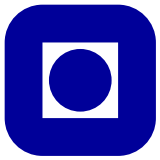


Norges teknisk–naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap (IDI)

TDT4105 IT Grunnkurs
Høst 2008

Løsningsforslag øving 4

Denne øvingen er koordinert med øving 5 i tma4100. Oppgave 1 består i å programmere oppgaver gitt i tma4100/5 i Matlab eller Octave på datamaskin. Oppgave 2 er en CAS-oppgave (Computer Algebra System, oppgave skrevet spesielt for bruk av datamaskin), foreslått av fagstaben ved matematiske fag. H-W-T m.n.o er nummeret på en oppgave i *Hass-Weir-Thomas, University Calculus*.

1 Vi skal programmere Newtons metode for å finne røtter i ligningen

$$f(x) = 0. \tag{1}$$

Framgangsmåte for å finne en løsning i (1) ved Newtons metode:

1. Gjett på en omtrentlig verdi x_0 slik at $f(x_0) \approx 0$.
2. Finn en ny og bedre løsning $x_{n+1} = x_n - f(x_n)/f'(x_n), \quad n = 0, 1, 2, \ldots$
3. Gjenta beregningen i punkt 2 inntil løsningen blir nøyaktig nok eller man har beregnet et maksimum antall nye verdier av x, $n_{\max}$.

a) Sett deg inn i artikkelen om pekere til funksjoner på Matlab-wikien, http://mediawiki.idi.ntnu.no/fag/tdt4105. Skriv funksjonen `newton` for å finne en rot i (1) med Newtons metode. Funksjonen skal ta inn parametere `(f,g,x0,nmax)`, hvor `f` er en funksjon $f(x)$ lagret i en variabel, `g` er den deriverte av funksjonen, $f'(x)$, lagret i en variabel, `x0` er en gjetning (startverdi) og `nmax` er antall nye verdier x_n vi skal regne ut. Funksjonen skal returnere x_{nmax}.

Løsning:

```
function x = newton(f, g, x0, n)
  x = x0;
  for i = 1:n
    x = x - f(x) / g(x);
  end
end
```

b) Bruk `newton` til å løse H-W-T 4.7.14: "Bruk Newtons metode for å finne de to reelle røttene i ligningen $p_4(x) = x^4 - 2x^3 - x^2 - 2x + 2 = 0$". Plott $p_4(x)$ for å finne fornuftige startverdier x_0 for de to røttene. Hint: bruk `x.^4`, `x.^3` osv. heller enn `x^4`, `x^3` når du lagrer funksjonen p_4 i en Matlab-variabel, f.eks. `p4`. Da kan du bruke `p4` både for å regne ut funksjonsverdien for ei liste av x-verdier i `plot`, og til å regne ut enkle verdier som i `newton`.

Løsning:

Fra plotting ser vi at $p_4(x)$ krysser x-aksen to ganger i intervallet $x \in [0, 3]$. Vi prøver derfor å finne røttene ved å starte med f.eks. $x_0 = 0$ og $x_0 = 2$:

```
>> format long;
>> p4 = @(x) x.^4 - 2*x.^3 - x^2 -2*x + 2;
>> p3 = @(x) 4*x.^3 - 6*x.^2 - 2*x -2;
>> rot1 = newton(p4,p3,0,6)
rot1 = 0.630115396163843
>> rot2 = newton(p4,p3,2,9)
rot2 = 2.57327196353519
```

Funksjonen gir endelig presisjon etter hhv. 6 og 9 utregninger (iterasjoner) for de to startverdiene $x = 0$ og $x = 2$. Studentene oppfordres til å sjekke resultater av beregninger, f.eks. ved her å sette inn rotverdiene og sjekke at du får $f(x_r) \approx 0$:

```
>> p4(rot1)/eps
ans = 0
>> p4(rot2)/eps
ans = 16
```

c) Bruk funksjonen `newton` for å løse H-W-T 4.7.25: "Prøv Newtons metode på $f(x) = (x-1)^{40}$, med startverdi $x_0 = 2$ og se hvor nærme du klarer å komme rota $x = 1$". Hva er verdien på `nmax` hvor neste utregning (neste iterasjon) ikke gir bedre resultat ?

Løsning:

```
>> format long;
>> f = @(x) (x-1).^40;
>> g = @(x) 40*(x-1).^39;
>> newton(f,g,2,737);
ans = 1.00000000779128
>> newton(f,g,2,738);
ans = 1.00000000779128
```

Etter 737 iterasjoner har vi $1 + 7.8 \cdot 10^{-9}$. Ytterligere iterasjonen klarer ikke å ta oss nærmere rota $x = 1$.

d) Å regne ut den deriverte og sende den inn som et ekstra argument vil i en del tilfeller være upraktisk. Lag en ny funksjon `diffnewton` som bruker `fdiff` fra wikiartikkelen til å regne ut $f'(x_n)$ istedet for å gi inn den deriverte som et argument. Sjekk `diffnewton` mot `newton` i delproblemene b) og c). Hva finner du ?

