Norges teknisk–naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

TDT4110 IT Grunnkurs
Høst 2012

Øving 7

Denne øvingen er en fellesøving laget i samarbeid med emnet TMA4100 Matematikk 1. I den sammenheng vil det være noen oppgaver som er ment for de som tar Matematikk 1. De som tar Matematikk 1 **MÅ** gjøre de matterelaterte oppgavene merket "TMA4100" (dvs. oppgave 1 og 2), men står fritt til gjøre oppgaver merket "øvrige"

Øvrige studenter vil få alternative oppgaver merket "øvrige", men står fritt til gjøre oppgaver merket "TMA4100".

Oppgaver merket "alle" må gjøres av alle.

Alle teorispørsmål skal besvares og begrunnes. Alle oppgavene skal demonstreres til en studentassistent på sal. I oppgaver der du skriver programkode skal også denne vises fram. Lykke til!

1 Newtons metode - TMA4100

Referer til oppgave 1 i Øving 8 TMA4100.

a) Lag en funksjon `polynom`, som tar inn ett argument `x`, og returnerer utregningen av polynomet $f(x) = x^5 - 4x^3 + 10x^2 - 10$. Lag også en funksjon `polynom_derivative` som tar inn ett argument `x`, og returnerer $5x^4 - 12x^2 + 20x$; mao. den deriverte av $f(x)$.

b) Implementer newtons metode fra kapittel 4.7 i læreboken i TMA4100 som en funksjon med navn `newton`.

Funksjonen skal ta inn en generell funksjon `func`, og dens deriverte `deriv`, en startverdi (gjettet nullpunkt) `start`, en feiltoleranse `threshold`, og et maksimalt antall iterasjoner `max_iterations`. Funksjonen skal returnere et estimat på et nullpunkt innenfor feiltoleransen. Hvis et estimat ikke kan finnes innenfor feiltoleransen i løpet av det maksimale antall iterasjoner, returnerer `False`.

Eksempel 1

Test implementasjonen din slik (riktig output står som kommentar):

```
print(newton(polynom, polynom_derivative, 1, 0.000001, 20)) # 1.201384
print(newton(polynom, polynom_derivative, -3, 0.000001, 20)) # -2.684766
print(newton(polynom, polynom_derivative, -1, 0.000001, 20)) # -0.882417
print(newton(polynom, polynom_derivative, 0.01, 0.000001, 20)) # False
print(newton(polynom, polynom_derivative, -2.08, 0.000001, 20)) # False
```

2 Simpsons metode - TMA4100

Referer til oppgave 2 i Øving 8 TMA4100.

Formelen det refereres til står øverst i den oppgaven.

Vi ønsker å regne ut integralet

$$\int_0^1 e^{-x^2} \mathrm{d}x$$

a) Implementér Simpsons metode som en pythonfunksjon `simpson`. Funksjonen skal ta inn en funksjon `func`, integrasjongrensene `a` og `b`, og antall delintervaller `h`. Finn en tilnærmet verdi av integralet gitt over ved hjelp av (din egen implementasjon av) simpsons metode med 1000 delintervaller.

TIPS: Implementer e^{-x^2} som en pythonfunksjon.

b) Feilen for Simpsons metode er gitt som en funksjon av $f^{(4)}$. Ofte er det vanskelig å finne gode skranker for $f^{(4)}(x)$, men vi ønsker likevel å ha en viss kontroll på hvor stor feil vi gjør. Vi skal nå se på en mulig praktisk løsning på problemet.

La S_n være tilnærmingen til integralet vi får når vi bruker Simpsons metode med n delintervaller. Det er rimelig å anta at S_8 er mye mer nøyaktig enn S_4. Det betyr at $|S_4 - S_8|$ kan være et rimelig estimat for feilen i tilnærmingen S_4.

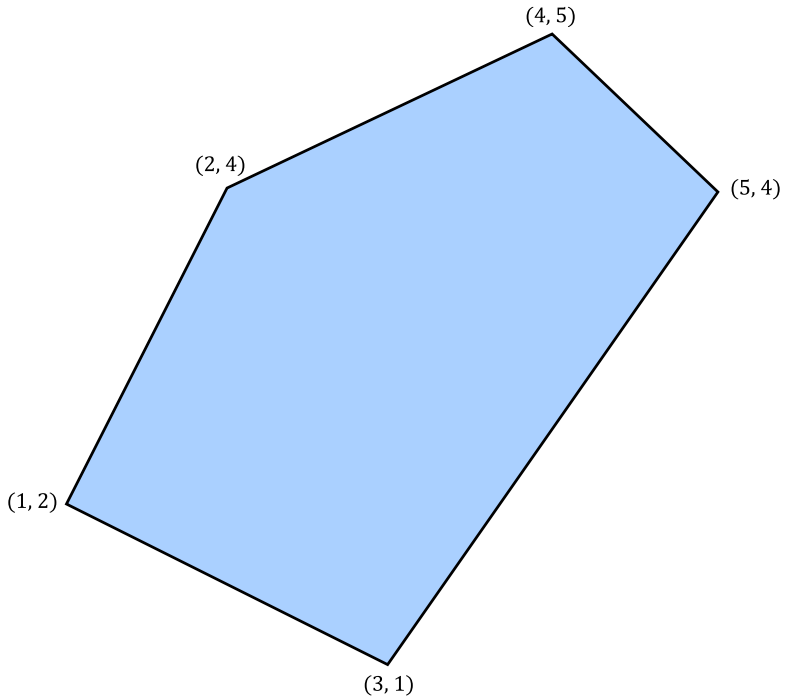
Hvis dette estimatet sier at feilen i S_4 var for stor kan vi i stedet bruke S_8 som tilnærming. Nå kan vi finne et estimat for feilen i S_8 ved å regne ut S_{16} og bruke $|S_8 - S_{16}|$ som estimat. Slik kan vi fortsette: regn ut tilnærmingen S_{2^i} og estimer feilen ved å regne ut $S_{2^{i+1}}$ og differensen $|S_{2^i} - S_{2^{i+1}}|$. Vi stopper når feilestimatet blir mindre enn en oppgitt toleranse.

