Introduksjon til vitenskapelig programmering
Uke 4
Sandvika vgs

Jonas van den Brink
j.v.d.brink@fys.uio.no

March 4, 2014

Vi er nå klare for å trekke sammen alle trådene vi har sett på så langt og begynne å lage et program som løser bevegelsesligningene for fallskjermhopping og strikkhopp. Vi starter med litt repetisjon av fysikken, og så ser vi på hvordan vi skal implementere det på datamaskin.

Kreftene

Et fallskjermhopp består hovedsakelig av to deler. Den første delen er det som skjer før fallskjerm er utløst, denne delen kalles gjerne *fritt fall*. Etter dette utløser hopperen fallskjermen som da drastisk reduserer fallhastigheten. I begge faser utsettes hopperen bare for to krefer, tyngdekraft og luftmotstand. Så vi har

$$\sum F = F_G + F_D,$$

der F_G er tyngdekraften og F_D er luftmotstand. Dere har sikkert kjennskap til at tyngdekraften er gitt ved $F_G = mg$, der m er den totale massen til hopperen, som vil se personvekten og alt utstyret, og g er tyngdens akselersjonskonstant. En fallskjermhopper vil falle med veldig høy hastighet, og vi bruker derfor en kvadratisk form på luftmotstanden, som vi kan skrive som følger

$$F_D = \frac{1}{2}\rho C A v^2,$$

her er ρ tettheten til luft, C er luftmotstandskoeffisienten som avhenger av formen på objektet som faller og A er frontarealet.

Det eneste som endrer seg når fallskjerm løses ut, er frontarealet A, og luftmotstandskoeffisienten C. Ellers er fysikken helt lik.

Parametere

Tallene m, g, ρ, C, A er det som kalles fysiske parametere, og det er størrelser vi velger - vi velger dem utifra hva slags simulering vi ønsker å gjøre, men vi regner dem altså generelt sett som kjent. I vår simulering kan vi bruke følgende parametere

Fritt Fall	
m	90 kg
g	9.81 $\mathrm{m/s^2}$
ρ	1 $\mathrm{kg/m^3}$
C	1.4
A	0.7 $\mathrm{m^2}$
Under fallskjerm	
C_p	1.8
A_p	44 $\mathrm{m^2}$

Newtons 2. lov

Om vi bruker Newtons 2. lov sammen våre kraftlover får vi et uttrykk fra akselerasjonen til fallskjermhopperen. Vi har:

$$\sum F = ma,$$

setter vi inn kreftene får vi

$$F_g + F_d = ma,$$

da har vi

$$mg - \frac{1}{2}\rho C A v^2 = ma,$$

vi deler på m for å få et uttrykk for a:

$$a = g - \frac{1}{2m}\rho C A v^2.$$

Vi ser at akselerasjonen avhenger av hastigheten, så vi kan skrive den som en funksjon av v:

$$a(v) = g - \frac{1}{2m}\rho C A v^2.$$

Oppgave - Regne ut terminalhastigheten

Terminalhastigheten er den raskeste hastigheten et objekt kan falle med. Regn ut terminalhastigheten for fallskjermhopperen vi ser på.
Hint: Ved terminalhastigheten faller man med konstant hastighet, da vet vi at akselerasjonen er null. Hva betyr dette med tanke på Newtons 2. lov?

Fremgangsmåte for å finne hastigheten til hopperen

Vi vet at dersom vi har en konstant akselerasjon, og en starthastighet v_0 så kan vi regne ut hastigheten ved en hvilken som helst tid t utifra:

$$v(t) = v_0 + at, \qquad \text{for konstant } a.$$

Så hva gjør vi når vi *ikke* har en konstant akselerasjon? I vårt tilfelle vil akselerasjonen endre seg over tid, fordi hastigheten endrer seg over tid. Men over en veldig kort tid Δt, så kan vi si at den er så godt som konstant, vi kan da si at

$$v_1 = v_0 + a(v_0)\Delta t.$$

Der v_0 er starthastigheten, v_1 er hastigheten etter ett *tidssteg*, altsa ved tiden $t_1 = \Delta t$. Vi skriver $a(t_0)$ for å tydliggjøre at vi bruker akselerasjonen som gjelder ved starttiden. Siden vi nå kjenner v_1, så kan vi regne ut $a(v_1)$ fra uttrykket vårt for akselerasjonen, og bruke denne til å gå et tidssteg til:

$$v_2 = v_1 + a(v_1)\Delta t.$$

Og slik kan vi fortsette, vi kan alltid regne oss ett tidssteg frem, ved å bruke resultatet fra forrige tidssteg

$$v_{n+1} = v_n + a(v_n)\Delta t,$$

for $n = 0, 1, 2, 3, \ldots$.