Løsning:

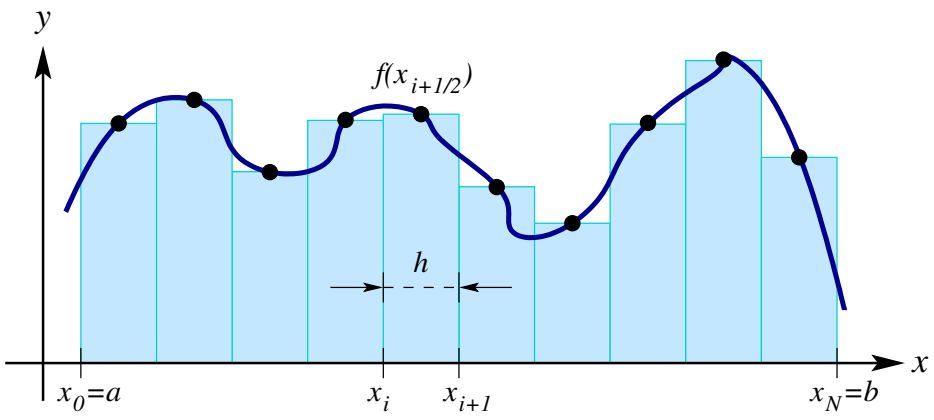
```
function x = diffnewton(f, x0, n)
  x = x0;
  for i = 1:n
    x = x - f(x) / fdiff(f, x);
  end
end
```

(løsningen fortsetter på neste side)

```
>> format long;
>> rot1 = diffnewton(p4,0,6)
rot1 = 0.630115396163843
>> rot2 = diffnewton(p4,2,9)
rot2 = 2.57327196353519
>> diffnewton(f,2,737);
ans = 1.00000012620504
>> diffnewton(f,2,1000000);
ans = 1.00000002969664
```

For $p_4(x)$ gir `diffnewton` med enkel numerisk (beregnet) derivasjon samme resultat som `newton` med analytisk derivert. For den mye brattere funksjonen $(x-1)^{40}$ er `diffnewton` klart dårligere. Selv etter 1000000 utregninger er feilen ≈ 3.8 ganger større enn `newton`. Begge funksjonenen er likevel langt unna å nå et resultat i nærheten av `1+eps`, det nærmeste man kan forvente å nå rota $x = 1$.

2 I denne oppgaven skal vi bruke midtpunktsmetoden til å beregne arealet under en kurve. Vi deler inn intervallet $x \in [a, b]$ i N linjestykker. Deretter finner vi funksjonsverdien eller høyden av rektangelet midt på linjestykket. Summeres arealet av de N rektanglene har vi en omtrentlig verdi for arealet under kurven mellom a og b, se figur 1. Oppgaven tilsvarer H-W-T 5.1.23.



Figur 1: Midtpunktsmetoden for å beregne arealet under en kurve.

a) Plott funksjonen $f(x) = \sin x$ gitt på intervallet $x \in [0, \pi]$.

Løsning:

```
>> x = 0:0.1:pi;
>> plot(x,sin(x));
```

b) Skriv en funksjon `area_midpoint` for å beregne arealet av en kurve ved midtpunktsmetoden. Ta inn funksjonen $f(x)$ i en variabel `f`, endepunktene i variable `a` og `b` og antall linjestykker `N` intervallet $[a, b]$ skal deles i.

Løsning:

```
function asum = area_midpoint(f, a, b, n)
  asum = 0;
  h = (b-a) / n;

  for i = 0:n-1                          % or 1:n
    asum = asum + f(a + (i+0.5)*h);   % and i-0.5
  end

  asum = asum * h;
end
```

c) Beregn gjennomsnittsverdien m_N av $\sin x$ på $[0, \pi]$ for N = 100, 200 og 1000 intervaller. Gennomsnittsverdien er arealet / lengden av intervallet.

Løsning:

```
>> format long;
>> f = @(x) sin(x);
>> m100 = area_midpoint(f, 0, pi, 100)/pi
m100 = 0.636645953060006
>> m200 = area_midpoint(f, 0, pi, 200)/pi
m200 = 0.636626317399378
>> m1000 = area_midpoint(f, 0, pi, 1000)/pi
m1000 = 0.636620034167044
```

d) Løs ligningen $f(x) = m_{1000}$. Hint: bruk Newtons metode fra oppgave 1.

Løsning:

```
>> f = @(x) sin(x) - m1000;
>> g = @(x) cos(x);
>> rot = newton(f,g,0,5)
rot = 0.690107430854598
>> f(rot)/eps
ans = 0
```

Det er godkjent å bruke hva som helst av startverdi i intervallet $x_0 \in [0, \pi]$ – uansett vil man trenge maksimum 3-5 iterasjoner. Det er også greit å bruke `diffnewton` istedet for `newton`. Gjenta også her at løsningen må sjekkes ved innsetting, f.eks. som vist.