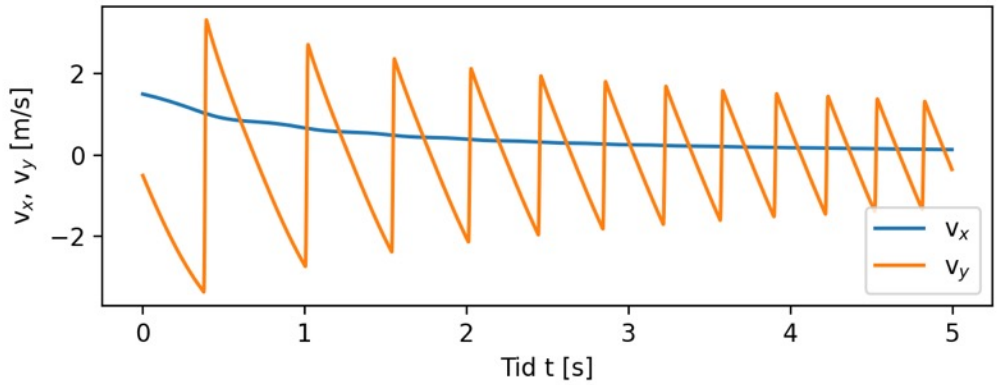
Men ikke så helt galt heller 😀

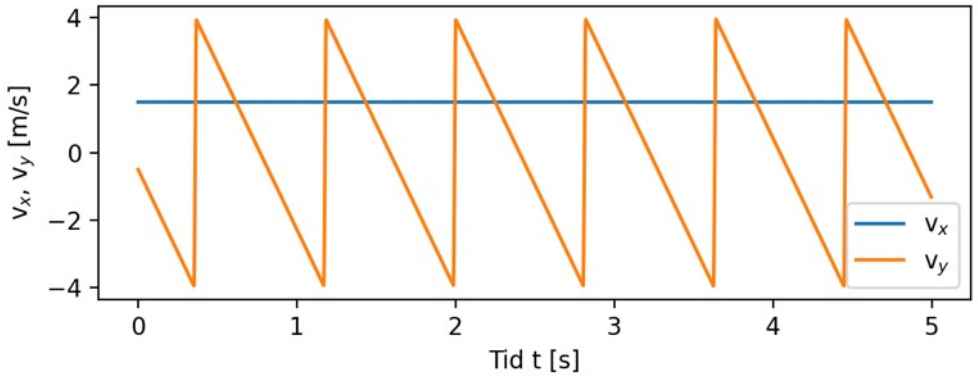
Resultater, bordtennisball 🏓 (numerisk)

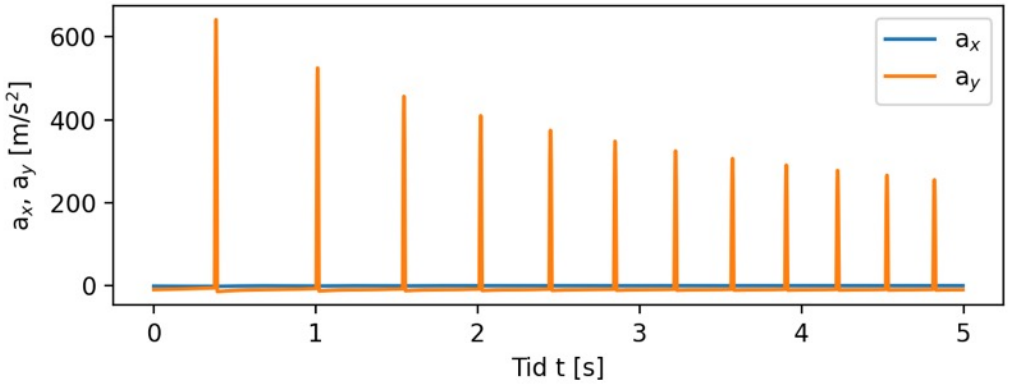
Dempet bevegelse i x- og y-retning → skyldes luftmotstanden

