Matematikk 1000

Øvingsoppgaver i numerikk – leksjon 6

for-løkker

I dette settet skal vi introdusere `for`-løkker. Først vil vi bruke `for`-løkker til å regne ut summer. Vi skal også se på hvordan vi kan implementere[1] et skript som løser liknigner ved hjelp av *haveringsmetoden*.

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

Oppgave 1 – Summer og `for`-løkker

Det er kjedelig å gjøre den samme jobben, eller nesten den samme jobben, om igjen og om igjen. Derfor er det bra vi har datamaskiner. I denne oppgava introduserer vi `for`-løkker, som er et veldig nyttig verktøy når man skal gjenta mer eller mindre like operasjoner mange ganger.

a) Skriv ut alle leddene i summen $\sum_{i=1}^{10} i^2$ ned på papir. Hva blir summen?

b) Åpne teksteditoren og skriv av følgende tekst:

```
1 S=0;
2 for i=1:10
3   S=S+i^2;
4 end
5 S
```

Lagre fila i ei passende mappe og gi den navnet `FinnSum.m`.

c) Manøvrér deg til den katalogen hvor fila ligger, og skriv '`» FinnSum`' i kommandovinduet. Får du opp svaret frå oppgave a)?

Dette skriptet inneholder ei `for`-løkke; den starter med '`for i=1:10`' (linje 2) og slutter med '`end`' (linje 4). Det som skjer når skriptet kjører, er at det som

[1] I følge ordboka til Språkrådet kan dette ordet definieres slik: implementere v2 (fra eng, av lat implere 'gjøre ferdig') innføre, iverksette; i edb: installere og få maskinvare el. program til å virke.

står inni løkka, `S=S+i^2;`, blir gjentatt i tur og orden for alle de `i`-verdiene som blir angitt. Første gang er `i` lik det første elementet i vektoren gitt ved `1:10` – altså 1. Da blir 1^2 lagt til variabelen `S` i linje 3, slik at `S` blir endra fra 0 til 1. Når vi kommer til `end` i linje 4, går vi tilbake til `for`-linja, linje 3, og setter `i` til å være det neste elementet i vektoren – altså 2. Så blir 2^2 lagt til `S`, slik at denne variabelen nå blir satt til å være 5. Dette gjentar vi til og med at `i` har fått den siste verdien i vektoren `1:10` – altså 10. Når vi har lagt til 10^2, er vi ferdig med løkka. Neste gang vi kommer til `end` vil vi da slutte å hoppe tilbake til linje 2, og i stedet bare fortsette videre med linje 5. Her står det helt enkelt `S` – uten semilolon. Dermed blir verdien av `S` skrevet til skjerm.

Ta gjerne bort semikolonet i linje 3 og kjør skriptet for å se hva som skjer fra gang til gang.

Som med `if`-satser, er det vanlig å ha et par innrykk på den delen av skriptet som står inni selve løkka; dette gjør skriptet lettere å lese. Denne "innmaten" kan bestå av flere kommandoer – ikke bare én som her.

d) Kommentér skriptet.

e) Justér skriptet over slik at den regner ut $\sum_{i=1}^{100} i^2$. (Dette hadde tatt ganske lang tid å gjøre for hånd, hadde det ikke?)

f) Endre skriptet ditt slik at det regner ut summen $\sum_{i=5}^{20} \sqrt{i}$.

g) Hvilken sum blir beregna i dette ukommenterte skriptet?

```
1 S=0;
2 for i=1:12
3   S=S+1/i;
4 end
5 S
```

Gi svaret ved hjelp av Σ-notasjon.

Oppgave 2 – Konvergens?

Du kan godt ta utgangspunkt i skriptet fra oppgave 1 når du gjør denne oppgava.

Regn ut summen

$$\sum_{k=-5}^{n} \frac{1}{\sqrt{k^2+1}}$$

for $n = 10$, $n = 100$, $n = 1000$ og $n = 10000$. Det gjør du ved å gå inn og endre den øverste grensa i `for`-løkka. Husk å lagre skriptet på nytt for hver gang.

Ser summen ut til å nærme seg noe bestemt når n øker?

Ser summen nedenfor ut til nærme seg et bestemt tall når n øker?

$$\sum_{k=0}^{n} k^2 \cdot 2^{-k/2} \ .$$

Oppgave 3 – Halveringsmetoden

I denne oppgava skal vi løse likninga

$$\sqrt{x} = \cos x \ .$$

a) i) Hvorfor trenger vi MATLAB, kalkulator eller noe lignende for å kunne løse denne likninga?

ii) Plott grafene til $\sqrt{x}$ og $\cos x$ sammen. Hvor mange løsninger ser likninga ut til å ha? Finner du en tilnærma verdi for en løsning?

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

b) Finn en tilnærma verdi for nullpunktet til $f(x)$ (der er bare ett) ved å følge denne algoritmen[2]:

1) Bestem to tall a og b slik at $a < b$ og $f(a)$ og $f(b)$ har ulike fortegn.
2) La c være midtpunktet 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) Hvis ikke – altså hvis $f(a) \cdot f(c)$ 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").
5) La til slutt løsninga være midtpunktet mellom a og b

Dette kan gjøres ved gjentatte operasjoner i kommandovinduet. Men det er tungvindt – dette kan gjøres mye enklere om vi lager et skript med ei `for`-løkke. Forsøk å gjøre dette. Du kan enten prøve å skrive en kode for dette helt på egenhånd, eller du kan ta utgangspunkt i det ukommenterte forslaget nedenfor. Hvis du velger det siste: Forsøk å forstå hva som skjer – linje for linje, og kommentér skriptet.

```
a=0;
b=pi/2;

Fa=sqrt(a)-cos(a);
Fb=sqrt(b)-cos(b);
```

[2]Vi tyr til ordboka igjen: **algoritme** m1 (gjennom eng. og, m.lat., fra arab. etter matematikeren al-Khuwarizmi, død ca 850) i edb, i matematikk: entydig sett med instrukser for løsning av en oppgave.

```
6
7  for i=1:10
8    c=(a+b)/2;
9    Fc=sqrt(c)-cos(c);
10   if Fa*Fc<0
11     b=c;
12   else
13     a=c;
14   end
15 end
16
17 x=(a+b)/2
```

Bestem hvor mange ganger `for`-løkka skal kjøre – antall *iterasjoner* – og kjør skriptet slik at du finner en verdi for nullpunktet. Lag et plott av $f(x)$ der du markerer denne nullpunkts-kandidaten. Ser svaret ut til å stemme?

c) Ved å justere på skriptet fra b), finn løsningene av følgende likninger:

i) $\cos\frac{x}{2} = \arctan x$

ii) $e^x = -\ln x$

iii) $x^5 - 3x^3 + x - 1 = 0$

Husk å kontrollere at du velger `a` og `b` slik at du faktisk har løsninger mellom disse to verdier. En måte å undersøke dette på, kan være å plotte funksjonen som skal være null. På den måten vil du også kunne undersøke om likninga har flere enn ei løsning.

Metoden vi hå har implementert, kaller vi *halveringsmetoden* (evt. intervallhalveringsmetoden). Dette er den mest kompliserte algoritmen vi skal implementere i dette kurset.

Ekstra-oppgave: Maksimering

Lag ei funksjonfil som tar en vektor `x` som argument (*input*) og returnerer den største verdien i `x` (*output*)[3].

[3]MATLAB har rettnok denne funksjonen fra før; den heter `max`. Men det kan jo være en god øvelse å lage den selv også