13. september, 2018

MAT-INF 1100: Obligatorisk oppgave 1

Innleveringsfrist: 27/9-2018, kl. 14:30 i Devilry

Obligatoriske oppgaver («obliger») er en sentral del av MAT-INF1100 og er utmerket trening i å besvare en matematisk oppgave. Denne veiledningen er ment for å klargjøre noen krav vi som retter obligene stiller, både for å forenkle vår jobb i retteprosessen og for at dere skal slippe å måtte levere på nytt på grunn av formaliteter.

For å få godkjent en oblig bør man ha fått til om lag 2/3 av oppgavesettet og gitt alle deloppgavene et seriøst forsøk. Dersom det allikevel skulle være en oppgave du overhodet ikke får til, prøv å sette ord på hva du har tenkt og hvor det ikke lot seg løse. Dersom man får «ikke godkjent» på en oblig og får et nytt forsøk vil det ha innleveringsfrist to uker etter første innlevering.

1 Krav til innlevering

- Besvarelsen skal leveres elektronisk via Devilry (https://devilry.ifi.uio.no/). Du velger selv om du skriver besvarelsen for hånd (og scanner besvarelsen) eller om du skriver løsningen direkte inn på datamaskin (f.eks. ved bruk av LaTeX). Skannede ark må være godt lesbare. Godkjente filtyper: PDF
- Hoveddelen av besvarelsen skal leveres som én fil og i PDF-format. Det er lov å skanne inn håndskrevne ark så lenge disse er godt lesbare.
- Besvarelsen skal inneholde navn, emne og oblignummer.
- Husk at andre skal lese og forstå din besvarelse! Besvarelsene bør derfor føres på en oversiktlig måte i den rekkefølgen de står oppgitt i.
- Alle plott og figurer som er med på å besvare oppgaven må inkluderes (husk også riktige akser og enheter). Du bør også skrive kort hva hver figur viser, for eksempel: *Figur 2: Plott av funksjonen* $f(x) = x^2 - 3$.

- Kode skrives i Python og leveres ved siden av besvarelsen, men **all diskusjon** og **all besvarelse** av oppgaven skal inn i hovedfila. Det vil f.eks. si at du ikke kan levere et python-skript som besvarelse på en oppgave hvor du blir bedt om å lage en algoritme. Det er viktig at programkoden du leverer inneholder et kjøreksempel, slik at det er lett å se hvilket resultat programmet gir.

- Det oppmuntres til samarbeid om oppgavene, men du skal selv ha formulert og skrevet den besvarelsen som leveres inn, og den skal gjenspeile din forståelse av stoffet. Du kan bli bedt om å redegjøre muntlig for innholdet i din besvarelse.

- Besvarelsen bør inneholde diskusjon om resultatene dere kommer fram til. Eksempler på spørsmål man kan stille seg for å vite hva man skal kommentere er: *Gir dette plottet noen mening? Ser virkelig grafen til* $f(x)$ *slik ut? Var dette resultatet som forventet? Kan dette skyldes numeriske feil?*

Husk at de to obligatoriske oppgavene i MAT-INF 1100 begge må bestås for å kunne gå opp til endelig eksamen i emnet.

Søknad om utsettelse av innleveringsfrist Hvis man blir syk eller av andre grunner trenger å søke om utsettelse av innleveringsfristen må du ta kontakt med studieadministrasjonen ved Matematisk institutt (7. et. Niels Henrik Abels hus, e-post: studieinfo@math.uio.no) i god tid før innleveringsfristen. For fullstendige retningslinjer for innlevering av obligatoriske oppgaver se

uio.no/studier/admin/obligatoriske-aktiviteter/mn-math-oblig.html

2 LaTeX

Vi oppmuntrer til bruk av LaTeX. Dette er et slags programmeringsspråk for å skrive naturfaglige dokumenter som det kan være greit å lære seg først som sist. Hjelp til LaTeX kan du få omtrent overalt, fra utallige forum på verdensveven til gruppelærere over hele MatNat-fakultetet. Dersom du trenger hjelp til å komme i gang vil du finne LaTeX-versjonen av denne teksten samme sted som du finner PDF-filen. Vi vil også legge ut en LaTeX-mal med eksempler på hvordan inkludere kode og figurer som du kan bruke til å gjøre obligene i MAT-INF1100 om du vil.

