ℹ # Forside

Eksamensoppgave i: INGA1002/INGG1002/INGT1002 Programmering og numerikk
Dato: 16.12.2024
Tid: kl 09:00 - 12:00

Faglig kontakt under eksamen:
Trondheim INGT1002
Programmering: Majid Rouhani
Numerikk: Markus Arthur Köbis

Gjøvik INGG1002
Programmering: Jon Yngve Hardeberg
Numerikk: Charles Curry

Ålesund INGA1002
Programmering: Kai Erik Hoff
Numerikk: Truls Helge Øi

Møter i eksamenslokalet: NEI

Hjelpemiddelkode/Tillatte hjelpemidler: D - Ingen trykte eller håndskrevne hjelpemidler tillatt. Bestemt, enkel kalkulator tillatt.

ANNEN INFORMASJON:

Les oppgavene nøye og gjør dine egne antagelser. Presiser i besvarelsen hvilke forutsetninger du har lagt til grunn i tolkning/avgrensing av oppgaven. Dette gjelder ikke-flervalgsoppgaver.

Faglig kontaktperson skal kun kontaktes dersom det er direkte feil eller mangler i oppgavesettet. Henvend deg til en eksamensvakt hvis du mistenker feil og mangler. Noter spørsmålet ditt på forhånd.

FAGSPESIFIKK INFORMASJON

Ingen håndtegninger:
Denne eksamenen tillater ikke bruk av håndtegninger. Har du likevel fått utdelt skanne-ark, er dette en feil. **Arkene vil ikke bli akseptert for innlevering, og de vil derfor heller ikke sendes til sensur.**

Vekting av oppgavene: Oppgavesettet er delt inn i 3 seksjoner (programmering, numerikk, og felles oppgaver). Programmering (1-6) og numerikk (7-12) oppgavene teller 33% hver, mens siste delen (13-17) teller 34%.

Varslinger: Hvis det oppstår behov for å gi beskjeder til kandidatene underveis i eksamen (f.eks. ved feil i oppgavesettet), vil dette bli gjort via varslinger i Inspera. Et varsel vil dukke opp som en dialogboks på skjermen. Du kan finne igjen varselet ved å klikke på bjella øverst til høyre.

Trekk fra/avbrutt eksamen: Blir du syk under eksamen, eller av andre grunner ønsker å levere blankt/avbryte eksamen, gå til "hamburgermenyen" i øvre høyre hjørne og velg «Lever blankt». Dette kan <u>ikke</u> angres selv om prøven fremdeles er åpen.

Tilgang til besvarelse: Etter eksamen finner du besvarelsen din i arkivet i Inspera. Merk at det kan ta én virkedag før eventuelle håndtegninger vil være tilgjengelige i arkivet.

1 Matematiske operasjoner

Gitt variabeldefinisjonen
c = 7

Hva blir verdien til c etter at vi utfører følgende operasjoner?
c = c ** 2
c %= 5
Velg ett alternativ:

- 2
- 4.0
- 3
- 4

Gitt variabeldefinisjonen
z = 8

Hva blir verdien til z etter at vi utfører følgende operasjoner?
z += 10
z /= 3

Velg ett alternativ

- 9
- 3
- 6.0
- 6

Maks poeng: 4

2 Funksjoner

Denne koden definerer en funksjon `is_prime(n)` som sjekker om et gitt tall `n` er et primtall.
Test av funksjonen gir disse resultatene:

is_prime(10) # returnerer False
is_prime(11) # returnerer True
is_prime(1) # returnerer False
is_prime(13) # returnerer True

Lag funksjonen `is_prime` ved å plassere kodefragment i rett rekkefølge. Noen av fragmentene skal **IKKE** brukes.

Pass på å plassere draområdene slik at de "snapper" til ønsket rute. Se bort fra innrykk

```
def is_prime(n):

if n <= 1:

return False

for j in range(2, n):

if n % j == 0:

return False

return True

from math import sqrt

if n // j == 0:
```

Maks poeng: 7

3 Sammenligningsoperatorer

Gitt følgende kode:

a = 5
b = 5.0
c = "5"
d = True
e = False

Nedenfor står en rekke uttrykk. For hvert uttrykk, kryss av for **True** hvis uttrykket er sant, eller **False** hvis det er usant, eller kryss av for **Error** hvis uttrykket vil gi syntaksfeil.