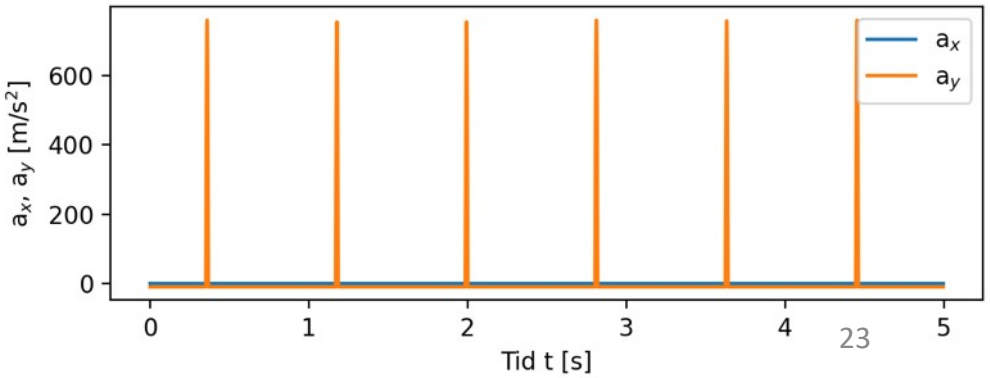




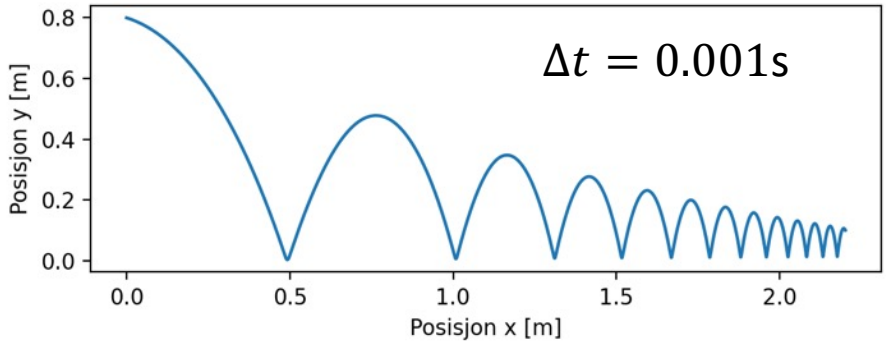


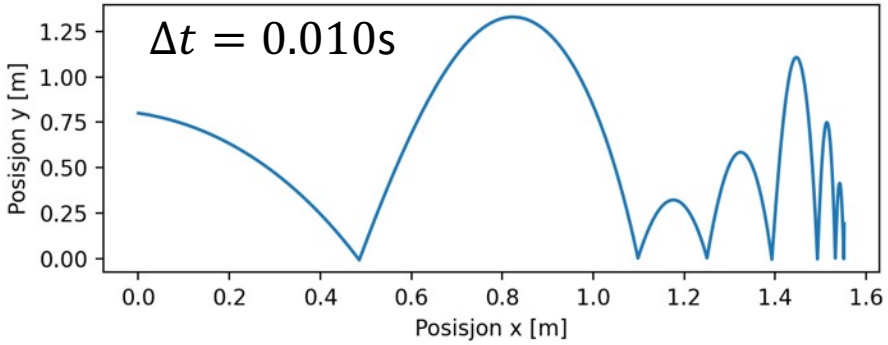


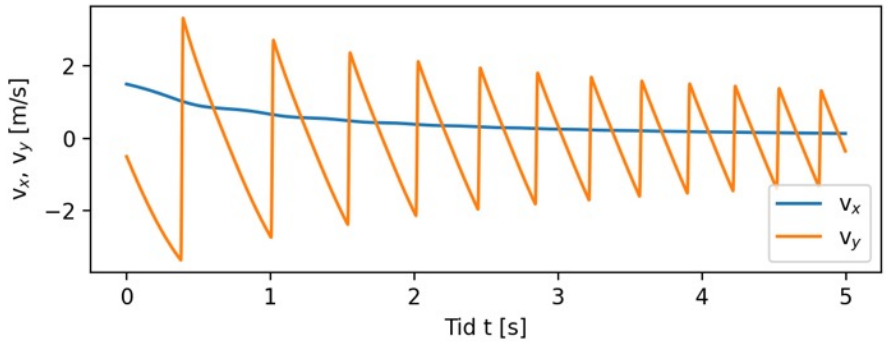


