Pilotprosjekt i faglig bruk av IKT i sivilingeniørutdannelsen:

«Numerisk fysikk»

- **Emnet/emnene «Fysikk»: Et av de store «fellesemnene» på Gløshaugen.**
- **Tas av (nesten) alle siving-studentene.**
- **Emner som vil inngå i prosjektet:**
  - **TFY4104 – Fysikk for MTPROD, MTMART, MTIØT (høst)**
  - **TFY4106 – Fysikk for MTBYGG, MTING (høst)**
  - **TFY4108 – Fysikk for MTENERG (høst) – nytt høsten 2012**
  - **TFY4115 – Fysikk for MTEL, MTTK, MTNANO (høst)**
  - **TFY4120 – Fysikk for MTKJ, MTMT (høst)**
  - **TFY4102 – Fysikk for MTDESIG, MTPETR, MTTEKGEO (vår)**
  - **TFY4125 – Fysikk for MTDT, MTIØT, MTKOM (vår)**
- **Målsetting: Gi studentene økt fortrolighet med programmering og numeriske metoder.**
- **Gjennomføring: Gå gjennom øvingsopplegget i hvert av kursene og innarbeide bruk av enkle numeriske oppgaver basert på Matlab, der dette passer. Synliggjøre en «beregningskomponent» i emnet.**
- **Pengebruk: En ekstra «beregnings-studass» i hvert emne.**
- **Egeninnsats IFY: Allokering av to lærerressurser høsten 2012, 1.aman. Peter Berg og Professor Alex Hansen (prosjektleder); begge med Numerisk fysikk som forskningsfelt.**
- **Evaluering: Spesifikke spørsmål i den rutinemessige studentevalueringen (som gjøres i alle emner ved instituttet).**
- **Relatert aktivitet: Institusjonsforankret strategisk universitetsprogram (ISP), *Multiscale Physics on the Computer: A Norwegian Network*, med deltagere ved Institutt for fysikk (NTNU), Fysisk institutt (UiO) og Institutt for matematiske realfag og teknologi (UMB).**

Institutt for Fysikk, NTNU
TFY4106 Fysikk for MTBYGG, MTING høsten 2011
Innleveringsfrist: Torsdag 3. november kl. 16.00.

MATLAB-øving: Skrått kast med luftmotstand og skru.

Innledning

I denne øvingen skal vi regne på et skrått kast med luftmotstand og skru. Luftmotstand er noe man er vant til å neglisjere, fordi beregningene blir vanskelige å utføre med penn og papir. Derfor skal vi bruke en numerisk metode og Matlab i denne øvingen.

Vi presenterer først fysikken bak luftmotstand og skru. Så presenteres den numeriske metoden som skal brukes, og hvordan dette gjøres enklest mulig i Matlab. Dersom du har kontroll på numerikk eller Matlab kan du skumlese disse avsnittene, da vi er svært grundige her. Det er oppgitt et Matlab- skript på side 3 som du kan ta utgangspunkt i. Det ligger også to matlab-filer på nettsidene (oppg1_utdelt.m og oppg2_utdelt.m), som du gjerne kan laste ned med en gang.

Fysikken bak

For å regne på luftmotstand, må vi først ha en fysisk modell for hvordan luftmotstanden virker. En rimelig antagelse er å si at luftmotstanden er proporsjonal med kvadratet av farten, og motsatt rettet av farten, altså:

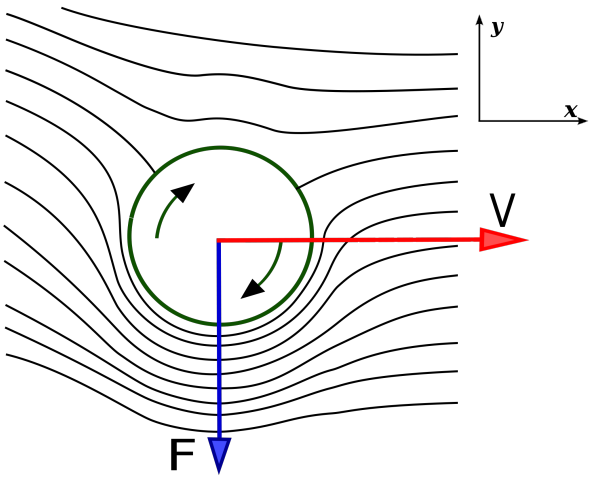
$$\vec{F}_l = -\hat{v} K v^2,$$

der K er en empirisk bestemt konstant.