Oppgaver

Oppgave 1. Vi skal se på differensligningen

$$x_{n+2} - 2x_{n+1} - x_n = 0, \quad \text{med } x_0 = 1 \text{ og } x_1 = 2. \tag{1}$$

a) Lag et dataprogram som simulerer denne ligningen og skriver ut følgen $x_2, x_3, \ldots, x_{100}$.

b) Simuler ligningen og skriv ut følgen $x_2, x_3, \ldots, x_{100}$ når startverdien x_1 endres til $x_1 = 1 - \sqrt{2}$

c) Vis at den generelle løsningen av ligningen $x_{n+2} - 2x_{n+1} - x_n = 0$ er på formen

$$x_n = C(1 - \sqrt{2})^n + D(1 + \sqrt{2})^n$$

og at initialverdiene $x_0 = 1$ og $x_1 = 1 - \sqrt{2}$ bestemmer den endelige løsningen til å være $x_n = (1 - \sqrt{2})^n$.

d) Sjekk om den analytiske løsningen i (c) stemmer med dine beregninger i (b) og forklar eventuelle avvik.

e) (Om du har lyst og tid). Bruk simuleringene i (b) til å estimere avrundingsenheten (the round-off unit) på maskinen din.

Oppgave 2. Binomialkoeffisienten $\binom{n}{i}$ er definert som

$$\binom{n}{i} = \frac{n!}{i!(n-i)!} \tag{2}$$

der $n \geq 0$ er et ikke-negativt heltall og i er et heltall i intervallet $0 \leq i \leq n$.

Binomialkoeffisientene dukker opp i mange ulike sammenhenger og må ofte beregnes på datamaskin. Siden alle binomialkoeffisientene er heltall (divisjonen i (2) gir aldri noen rest) så er det rimelig å bruke heltallige variable i slike beregninger. For små verdier av n og i går det bra, men for større verdier får vi fort problemer fordi både teller og nevner i (2) lett blir større enn det største heltallet som kan representeres med 32- eller 64-bits heltall selv om binomialkoeffisienten i seg selv ikke er så stor. I mange språk vil dette føre til en form for «overflow», men selv i et språk som Python som unngår dette ved at innebygget programvare kommer til unnsetning, vil beregningene gå langt saktere enn ellers. Vi kan bruke flyttall i stedet, men selv da vil vi lett få «overflow» underveis i beregningene. I denne oppgaven skal vi se hvordan vi kan unngå slike problemer.

Hvis vi ser nærmere på definisjonen (2) legger vi merke til at vi kan forkorte i stor skala,

$$\binom{n}{i} = \frac{1\cdot 2\cdots i\cdot(i+1)\cdots n}{1\cdot 2\cdots i\cdot 1\cdot 2\cdots(n-i)} = \frac{i+1}{1}\cdot\frac{i+2}{2}\cdots\frac{n}{n-i}.$$

Ved hjelp av produktnotasjon kan vi derfor skrive $\binom{n}{i}$ som

$$\binom{n}{i} = \prod_{j=1}^{n-i}\frac{i+j}{j}. \tag{3}$$

a) Skriv et program som beregner binomialkoeffisienter ved hjelp av formelen (3). Test metoden på eksemplene

$$\binom{5000}{4} = 26010428123750,$$

$$\binom{100000}{60} = 1.18069197996257\cdot 10^{218},$$

$$\binom{1000}{500} = 2.702882409454366\cdot 10^{299}.$$

Hvorfor må du bruke flyttall og hvilke resultater får du?

b) Er det nå mulig at du underveis får «overflow» om binomialkoeffisienten du skal beregne er mindre enn det største flyttallet som kan representeres på maskinen din?

c) I vår utledning av (3) forkortet vi $i!$ mot $n!$ i (2). En alternativ metode kan utledes ved å forkorte $(n-i)!$ mot $n!$ isteden. Utled denne alternative metoden på samme måte som over og diskuter når de to metodene bør brukes (du trenger ikke programmere denne metoden, det holder å argumentere matematisk).

Oppgave 3. Følgende Python-program er gitt:

```
from random import random

antfeil = 0; N = 100000

for i in range(N):
    x = random(); y = random(); z = random()
```

```
    res1 = (x*y)*z
    res2 = x*(y*z)

    if res1 != res2:
        antfeil += 1
        x0 = x; y0 = y; z0 = z
        ikkeass1 = res1
        ikkeass2 = res2

print (100. * antfeil/N)
print (x0, y0, z0, ikkeass1 - ikkeass2)
```

En kjøring av programmet ga utskriften

```
35.056
0.012421854744752325 0.1751776618141262 0.6421241188091763 -2.168404344971009e-19
```

a) Forklar hva programmet gjør og hva utskriften forteller oss.

b) Endre programmet slik at det i stedet sammenligner de to størrelsene `x*(y+z)` og `x*y+x*z`, og kjør det på nytt. Du skal nå se at det første tallet som blir skrevet ut blir ulikt det første tallet som ble skrevet ut over (d.v.s. `35.056`). Kan du tenke deg en mulig forklaring på dette?

Lykke til!