Oblig 1 FYS-MEK/F 1110 - Februar 2006

Hva det hele går ut på

I oblig 1 i år blir oppgaven å utforske, ved numeriske beregninger, ulike fysiske parametre som karakteriserer hvordan vanndråper i regn og hagl faller i luft. Hele spekteret av dråpe- og haglstørrelser skal undersøkes. Det skal brukes en modell for luftmotstand som håndtere såvel lav, midlere som høy fart på dråpene, og resultatet fra de numeriske beregningene må sjekkes mot analytisk resultat der dette lar seg gjøre. En rapport over resultatene må skrives, og dataprogrammet man har laget for å løse oppgaven (fortrinnsvis i Matlab) skal legges ved. Rapporten kan leveres elektronisk eller i papirformat, men MÅ leveres etter angitte retningslinjer!

Litt mer presist er oppgaven som følger:

1. Lag et dataprogram (fortrinnsvis i Matlab) som kan beregne hvordan hastighet og posisjon til vertikalt fallende vanndråper eller hagl i luft varierer med tiden. Luftmotstanden skal håndteres som angitt i "Notat om luftmotstand" for hastigheter som svarer til Reynoldstall som er innenfor figur 1 i angitte notat. For lavere hastigheter skal man bruke en luftmotstand som er proporsjonal med hastigheten i første potens. Man må tilpasse disse beskrivelsene til hverandre slik at de går sømløst sammen for Reynoldstall 0.1. Man kan anta at vi har sirkulære vanndråper (eller sirkulære hagl). Velg startposisjon og starthastighet begge lik null. (Nyttige detaljer er gitt nedenfor.)

2. Beregninger skal så gjennomføres for seks dråpestørrelser og fem haglstørrelser plukket ut fra tabell 1 og 2 nedenfor. Start gjerne med en dråpediameter 1.0 mm, og følg bevegelsen i 3 sekunder (falltid) med tidssteg lik 1/2000 av falltiden. Arbeid deg deretter mot litt større diameter, og juster falltiden til du får et velegnet plott av hastighet vs tid. Arbeid deg deretter fra 1.0 mm dråpen igjen mot mindre diameter, og juster falltiden til du får gode plot. Bestem følgende for hver av diametrene: a) Terminal-hastigheten, b) Hvor lang tid det tar før man har kommet opp i 90 % av terminalhastigheten, c) Hvor lang tid vil dråpen/haglet bruke på å falle 10 m etter at terminalhastigheten er oppnådd, d) Reynoldstallet ca ved terminalhastigheten, og d) Hvor stor kinetisk energi har partikkelen ved terminalhastigheten.

3. Velg og skriv ut to plot som viser tidsutviklingen av hastigheten for en middels stor dråpe og for et hagl omtrent midt på skalaen i tabell 3 nedenfor. Velg en falltid slik at du får fram interessante trekk ved kurvene. Skriv ut ett plott som viser høyde vs tid for fallbevegelsen for en eller annen dråpestørrelse. Angi selvfølgelig dråpestørrelsen på plottene.

4. Sammenlign dine numeriske resultater med analytiske resultater (som er gjennomgått på forelesning og som står i læreboka) for én dråpestørrelse som egner seg for en slik test.

5. Ut fra de beregnede Reynoldstallene, og figur 1 i Notat om luftmotstand, ber vi deg vurdere hvorvidt man trenger en såpass komplisert beskrivelse av luftmotstanden som vi har lagt opp til i denne oppgeven, eller om man godt kunne ha nøyd seg med en enklere beskrivelse.

6. Skriv en rapport der du gir utledningen som du bruker for å sy sammen de to modellene for friksjonskraften (luftmotstanden) nevnt i punkt 1 ovenfor. Rapporten skal også inneholde resultater fra beregningene sammen med kommentarer og vurderinger av disse. Dataprogrammet skal være vedlegg til rapporten. Skriv gjerne også litt om hva har du lært av arbeidet med obligen.

