15. september 2022

MAT-INF 1100

Obligatorisk oppgave 1 av 2

Innleveringsfrist

Torsdag 29. september 2022, klokken 14:30 i Canvas (canvas.uio.no).

Instruksjoner

Du velger selv om du skriver besvarelsen for hånd og skanner den, eller om du skriver løsningen direkte inn på datamaskin (for eksempel ved bruk av LaTeX). Skannede ark må være godt lesbare. Det forventes at man har en klar og ryddig besvarelse med tydelige begrunnelser. Besvarelsen skal leveres som én PDF-fil. Husk å inkludere eventuell kode og kjøreeksempel, samt relevante plott og figurer i PDF-filen.

Merk at man har ett forsøk på oppgaven. Samarbeid og alle slags hjelpemidler er tillatt, men den innleverte besvarelsen skal være skrevet av deg og reflektere din forståelse av stoffet. Er vi i tvil om du virkelig har forstått det du har levert inn, kan du bli bedt om en muntlig redegjørelse.

Søknad om utsettelse av innleveringsfrist

Hvis du blir syk eller av andre grunner trenger å søke om utsettelse av innleveringsfristen, må du ta kontakt med studieadministrasjonen ved Matematisk institutt (e-post: studieinfo@math.uio.no) før innleveringsfristen. Vitenskapelig ansatte kan ikke innvilge utsettelser.

For å få adgang til avsluttende eksamen i dette emnet, må man bestå alle obligatoriske oppgaver i ett og samme semester.

For å få godkjent denne første obligatoriske oppgaven må du ha gjort seriøse forsøk på å løse alle deloppgavene, og minst halvparten av oppgavene må være tilfredsstillende besvart.

For fullstendige retningslinjer for innlevering av obligatoriske oppgaver, se her:

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

LYKKE TIL!

Oppgaver

Oppgave 1. Vi skal se på differensligningen

$$x_{n+2} - 6x_{n+1} + 3x_n = 0, \quad \text{med } x_0 = 1 \text{ og } x_1 = 1. \tag{1}$$

a) Skriv et program som regner ut og skriver ut $x_2, x_3, \ldots, x_{100}$.

b) Vi endrer den ene startverdien x_1 til $x_1 = 3 - \sqrt{6}$. Skriv et nytt program som regner ut og skriver ut $x_2, x_3, \ldots, x_{100}$.

c) Vis at den generelle løsningen av $x_{n+2} - 6x_{n+1} + 3x_n = 0$ er

$$x_n = C(3 - \sqrt{6})^n + D(3 + \sqrt{6})^n$$

og at initialverdiene $x_0 = 1$ og $x_1 = 3 - \sqrt{6}$ gir den spesielle løsningen $x_n = (3 - \sqrt{6})^n$.

d) Prøv deg frem i programmet fra b) med å øke antall iterasjoner ytterligere (til mye mer enn 100). Når antall iterasjoner blir stort, stemmer det du fant i b) med den analytiske løsningen du fant i c)?

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 $0 \leq i \leq n$.

Disse dukker opp i mange ulike sammenhenger og må ofte beregnes på datamaskin. Siden n og i er heltall 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 kan vi få problemer siden både teller og nevner i (2) kan bli større enn det største heltallet som kan representeres med 32- eller 64-bits heltall (selv om $\binom{n}{i}$ i seg selv ikke er så stor). I mange språk vil dette føre til «overflow», men selv i et språk som Python (som unngår dette ved hjelp av innebygget programvare), vil beregningene gå sakte, med mindre man forenkler formelen (2). Alternativt til heltall, kan vi bruke flyttall til å beregne (2), men da vil vi lett få «overflow» underveis i beregningene. I denne oppgaven skal vi se på disse to måtene å beregne $\binom{n}{i}$ på, og hvordan man kan unngå problemer med «overflow».