Slik fortsetter vi til vi har simulert nok tidssteg til å nå en slutttid vi har valgt på forhånd. Vi ønsker gjerne at Δt skal være minst mulig, slik at vi får et mer nøyaktig resultat. For eksempel kan vi bruke $\Delta t = 0.01$ s i vår simulering, altså tar vi skritt på ett hundredelssekund frem i tid av gangen. I et vanlig fallskjermhopp er hopperen ca. 1 minutt i fritt fall, så vi lar sluttiden være $T = 60$ sekunder. Det betyr at vi trenger å ta totalt

$$n = \frac{T}{dt} = \frac{60 \text{ s}}{0.01 \text{ s}} = 6000,$$

tidssteg.

På tide å programmere

Vi er nå klare for å sette igang, her er malen på programmet dere kommer til å skrive:

1 Importer pylab, det er alt vi kommer til å trenge.

2 Skriv inn alle parameterene vi trenger, det vil si m, g, ρ, A, C, A_p, C_p, v_0.

3 Definer akselerasjonen som en funksjon av hastigheten.
Hint: `def a(v):`, og husk å returnere noe!

4 Definer $\Delta t = 0.01$ (Hint: kall variablen `dt` i programmet ditt), $T = 60$ og $n = T/dt$

5 Opprett to *arrays*, ett for hastigheten v og et for tiden t. Vi vil at de skal være tomme, og ha plass til $n+1$ elementer, så bruk `zeros` kommandoen. Merk at nå vil `v[i]` i programmet ditt svare til v_i i matematikken.

6 Lag en `for`-løkke som går over $i = 0, 1, 2, \ldots, n$. (Hint, bruk `range`.)

7 Inne i løkka, regn ut $v[i+1]$ fra $v[i]$ ved å bruke formelen vi har funnet. Oppdater også tiden (Hint: `t[i+1] = t[i] + dt`).

8 Plot resultatet for å sjekke at alt har blitt gjort riktig (Hint: `plot(t,v)`).

Oppgaver

Etter at du har fått programmet ditt til å funke kan du besvare følgende oppgaver

a) Pynt på plottet ved å lagge til navn på aksene (`xlabel` og `ylabel`), et grid (`grid()`) og eventuelt tittel og lignende.

b) Hvor lang tid tar det før falleren har nådd tilnærmet terminalhastighet? Les av plottet.

c) Skriv ut terminalhastigheten hopperen får i programmet ditt. Hint: Det er en funksjon `max`, som henter ut det største elementet i et array. Sammenlign denne med terminalhastigheten du fant med penn og papir tidligere, hvor like blir verdiene? Ser det ut som programmet ditt regner riktig?

Videre

Nå har vi laget et program, som finner hastigheten til hopperen under fritt fall med luftmotstand. Men vi må fortsatt legge til at fallskjerm utløses. Vi kommer til å se på dette i fellesskap neste gang, men de av dere som har fått til alt så langt, kan begynne å bryne dere på hvordan vi skal gjøre dette. Grunnidéen er ihvertfall denne: det er bare frontarealet A og luftmotstandskoeffisienten C som endrer seg når fallskjerm løses ut, så hvis vi greier å endre disse verdiene i programmet vårt til riktig tidspunkt, så simulerer vi effektivt at fallskjerm er løst ut. I løkka vår, så har vi tiden t_i, så kanskje vi greier å bruke en `if`-test til å endre A og C til riktig tid?

Neste gang kommer vi også til å regne ut og plotte g-kreftene fallskjermhopperen opplever. Vi kommer også til å finne hastigheten til en strikkhopper på samme måte som vi har gjort for fallskjermhopperen idag.