7. Rapporten kan leveres enten i elektronisk form eller i papirformat. Leveres den elektronisk, sender du den til Siw Bruer, e-mail: **s.m.m.bruer@fys.uio.no**. Siw sender kvittering via e-mail om at oppgaven er mottatt, og sender den videre til gruppelærer. Dersom du leverer i papirform, henvender du deg til Siw på ekspedisjonskontoret til Fysisk institutt. Hun krysser av på en liste og leverer besvarelsen videre til riktig gruppelærer. Uansett hvilken innleveringsform man velger **er meget viktig at hver enkelt student skriver fullt navn på besvarelsen sin og at man i tillegg forteller hvilken gruppe man tilhører (!!!). For de som leverer elektronisk må begge disse opplysningene også stå i selve følgemailen som sendes Siw. Leveringsfrist fredag 17. februar kl 1500. Kontakt kursansvarlige før denne fristen dersom man trenger litt utsettelse, f.eks. ved sykdom.**

8. De som ønsker en ekstra utfordring kan forsøke å bestemme hvilket område tidsstegene kan varieres over uten at resultatet endrer seg nevneverdig. I så fall må man tenke nøye gjennom hvor i tidsforløpet testen bør gjennomføres. Men ikke bruk mye tid på dette, for ved svært små tidssteg øker regnetiden fort til timer.... (For passe tidsteg bør ikke hver utregning ta mer enn få sekunder.)

9. En annen utfordring kan nevnes for dem som ønsker slike: Forsøk å bruke Eulers midtpunktmetode eller fjerde ordens Runge-Kutta *i tillegg til* den vanlige Eulers metode vi har lagt opp til her. Se omtalen på side 5. Men *ikke* begynn på slike ekstra utfordringer før du har *hele* standardversjonen av obligen på plass, inklusiv rapportskrivingen!

På de neste sidene følger en del fakta og tips som kan være nyttige ved løsning av obligen:

Dråpe- og haglstørresler

Det er store variasjoner i størrelser av vanndråper i luft. I “Weather Almanac for July 2000” (http://www.islandnet.com/~ess/weather/almanac/arc2000/alm00jul.htm) presenterer Keith C. Heidorn en fin oversikt som jeg gjengir noen tall fra her:

Tabell 1: Vanndråpestørresler i luft

Typisk dråpe-type	Typisk diameter (mm)
Dråper i skyer	0.0012
Store dråper i skyer	0.1
Tåke-dråper	0.5
Dråper i lett regn (drizzle)	1.2
Vanlige regndråper	3.0
Store regndråper	6.0

Det er interessant å merke seg at regndråper ikke blir stort større enn 6 mm i diameter. Dråper vokser fra små til store etter som regn faller ned mot jorda, men blir størrelsen over en kritisk grense, blir dråpen ustabil og splittes opp igjen. Ellers poengterer Heidorn at det ikke er slik at alle dråper er like store. Spesielt i regnvær finnes det et utall av ulike størrelser til stede samtidig.

Hagl kan også komme i ulike størrelser. TORRO (Tornado and Storm Research Organization) gir på en av deres websider (http://www.torro.org.uk/TORRO/severeweather/hailscale.php) en oversikt over haglstørrelser som forekommer, og jeg gjengir noen tall derfra:

Tabell 2: Haglstørresler i luft

Størrelseskode	Størrelsesbeskrivelse	Typisk diameter (mm)
1	Ert	5 - 10
2	Hasselnøtt	11 - 15
3	Klinkekule, kirsebær	16 - 20
4	Valnøtt	21 - 30
5	Golfball, bordtennisball	31 - 45
6	Hønseegg, billiardball	46 - 60
7	Tennisball, stort eple	61 - 80
8	Grapefrukt, stor appelsin	81 - 100
9	Melon	101 - 125
10	Kokusnøtt	< 125