Alle som har spilt fotball, vet at det er livsviktig å kunne skru ballen. Den fysiske forklaringen på at man kan skru en ball kalles Magnus-effekten, etter tyskeren Heinrich Magnus som beskrev den i 1852. Kraften er gitt ved

$$\vec{F_m} = S(\vec{\omega} \times \vec{v}),$$

der S er en konstant som avhenger av konstanten K i forrige likning. Vi antar her at legemet som kastes er en kule. Hvorfor denne kraften oppstår, kan ses fra figur 1.



Figur 1: Magnus-effektens opphav og retning på et legeme som spinner. Et koordinatsystem vises øverst til høyre, her går z-aksen ut av papirplanet.[1]

Vi ser med en gang analogien til en flyvinge: på den siden der luften får "hjelp" av rotasjonen til å bevege seg fortere, blir trykket lavere (Bernoullis lov). Dermed vil kraften virke i denne retningen. De som er veldig glad i fluidmekanikk, vil kanskje en gang lære at dette kommer fra Kutta-Joukowskis teorem.

Dersom du er ukomfortabel med $\vec{\omega}$, er det ingen grunn til bekymring, det hele er meget enkelt. Størrelsen på $\vec{\omega}$ er som vanlig 2π delt på omløps-perioden, og retningen bestemmes av høyrehåndsregelen: la fingrene på høyre hånd dreie rundt med rotasjonen, da vil tommelen peke i $\vec{\omega}$ sin retning.

[1] Figur av Bartosz Kosiorek, lisensiert under Creative Commons 3.0. Original utgave finnes her.

Numerikk

Siden akselerasjonen er proporsjonal med kraften, $\vec{a} = \vec{F}/m$, og vi har innført en hastighetsavhengig kraft, vil dette si at akselerasjonen også er hastighetsavhengig. Vi har dermed differensiallikninger for posisjon og hastighet som må løses med numeriske metoder. Den enkleste måten å gjøre dette på er Eulers metode, som er en veldig naturlig metode: ved tiden t_0 har vi initialbetingelsene (her bare for luftmotstand):

$$\vec{r} = \vec{r}_0,$$

$$\vec{v} = \vec{v}_0,$$

$$\vec{a} = -g\hat{z} - \hat{v}\frac{Kv^2}{m}.$$

Her peker z-aksen oppover, slik at tyngdekraften virker i negativ z-retning. Hvis vi nå går et lite skritt fremover i tid, til $t' = t_0 + \Delta t$, så er det naturlig å si at

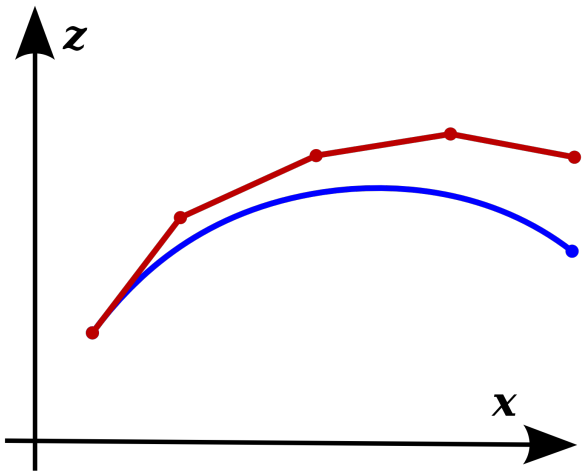
$$\vec{r'} = \vec{r}_0 + \vec{v}_0\Delta t,$$

$$\vec{v'} = \vec{v}_0 + \vec{a}\Delta t,$$

$$\vec{a'} = -g\hat{z} - \hat{v'}\frac{Kv'^2}{m}.$$

Det vi antar her er at hastigheten (og akselerasjonen) endrer seg lite i løpet av Δt, så vi bruker verdiene for hastighet og akselerasjon som vi har fra tiden t_0. Men vi oppdaterer hastigheten og akselerasjonen, slik at i neste skritt bruker vi disse nye verdiene som en god tilnærming. Merk at det står v' i den siste likningen for akselerasjonen! Vi bruker altså den nyeste verdien for hastigheten her.