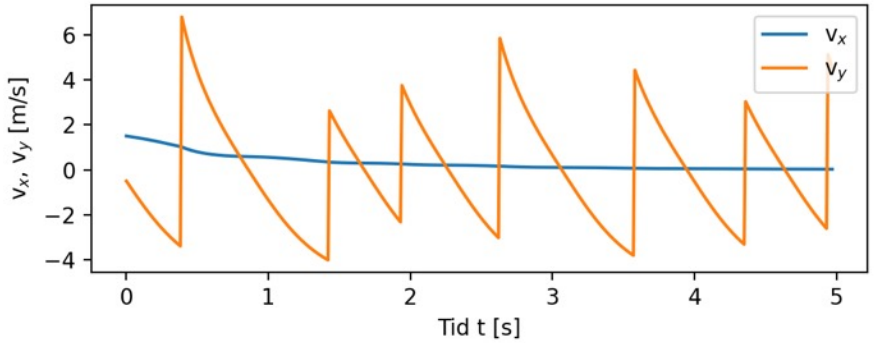


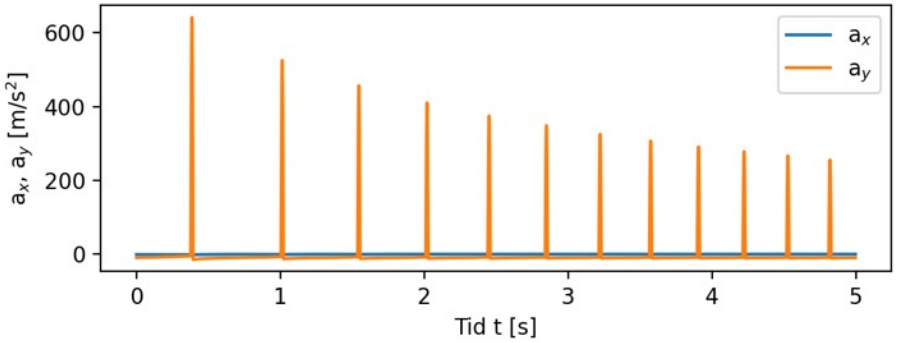
Hva med valg av størrelsen på tidsintervallet? 🤔