Modellering av bevegelsen i luft

Vi ønsker å foreta numeriske beregninger av bevegelsen til dråpene og haglene i luft, og må da bruke beskrivelsen gitt i "Notat om luftmotstand" utlagt på kurswebsidene 6. februar 2006 for alle hastigheter som svarer til Reynoldstall innenfor figur 1 i dette notatet. Den matematiske beskrivelsen av luftmotstandskraften F_m er:

$$F_m = \frac{1}{2}\rho S C_D v^2 \qquad (1)$$

Hvor ρ er tettheten til luft (1.293 kg/m^3 ved bakken ved 'normalt' trykk og temperatur), S er tverrsnittet av dråpen/haglet ("arealet av legemets projeksjon ned på et plan normalt på bevegelsen"), og v er legemets fart i forhold til lufta den farer gjennom. C_D er en luftmotstandskoeffisient som er avhengig av farten til legemet relativt til lufta, og til legemets form. For glatte kuler varierer C_D som angitt i figur 1 i notatet om luftmotstanden.

For hastigheter som er så små at de svarer til et Reynoldstall mindre enn 0.1, bør man bruke følgende uttrykk for luftmotstanden:

$$F_m = kv \qquad (2)$$

Du må selv finne et uttrykk for konstanten k slik at luftmotstanden blir identisk ved Reynoldstall 0.1 enten man bruker ligning 1 eller 2. Husk at Reynoldstallet er gitt ved:

$$R = \frac{\rho d}{\eta} v$$

som angitt i notatet om luftmotstand.

Vi vil overalt gjøre den antakelsen at dråpene er omtrent kuleformet. Dette er en god tilnærming for små dråper, siden overflatehinna da sørger for at dråpen faktisk er temmelig sfærisk. For store dråper får man nok mer "dråpeform", men samtidig må vi huske at store dråper ikke har en statisk form, fordi formen hele tiden endrer seg (faktisk oscillerer iblant) som følge av kollisjoner med andre dråper.

Om å integrere opp Newtons 2. lov

Newtons 2. lov sier at akselerasjonen til et system er proporsjonal med summen av alle krefter som virker på systemet. Proporsjonalitetskonstanten er den iverse av massen til systemet. Med andre ord:

$$\boldsymbol{a} = \frac{1}{m}\sum \boldsymbol{F}$$

Her betegner fete bokstaver at vi har med vektorer å gjøre.

Kjenner vi til enhver tid kreftene som virker på et system, og kjenner vi initialbetingelsene, kan vi da i prinsippet entydig integrere opp denne likningen for å finne ut hvordan hastighet og posisjon varierer med tiden.

Det er likevel et problem. Dersom kreftene endrer seg med tiden, er det ikke noe enkelt uttrykk som forbinder hastighet og posisjon med akselerasjonen (og kreftene). Vi må da ofte gjennomføre integrasjonen numerisk. Vi gjør da ofte det trikset at vi bruker to første ordens, koblede differentialligninger i stedet for å ha én annen ordens ligning. Det vil si at vi skriver:

$$\frac{d\boldsymbol{v}}{dt} = \boldsymbol{a} = \frac{1}{m}\sum \boldsymbol{F} \qquad \text{og}$$

$$\frac{d\boldsymbol{r}}{dt} = \boldsymbol{v}$$

Dersom vi nå lar dt få en endelig (men liten) størrelse, går disse ligningene over til:

$$\Delta\boldsymbol{v} = \boldsymbol{a}\Delta t = \left(\frac{1}{m}\sum \boldsymbol{F}\right)\Delta t \qquad \text{og}$$

$$\Delta\boldsymbol{r} = \boldsymbol{v}\Delta t$$

Dette kan lett omsettes til to rekursjonsformler:

$$\boldsymbol{v}(t+\Delta t) = \boldsymbol{v}(t) + \boldsymbol{a}(t)\Delta t = \boldsymbol{v}(t) + \left(\frac{1}{m}\sum \boldsymbol{F}\right)\Delta t \qquad \text{og} \qquad (3)$$

$$\boldsymbol{r}(t+\Delta t) = \boldsymbol{r}(t) + \boldsymbol{v}(t)\Delta t \qquad (4)$$

Vi ser at vi gjør en systematisk feil ved å bruke disse formlene, fordi i stedet for å f.eks. bruke startverdien $\boldsymbol{a}(t)$ for intervallet t til $t+\Delta t$ vi integrerer over, burde vi brukt en *gjennomsnitts-verdi* for akselerasjonen (eller kreftene) for hele det lille intervallet. Likedan med hastighet i siste ligning.