Til høyre ses en illustrasjon av hvordan dette kan se ut for posisjonen når vi gjør 4 tids-steg. Dette ser kanskje ikke ut som en god tilnærming, men dersom vi gjør skrittlengden kortere og kortere, vil tilnærmingen ganske fort bli veldig bra.



Figur 2: Den nederste linjen er den virkelige posisjonen, mens den øverste er tilnærmingen. Merk forskjellen med koordinatsystemet fra forrige figur: her ser vi kastet fra siden, i sted så vi kastet ovenfra.

Programmering

Når vi skal skrive et skript i Matlab som skal løse dette iterativt, må vi først definere initialbetingelsene, og sette inn tallverdier. Vi må også bestemme størrelsen på Δt. Vi trenger kun 2 koordinater x og z når vi bare ser på luftmotstand, siden legemet beveger seg i en rett linje sett ovenfra.

Så må vi finne et stopp-kriterium for løkka som skal kjøre. Vi kan velge å bruke en "for"-løkke som skal kjøre så den finner f.eks 100 punkter, eller vi kan bruke en "while"-løkke som kan kjøre til et bestemt punkt nås. Uansett er det størrelsen på Δt som bestemmer hvor nøyaktig resultatet blir, jo mindre jo bedre. Men hvis den blir altfor liten, vil programmet vårt bruke veldig lang tid. Altså må vi velge et kompromiss.

Når vi skal plotte punktene datamaskinen finner for oss, velger vi å lagre x- og z-verdiene for alle stegene i to separate vektorer lagraX og lagraZ, slik at $[lagraX(i), lagraZ(i)]$ vil være en vektor der vi kan variere i for å finne posisjonen ved forskjellige tidspunkter. For å gjøre det, legger vi inn verdier i lagraX og lagraZ underveis i løkka. I Matlab skriver vi da `lagraX=[lagraX, x]` for å "fylle på" med den nye verdien x bakerst i lagraX.

Her kommer det utlovede eksempel-skriptet, der vi endrer akselerasjonen i z-retning med 0.07 for hvert tidssteg. Dette kan du bruke som grunnlag når du skal løse oppgaven, og det ligger som en Matlab-fil på nettsidene så du skal slippe å kopiere og lime inn herifra.

```
r=[0,20];   %Dette er startverdien for r=[x,z]
lagraX=[r(1)];   %Startverdien for x = r(1) lagres i lagraX
lagraZ=[r(2)];
deltat=0.01;   %En ganske fornuftig verdi for deltat
v=[5,2];   %Dette er utgangshastigheten.
a=[0,-5];   %Dette er startverdien for akselerasjonen
ztopp = 0;   %Denne skal lagre maksimal z


while (r(2)>=0)     %Vi kjorer helt til vi treffer bakken
    r=r+v*deltat;    %Her endrer vi r-verdien som tidligere forklart.
    v=v+a*deltat;    %Her endrer vi v likedan.
    a=[a(1), a(2) - 0.07];   %Her endres kun z-verdien av akselerasjonen.
    lagraX=[lagraX, r(1)];   %Legger til den nye x-verdien til lagraX-vektoren.
    lagraZ=[lagraZ, r(2)];
    if (v(2) > 0)
        ztopp = r(2);  %Mens farten i z-retning er positiv, oppdaterer vi ztopp.
    end
end

plot(lagraX, lagraZ) %Plotter punktene vi har funnet, og viser grafen.
disp(x)   %Skriver ut x-verdien for punktet der objektet lander.
```

oppg1_utdelt.m

Oppgave 1

Vi skal undersøke et skrått kast fra en høyde på 5.0 m. Utgangshastigheten skal være 4.0 m/s og ha retning 60° i forhold til horisontalplanet. Vi beskriver luftmotstanden med $K = 2.0$ kg/m. Massen til objektet som kastes er 3.0 kg.

a) Plott banen til kastet. HUSK: gravitasjonskraft

b) Undersøk hva farten er i x- og z-retning rett før den treffer bakken.

c) Hva er farten når objektet er på sitt høyeste punkt i banen?

d) Prøv å sette $K = 0$. Hvilken bane får du nå? Hva er den fysiske tolkningen?