Du har allerede regnet ut S_4 og S_8, samt feilestimatet $|S_4 - S_8|$. Fortsett prosessen til du estimerer feilen i S_{2^i} til å være mindre enn toleransen 10^{-8}.

Skriv funksjonen `simpson_error` som tar inn en funksjon `func`, Integrasjonsgrensene `a` og `b`, og feiltoleransen `error`. `a`, `b`, og `error` er alle tall.

Regn ut integralet definert i begynnelsen av oppgaven med en feiltoleranse på 10^{-8}.

3 Omkrets - Øvrige



Figur 1: Polygon

Et ploygon kan representeres som to lister med henholdsvis x- og y-koordinatene i xy-planet. Femkanten på figuren kan representeres i Python slik:

`x = [1, 2, 4, 5, 3]; y = [2, 4, 5, 4, 1]`.

Lag en funksjon `perimeter` som tar inn to lister, `x` og `y`, som parameter. Disse listene anngir punkter i xy-planet. Funksjonen skal returnere lengden langs kanten (omkretsen) av polygonet beskrevet av de to listene.

Lengden av en kant i polygonet er gitt av Pytagoras' formel:

$$\sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2}$$

Om man regner ut lengden av den første kanten i eksempelet over får man:

$$x_i = 1, x_{i+1} = 2, y_i = 2, y_{i+1} = 4$$

$$\text{lengde} = \sqrt{(1-2)^2 + (2-4)^2}$$

PROTIP: Implementer pytagoras som en funksjon og benytt en for-løkke inni `perimeter`-funksjonen.

Eksempel 2

Test funksjonen din slik (riktig output står som kommentar):

```
print(perimeter([1, 1, 2] , [1, 2, 2])) # 3.4142
print(perimeter([1, 1, 2, 2] , [1, 2, 2, 1])) # 4
print(perimeter([1, 2, 4, 5, 3], [2, 4, 5, 4, 1])) # 11.7280
```

4 Løkker - Alle

a) Et primtall er et tall større enn 1 som kun er delelig med seg selv og 1.
Lag en funksjon `is_prime` som tar inn et tall som parameter og tester om det er et primtall ved hjelp av en for-løkke. Funksjonen skal returnere `True` hvis tallet er et primtall og `False` hvis det ikke er det.

b) Lag en funksjon `separate` som tar inn to argumenter: en liste med tall `numbers`, og et tall `threshold`. Funksjonen skal returnere to lister: den første listen skal inneholde alle elementer fra `numbers` som er mindre enn threshold, og den andre listen skal inneholde alle elementer fra `numbers` som er større enn eller lik `threshold`.

c) En matrise er et rektangulært sett av elementer ordnet i rader og kolonner. Radene er de horisontale linjene, og kolonnene er de vertikale linjene. Elementet øverst til venstre med verdi 1 er i rad 1, kolonne 1.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Figur 2: En 3×3 matrise

På litt samme måte som at man kan representere matematiske vektorer med lister, kan man representere matriser ved hjelp av lister som inneholder lister:

```
matrise = [
 [1 2 3],
 [4 5 6],
 [7 8 9]]
```

Elementene i matrisen aksesseres på følgende måte: `matrise[radnr][kolnr]`.
Husk at radnummer og kolonnenummer er null-indeksert (begynner på 0) i Python i motsetning til matematisk notasjon der de er 1-indeksert!
Lag en funksjon `multiplication_table` som tar inn et tall `n` som parameter og returnerer gangetabellen fra 1 til n som en matrise med n rader og n kolonner.
TIPS: Man kan legge til elementer (som også kan være lister) på slutten av en liste med `liste.append(element)`.

Eksempel 3

`multiplication_table(3)` returnerer følgende matrise:

```
[[1 2 3],
 [2 4 6],
 [3 6 9]]
```

5 Strenghåndtering - Alle

a) Lag en funksjon som sjekker om to strenger er like ved å sammenligne dem tegn for tegn. Funksjonen returnerer `True` hvis strengene er like; `False` ellers.

b) Lag en funksjon som tar inn en streng, reverserer den og returnerer den reverserte strengen. Dette skal gjøres tegn for tegn med en løkke.

c) Et palindrom er et ord som staves likt begge veier (eks. "abba"). Lag en funksjon som returnerer `True` om en streng er et palindrom; `False` ellers.

d) Lag en funksjon som tar inn to strenger og sjekker om den første strengen inneholder den andre. Dersom den gjør det, returner posisjonen den andre strengen begynner på (fra 0). Returner `False` ellers.