Ved å velge små tidssteg Δt vil man likevel ofte komme temmelig nær en eksakt riktig løsning, men metoden ovenfor er likevel ganske grov i forhold til bedre metoder. Metoden vi har skissert kalles Eulers metode (med fast skrittlengde). Denne metoden og problematikken knyttet til å finne en mer nøyaktig metode er beskrevet i kapittel 10.8 i Tom Lindstrøms bok, Kalkulus, bind 1. I denne obligen er det tilstrekkelig å bruke den enkle Eulers metode angitt i ligningene 3 og 4. De som har lyst på en ekstra utfordring, kan som nevnt ovenfor forsøke å implementere Eulers midtpunktsmetode eller fjerde ordens Runge-Kuttas metode (begge beskrevet i Lindstrøms bok), men i så fall *må* man først få den enkle Eulers metode til å gå tilfredsstillende før man forfiner programmet.

I obligen kan man da finne hvordan hastigheten til en dråpe endrer seg ved å angi initialbetingelsen:

$$\boldsymbol{v}(1) = \boldsymbol{0}$$

og dernest gå inn i en løkke hvor man bruker rekursjonsformelen:

$$\boldsymbol{v}(i+1) = \boldsymbol{v}(i) + \left(\frac{1}{m}\sum \boldsymbol{F}(\boldsymbol{v}(i))\right)\Delta t$$

Her har jeg allerede pekt på en detalj, nemlig at Matlab-arrayer starter med indeksen 1 (ikke 0 som i Java). Jeg har også indikert at sum av krefter som virker på f.eks. en regndråpe, vil være avhengig av hastigheten til dråpen i forhold til lufta rundt.

På liknende måte kan man gi initialverdien for posisjon og så sette inn i samme løkke som ovenfor, en rekurssjonsformel som gir neste posisjon ut fra forrige posisjon og hastighet.

Noen Matlab-tips

Man bør lage et Matlab-program ("function", .m-fil) som kan lagres og kjøres ved behov. Det er ikke meningen å lage flere ulike programmer i denne obligen for å kunne svare på alle spørsmålene, - ett program som gjør det meste holder (i tilfelle du forsøker noe annet enn Eulers metode, kan det være aktuelt å lage et separat program). Siden koden blir kort (anslagsvis 60 setninger), kan man kjøre hele koden inn i én funksjon uten at det går særlig ut over lesbarheten, eller man kan dele opp koden i flere funksjoner (en i hver sin m-fil slik vi gjennomgikk på forelesningen 8. februar).

Dersom dette er et av de første Matlab-programmet du skal lage, anbefaler jeg sterkt (!) at du først henter opp et av de små Matlab-programmene som ligger på kursets websider ("Materiell fra enkelte forelesninger") og kjører det og gjør små endringer for å bli vant med "programmeringsspråket" og selve Matlab-omgivelsene (inkl. dette å lagre fil på en katalog du ønsker, og få satt opp Matlab slik at det hele fungerer).

Overføre parametre ved kall

Når du skal variere radius i dråpene, kan du selvfølgelig forandre programmet hver gang før det kjøres. Et alternativ er å overføre parametre når man kaller programmet. En måte man kan gjøre dette på er vist skjematisk i følgende programsnutt:

```
function produkt=multiplikasjon(a,b)
% Gir navn på funksjon og returvariabelnavn, samt parametre
% a og b som gis ved kall på funksjon
c = a*b;
produkt = c;
```

Merk at vi altså ikke angir variable*type*. Matlab velger selv f.eks. mellom integer, float og double precision. Vi kunne sløyfet "produkt" helt, og heller skrevet:

```
function multiplikasjon(a,b)
% Gir navn på funksjon samt parametre a og b som gis ved kall på funksjon
c = a*b
```

Ved å sløyfe semikolon til slutt i en setning, vil størrelsen som er beregnet bli skrevet ut (sammen med navnet). Dette kan man gjøre for flere parametre dersom flere skal skrives ut. Merk forøvrig at kommentarer markeres med en % foran selve kommentaren.