Oppgave 2

Vi skal nå legge til Magnus-effekten. Det vil si at du må legge til et ledd i uttrykket for akselerasjonen. Anta at $\vec{\omega}$ er konstant. Husk å dele på massen for å få akselerasjon! Her vil den observante straks bemerke at vi trenger 3 koordinater, så du kan ta utgangspunkt i den andre Matlab-filen som ligger på nettsidene (oppg2_utdelt.m). Les også kommentarene i denne filen.

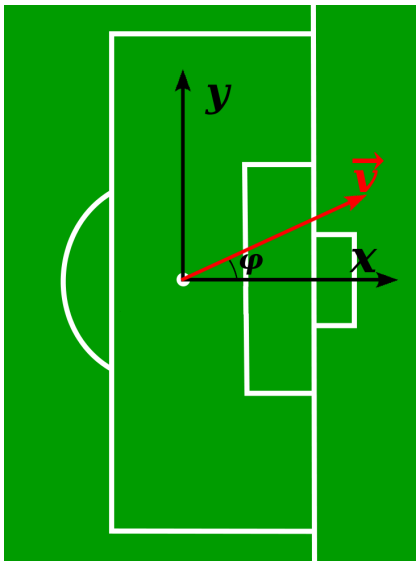
Det anbefales at du skriver akselerasjonen på vektorform i Matlab-koden, slik som posisjonen og hastigheten, da dette er mye enklere. I Matlab skriver vi `c = cross(a,b)` for å få kryssproduktet mellom a og b. Det er noen linjer i Matlab-filen som er kommentert ut. Disse tegner et fotball-mål og skal brukes i del b).

a) Plott banen til kastet med følgende verdier for de ulike størrelsene som inngår: Startposisjon: $(x, y, z) = (0, 0, 0)$. Starthastighet: $v_x = v_0 \cos 60°$, $v_y = 0$, $v_z = v_0 \sin 60°$, med $v_0 = 4$ m/s. $K = 2.0$ kg/m, $S = 0.2$ kg, $m = 3.0$ kg, $\vec{\omega} = -8\pi\hat{z} \cdot \mathrm{s}^{-1}$. Ser dette fornuftig ut, dvs. skrur kastet riktig vei? (HINT: retningen på kraften kan finnes med høyrehåndsregelen.)

b) Vi skal så regne på et straffespark i fotball. Da trenger vi litt data:

- Et straffespark tas fra 11.0 m, og en fotball veier 450 g.
- Se figuren under for koordinatsystem, og definisjon av vinkelen ϕ, som skal være slik at utgangshastigheten peker mot midten av målet.
- Vinkelen θ i forhold til horisontalplanet er på 18.3°, som gjør at ballen går like under tverrliggeren.
- Utgangshastigheten for en fotballproff er 80 km/t for et skudd som skal skrus mye.
- Enhetsvektoren i dette koordinatsystemet er $[\cos\phi\,\cos\theta,\ \sin\phi\,\cos\theta,\ \sin\theta]$.
- Realistiske verdier er $K = 4.655{\cdot}10^{-3}$, $S = 4.190{\cdot}10^{-3}$. [1]

Eksperimenter med forskjellige verdier for $\vec{\omega}$, og se om du klarer å skru ballen i venstre kryss. Er denne verdien for $\vec{\omega}$ realistisk? (HINT: her kan du bruke den koden som tegner et fotballmål!)



Figur 3: Her vises koordinatsystemet, og en utgangshastighet $\vec{v}$ er lagt inn for å vise vinkelen ϕ.

[1] Se f.eks. Reilly, Cabri, Araújo: "Proceedings of the Fifth World Congress on Science and Football", Routledge (2005)