Matematikk 1000

Øvingsoppgaver i numerikk – leksjon 5

Halveringsmetoden

I dette settet skal vi implementere en teknikk som er i stand til å løse ligninger som er uløselige med papir og blyant. I denne sammenhengen introduserer vi `while`-løkker.

Vi minner, som vanlig, om at det forutsettes at de tidligere oppgavesettene er gjort.

Halveringsmetoden

I denne oppgava skal vi ganske enkelt gjøre oppgave 9 fra delkappittel 2.5 i læreboka (Kalkulus). Det går fint å løse oppgava slik den står i læreboka – enten ved å bruke MATLAB eller kalkulator. Men her er det lagt opp til å gjøre ting litt mer detaljert. Hovedpoenget kommer til slutt – nemlig hvordan vi kan implementere *halveringsmetoden* i MATLAB.

a) Forklar hvorfor likninga $x = \cos x$ kan skrives på forma $f(x) = 0$; hva blir $f(x)$?

b) Forklar hvorfor $f(x)$ må ha minst ett nullpunkt intervallet $[0, \pi/2]$.

c) Finn en tilnærma verdi for nullpunktet (der er bare ett) ved å følge denne algoritmen (i kommandovinduet i MATLAB):
1) Bestem to tal a og b slik at $a < b$ og $f(a)$ og $f(b)$ har ulike fortegn.
2) La c være midpunktet mellom a og b og regn ut $f(c)$.
3a) Dersom $f(a) \cdot f(c)$ er negativ ($f(a)$ og $f(c)$ har motsatt fortegn), la c bli din nye b (høgre grense).
3b) Dersom $f(a) \cdot f(c)$ er positiv ($f(a)$ og $f(c)$ har samme fortegn), la c bli din nye a (venstre grense).
4) Gå tilbake til punkt 2 og gjenta flere ganger helt til $b - a$ er liten nok (du bestemmer selv hva som kvalifiserer til "liten nok").

d) Ett av poengene med deloppgave c) er å illustrere hva en *algoritme* er. Et annet poeng er å gjøre det tydelig hvor kjedelig det er å gjøre slike

iterasjoner for hånd. Vi håper dette illustrerer hvorfor det kan være nyttig å skive små programmer, eller skript, av og til. Om du har litt erfaring med koding: Prøv å skrive et skript som impelenterer halveringsmetoden for dette problemet. Alternativt finner du et forslag her:

```
% Implementering av halveringsmetoden.
% Løser likninga x-cos(x)=0.

% Fikserer endepunktene:
a=0;
b=pi/2;

% Funksjonsverdiene i endepunktene:
fa=a-cos(a);
fb=b-cos(b);

Pres=5e-4;        % Angir ønsket presisjon

% While-løkke som gjentar seg så lenge intervallet mellom a og b har
% lengde større enn Pres
while b-a>Pres
  c=(a+b)/2;      % Regner midpunktet og tilordner dette til c
  fc=c-cos(c);    % Funksjonsverdien i midpunktet
  if fa*fc<0      % Undersøker om f(a) og f(c) har ulike fortegn
    b=c;          % Lar midpunktet bli den nye høgre enden
  else
    a=c;          % Lar midpunktet bli den nye venstre enden
  end             % Avslutter if-sats
end               % Avslutter while-løkke

NullPunkt=(a+b)/2 % Skriver nullpunktet til skjerm
```

Linjenummereringa til venstre er ikke en del av selve skriptet; vi har bare tatt de med for å enklere kunne referere til bestemte deler av skriptet senere i teksten. Skriv av eller kopiér dette skriptet inn i en teksteditor og lagre det, eller last det ned fra Fronter. Du kan gi skriptet navnet '`HalveringsEksempel.m`'.

Kjør skriptet og finn en verdi for nullpunktet. Lag et plott av $f(x)$ der du markerer denne nullpunkts-kandidaten på x-aksen. Ser svaret ut til å stemme?

Som du sikkert har sett, har vi innført en ny kommando: `while`. Denne kombinerer egenskapene til både `for` og `if`. Som med `for`, blir sekvensen av kommandoer mellom `while` og `end` utført flere ganger. Men med `for` blir dette antallet bestemt på forhånd; her vil det være avhengig av den logiske påstanden: Kommandoene inne i løkka blir utført så lenge den logiske påstanden, her '`b-a>Pres`', er sann. På den måten ligner satsen på `if`. Men når man i en `if`-sats bare går

videre når man kommer til `end`, vil man gå tilbake til starten av løkka i ei `while`-løkke.

Legg merke til at denne konstruksjonen er alt annet enn "idiotsikker". Om vi skulle komme i skade for å skrive en påstand som alltid er sann, vil løkka heller ikke stanse noen gang. Om dette skjer, kan skriptet stoppes fra kommandovinduet ved å trykke `Ctrl+C`.

e) Hvordan kan vi justere skriptet slik at svaret blir mer nøyaktig?

f) Hvordan kan skriptet justeres slik at vi også kan telle hvor mange ganger `while`-løkka har blitt kjørt – hvor mange iterasjoner vi har brukt?

g) Juster skriptet slik at det løser b)-oppgava i oppgave 2.5: 9. Sett krav om litt høyere nøyaktighet denne gangen.

h) Skriptet kan gjøres litt mer smidig, eller *generisk*, om vi innfører ei funksjonsfil for den funksjonen vi skal finne nullpunktene til. I koden over ser vi at funksjonen $f(x) = x - \cos(x)$ har blitt skrevet inn fire ganger – i linje 2, 9, 10 og 18. Forsøk gjerne å implementere halveringsmetoden på en slik måte at du i stedet refererer til ei funksjonfil, og lag ei slik funksjonfil. Hva er fordelen med ei slik implementering?

Halveringsmetoden er den mest kompliserte algorimten vi skal implementere i dette kurset.