Dimensjonering av arrays

Matlab har to-dimensjonale arrays som default, så dersom man skal reservere plass i hukommelsen til en endimensjonal array av 2000 tall (f.eks. doule precision), kan det gjøres slik:

```
x = zeros(1,2000);
```

Da nullstilles også arrayelementene.

Dersom man siden skal plotte data og man ikke har brukt alle 2000 elementene, kan man fjerne overskytende elementer. Dersom man f.eks. i en while-løkke bare har brukt *n* elementer, kan man fjerne de øvrige ved å skrive:

```
x = x(1:n);
```

Merk at når man plotter, så som i:

```
plot(x,t);
```

må arrayen *x* og arrayen *t* ha like mange elementer, naturlig nok.

Interpolasjon

Den eksperimentelle sammenheng mellom luftmotstandskoeffisienten C_D gitt i figur 1 i "Notat om luftmotstand" må inn i programmet på en eller annen måte. Det kan vi få til ved å angi dis-

krete punkter på kurven, hvor avstanden mellom punktene velges slik at kurven er en temmelig rett linje mellom de valgte punktene. Vi definerer så en array for x-koordinatene til de valgte punktene, og en annen array for y-koordinatene. For vår kurve kan denne f.eks. se slik ut:

```
% Vi setter nå inn tabellerte verdier som gir sammenheng ml R og CD.
Reynold=[0.1 0.4 1.0 4.0 10 100 1000 4000 1.0e4 4.0e4 1.0e5 ...
        2.0e5 2.6e5 3.2e5 4.0e5 6.0e5 8.0e5 1.0e6];
Koeff=[260 65 28 8.5 4.1 1.1 0.48 0.40 0.405 0.47 0.42 ...
    0.405 0.39 0.10 0.09 0.105 0.13 0.16];
% Har her brukt et triks for å splitte en lang linje i Matlab.
```

Det er lagt ut en tekst-fil på websidene våre som gir de aktuelle tallene ovenfor, slik at du slipper å skrive inn alle tallene på ny. Skift gjerne til andre variabelnavn dersom du ønsker dette når du kopierer de fire linjene til ditt eget program!

Legg merke til at vi altså kan splitte en Matlab-setning i to linjer ved å la første linje ende på ... (tre punktum etter hverandre). Praktisk!

Når vi så i løkka i progammet skal finne friksjonskraften i luft (luftmotstanden), må man først beregne Reynoldstallet R for den hastigheten man til enhver tid har. Dersom hastigheten er så lav at Reynoldstallet kommer under angitt grense, beregnes friksjonskraften ved hjelp av det analytiske uttrykket $F_m = kv$. For større hastigheter, bruker vi interpolasjon basert på de eksperimentelle dataene vi allerede har matet inn i programmet. Luftmotstandskoeffisienten C_D for en gitt hastighet (tilsvarende Reynoldstallet R) kan i så fall finnes slik:

```
CD=interp1(Reynold,Koeff,R);
```

Her vil CD beregnes som om det faktisk var en rett linje mellom punktet i Reynold - Koeff - arrayene med Reynoldstall litt lavere enn (eller lik) R og punktet i arrayen med Reynoldstallet litt større enn R. Dersom du ikke kjenner ordet “interpolering” fra før, bør du kanskje slå det opp i en fremmedordbok eller spørre andre om betydningen (f.eks. gruppelærer) inntil du får litt tak på hva ordet betyr. Dette er nemlig et ord du trenger også senere!