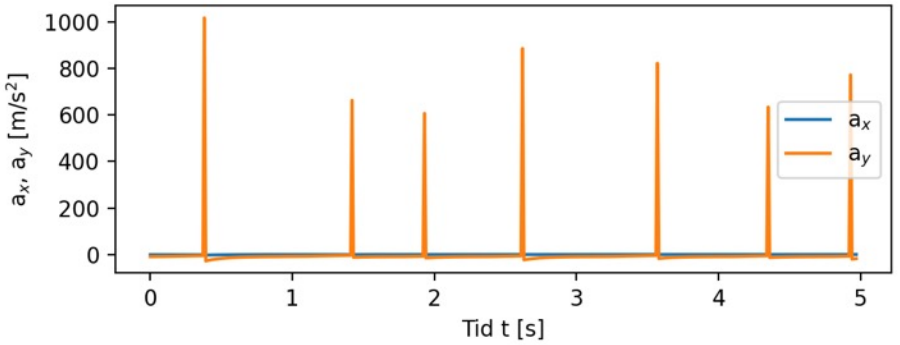












Mentimeter

www.menti.com kode XX

Hva er det som skjer? 🤔

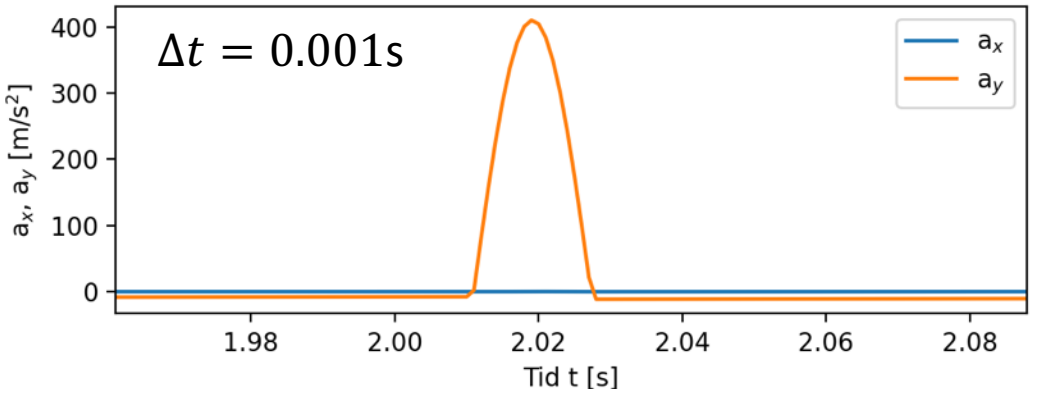
<u>Kontakt med bakken</u>:
Normalkraften er **stor** og virker i et **kort** tidsintervall.
Store forandringer i v_y *og* a_y.

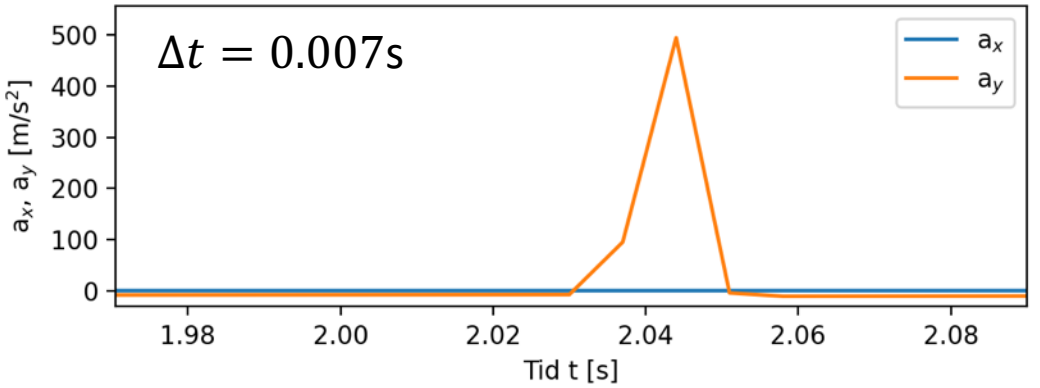
<u>Bevegelse gjennom lufta</u>:
jevn uten brå endringer (v_x,a_x).

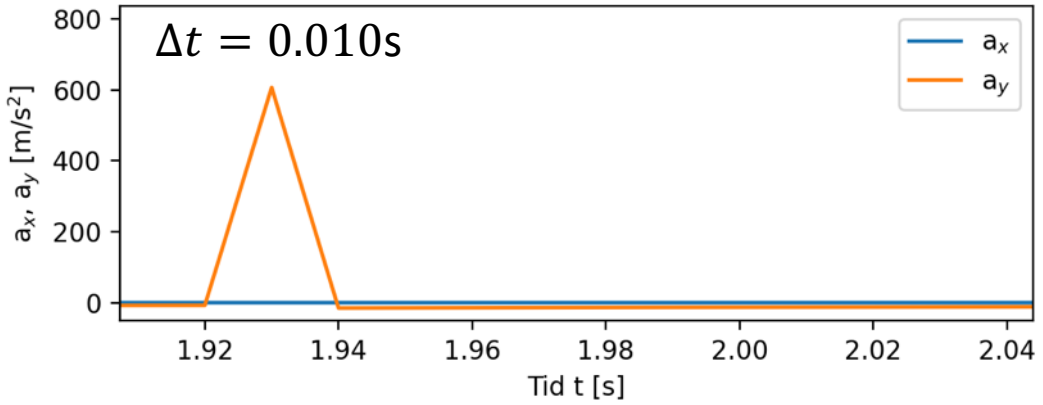
$$\vec{a}(t) = \lim_{\Delta t \to 0} \frac{\vec{v}(t+\Delta t) - \vec{v}(t)}{\Delta t}$$

For ***små*** tidsintervaller Δt:

$$\vec{a}(t) \approx \frac{\vec{v}(t+\Delta t) - \vec{v}(t)}{\Delta t}$$







Takk for i dag! 😀