Ser vi nærmere på (2) finner vi at vi kan forkorte i stor skala,

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

Fra dette finner vi ved hjelp av produktnotasjon to ekvivalente formler for $\binom{n}{i}$. En hvor multiplikasjon gjøres separat i teller og nevner:

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

Og en bestående av et produkt av rasjonale tall:

$$\binom{n}{i} = \prod_{j=1}^{i} \frac{n-j+1}{j}. \tag{4}$$

a) Vi skal bruke heltallsvariable til å beregne $\binom{n}{i}$ i Python, og se at slike beregninger er pålitelige, men kan ta lang tid.

Skriv et program som beregner $\binom{n}{i}$ ved hjelp av (3), dvs ved å beregne teller of nevner hver for seg som produkt av heltall.

Test metoden på følgende tre binomialkoeffisienter (siste eksempel kan ta rundt 10 sekunder å beregne med denne framgangsmåten):

$$\left.\begin{aligned} \binom{5000}{4} &= 26010428123750, \\ \binom{1000}{500} &\approx 2.7028824094543655 \cdot 10^{299}, \\ \binom{100000}{99940} &\approx 1.1806919799625687 \cdot 10^{218} \end{aligned}\right\} \tag{5}$$

Hvilke resultater får du?

b) Vi skal nå i stedet bruke flyttallsvariable til å beregne $\binom{n}{i}$, og se at dette kan være mer effektivt enn heltallsvariable, men at det kan lede til overflow ved beregning av store binomialkoeffisienter.

Skriv et program som beregner $\binom{n}{i}$ ved hjelp av formelen (4), dvs. som et produkt av flyttall. Test metoden på de tre binomialkoeffisientene i ligning (5).

Hvilke resultater får du? Merk at avrundingsfeil her kan lede til små avvik, og at «overflow» kan oppstå. Selv om $\binom{n}{i}$ er mindre enn det største flyttallet maskinen kan representere, kan du likevel få «overflow»?

c) Anta $i > n/2$. Kan du tenke ut en enkel modifisering av programmet fra b) som vil hjelpe oss med å unngå «overflow» i en del situasjoner?

Oppgave 3. For tre vilkårlige relle tall $x, y, z \in \mathbb{R}$ vet vi fra kalkulus at:

$(x+y)+z = x+(y+z)$ **(Assosiative loven for addisjon)**

$x(y+z) = xy+xz$ **(Distributive loven for multiplikasjon mhp addisjon)**.

I denne oppgaven skal vi se om om disse disse to lovene også gjelder når vi bruker flyttall i Python. Følgende Python-program er gitt:

```
from random import random

antfeil = 0; N = 100000
x0 = 0; y0 = 0; z0 = 0; ikkelik1 = 0; ikkelik2 = 0;
for i in range(N):
    x = random(); y = random(); z = random()
    res1 = (x + y) * z
    res2 = x*z + y*z
    if res1 != res2:
        antfeil += 1
        x0 = x; y0 = y; z0 = z
        ikkelik1 = res1
        ikkelik2 = res2
print (100. * antfeil/N)
print (x0, y0, z0, ikkelik1 - ikkelik2)
```

Hvor "random()" genererer et tilfeldig tall i $(0, 1)$. En kjøring ga som output

```
30.859
0.6087077776638925 0.9204274878392227 0.06851310883531125 -1.3877787807814457
```

a) Forklar hva programmet gjør, og tolk output.

b) Forklar og kjør følgende program (koden kan du finne på kurssidene), og tolk output.

```
from random import random

antfeil = 0; N = 100000
x0 = 0; y0 = 0; z0 = 0; ikkelik1 = 0; ikkelik2 = 0;
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
        ikkelik1 = res1
        ikkelik2 = res2
print (100. * antfeil/N)
print (x0, y0, z0, ikkelik1 - ikkelik2)
```

Kan du eksperimentelt ved hjelp av en endring i programmet si i hvilken rekkefølge tre flyttall i Python blir addert (når du skriver $x + y + z$)? Er det $(x + y) + z$ eller er det $x + (y + z)$?