Løkker i Matlab

De to vanligste (?) løkkene vi har i Matlab har strukturen:

```
while t(n-1)<=falltid
   R = ........
end
```

og

```
for i=1:2000
.......
end
```

Annen kommentar til programmering i FYS-MEK/F 1110

I kurset vårt vil vi bruke noen Matlab-program for å utforske fysikk. Programmene er da bare rene hjelpemiddler, og fungerer som en avansert kalkulator. Og når vi bruker en kalkulator vanligvis, skriver vi ned resultater vi får underveis. Vi lager IKKE et script som skal gi ALLE svar vi er ute etter gjennom EN kjøring! Det måtte vi gjøre i gamle dager da vi leverte inn hull-kortbunken en dag og fikk tilbake resultatet på lange papirutskrifter dagen etter. I dag gir du inn-parametrene mens du sitter ved datamaskinen, og få sekunder etter kommer svaret ut. Du kan notere på et stykke papir hva du puttet inn, og hva du fikk ut, og så gi et nytt sett med inp-utparametre for en ny kjøring. Om og om igjen, helt til du er ferdig. For STORE datamengder egner ikke denne metoden seg selvfølgelig, men det vil neppe bli aktuelt i vårt kurs i år.

Forøvrig vil jeg nevne at jeg fant det nyttig i utviklingen av denne oppgaven å kalle program-met mitt med to inputparametre allerede ved kallet. De to parametrene var dråpediameter og falltid som jeg skulle gjøre beregningene for. Da kunne jeg variere disse parametrene uavhen-gig av hverandre og uten å endre programmet fra en kjøring til den neste. Det er spesielt nyttig fordi vi, etter som diameteren endrer seg, må endre ganske mye på hvor lang tid vi ønsker å følge bevegelsen (tiden jeg har kalt falltid) for å få rimelig gode resultater.

Ellers er det også i Matlab nødvendig å gi kommentarer underveis i programmet slik at du selv om et år eller fem skal kunne skjønne hvordan det er bygd opp.

En annen grunntanke ved programmering som dere lærte i høst, er å ikke forsøke å lage et fullt ferdig program i én jafs. Forsøk å lage en svært forenklet versjon av programmet først, og når den delen synes å være uten feil, kan du legge til litt etter litt, med testing underveis, til du til slutt har et fullt program. Det er ikke like lett å gjøre dette i Matlab som i Java, men forsøk likevel!

Det vil garantert være en god del som sliter med programmeringen og andre deler av obligen. ***Husk at du kan kontakte gruppelærer (eller kursansvarlig) for å få råd og hjelp.*** Det kan også være lurt å diskutere programmet ditt med medstudenter, men det er IKKE lurt å skrive av hva andre har fått til, spesielt ikke dersom du ikke *fullt ut* skjønner koden når du har gått gjen-nom den for deg selv etterpå.

****

For ordens skyld kan det nevnes at om lag 70-80 % av en fullstendig besvarelse (punkt 8 og 9 på andre side av dette skrivet ikke inkludert) må til for å få obligen godkjent. Det klarer alle som faktisk spør og graver og ber om så mye hjelp som nødvendig for å komme gjennom. Men start ikke for sent med oppgaven! Gruppetimene i uke 11 blir stort sett på en av kurs-PC-stuene våre (rom 245 med inngang fra trapp opp i nordøstre hjørne av vestibylen der pendelen henger, eller rom 329 i tredje etasje Fysikkbyggets vestfløy). Spør gruppelærer hvor akkurat din gruppe skal være.

Spesielt vil jeg nevne at alle “off-campus-studentene” våre anbefales å ta kontakt med den gruppelæreren som formelt er deres gruppelærer når dere trenger hjelp. Går ikke det så bra, får dere kontakte meg på e-mail på a.i.vistnes@fys.uio.no eller telefon 2285 5646 eller 9345 1191, så skal jeg forsøke å hjelpe innenfor den tiden jeg har tilgjengelig.