Finn de som passer sammen:

	False	True	Error
b > a	○	○	○
a >= b	○	○	○
d = = 1	○	○	○
a != c	○	○	○
e < = 0	○	○	○
a == b	○	○	○
d < e	○	○	○

Maks poeng: 7

4 Lister og indekser

Gitt følgende programkode:

```python
temperatures = [15.5, 17.2, 16.8, 14.9, 18.3, 19.0, 16.5]

def f1(temps):
    return sum(temps) / len(temps)

def f2(temps):
    return sum(temps[:3]) / len(temps[:3])

def f3(temps):
    return sum(temps[-4:]) / len(temps[-4:])

def f4(temps):
    return max(temps)

def f5(temps):
    return temps[::-1]

def f6(temps):
    result = []
    for temp in temps:
        if temp > 17:
            result.append(temp)
    return result
```

Hver rad nedenfor har et kall til funksjonene ovenfor. Skriv i tekstfeltet hva svaret blir. Flyttall avrundes til en desimal.

print(f"{f1(temperatures):.1f}") skriver ut

print(f"{f2(temperatures):.1f}") skriver ut

print(f"{f3(temperatures):.1f}") skriver ut

print(f"{f4(temperatures):.1f}") skriver ut

print(f"{f5(temperatures)}") skriver ut

print(f"{f6(temperatures)}") skriver ut

Maks poeng: 6

5 Numpy og matriser

Skriv ferdig funksjonen **get_diagonal** (velg kode fra nedtrekksmeny) som tar inn en matrise *A*, og returnerer en ny matrise *A_diag* der det **kun** er diagonalelementene fra *A* som er med (resten av matriseelementene er lik ***0***).

For eksempel, dersom input til funksjonen er matrisen **A** nedenfor, så skal output være matrisen **D**.

Et matriseelement ligger langs diagonalen dersom i = j. Eksempel:

$$A = \begin{bmatrix} 10 & 2 & 4 & 9 \\ 3 & -5 & 15 & 8 \\ 1 & 7 & 6 & 3 \\ 17 & 0 & 17 & 10 \end{bmatrix} \rightarrow D = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & -5 & 0 & 0 \\ 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix}$$

```
import numpy as np

def get_diagonal(A):
    rows, cols = np.shape(A)
    A_diag = np.zeros([rows, cols])

    for i in range(rows):
        for j in range(cols):
            if i==j:

                Velg alternativ (A_diag[i, j] = A[i, j], A_diag[i, j] = A[rows, cols], A_diag[j, j] = A_diag[i, i], A[i, j] = A_dialg[i, j])
    return A_diag
```

Maks poeng: 4

6 Analyse av listedata

Lag en funksjon **bmi_statistikk** som kan ta inn en liste med persondata (høyde og vekt), og returnerer andelen av personene som har en kroppsmasseindeks (BMI) som er over en viss terskel i prosent.

Formel for utregning av BMI (høyde er gitt i meter):

$$\text{BMI} = \frac{\text{vekt}}{\text{høyde}^2}$$

Eksempel på data:

```
bmi_data= [[180, 90],
           [195, 92],
           [165, 66]]
terskel = 25
```

I tabellen representerer hver rad en person, der første kolonne er høyde målt i centimeter (cm) og andre kolonne er vekt målt i Kg.

Eksempel på funksjonskall (gitt at variabelen **bmi_data** og **terskel** er definert som ovenfor):

```
andel = bmi_statistikk(bmi_data, terskel)
print(f"Andel av personene med BMI over {terskel}: {andel:.2f}%")
```

Skriver ut: **Andel av personene med BMI over 25: 33.33%**

Skriv ditt svar her

```
1
```

Maks poeng: 5

7 Numerisk derivasjon

Vi måler følgende verdier for høyden til en drone:
$h = [153, 155, 158, 163, 170, 178, 185, 189, 188, 183]$

Målingene er gjort ved følgende tidspunkt:
$t = [0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0]$

Vi ser bort fra enhetene til h og t.

Vi antar at dronen kun beveger seg i høyderetningen. Hva gir forroverdifferanse som en tilnærming til den tidsderiverte av høyden?

$v_{\text{forover}}(1.0) =$: ____

Hva gir bakoverdifferanse?

$v_{\text{bakover}}(1.0) =$ ____

Hva gir sentraldifferanse?

$v_{\text{sentral}}(1.0) =$ ____

Maks poeng: 6

8 Hvilken metode?

Finn hvilken numerisk metode som kan brukes til å løse hvilken oppgave.

Hvert svaralternativ passer til nøyaktig en oppgave.

A: Vi har målt hastigheten $v(t)$ og startposisjonen $s(t_{\text{start}})$ til en drone, og er interesserte i å vite posisjonen. Det vil si: Vi kjenner $v(t) = \frac{ds}{dt}$ for tiden t i $[t_{\text{start}}, t_{\text{slutt}}]$, og vil finne $s(t)$ for t i samme tidsrommet $[t_{\text{start}}, t_{\text{slutt}}]$.
B: Vi har målt posisjonen til en drone, og er interesserte i å finne hastigheten. Det vil si: Vi kjenner $s(t)$ for tiden i $[t_{\text{start}}, t_{\text{slutt}}]$, og vil finne $v(t) = \frac{ds}{dt}$ for tidsrommet $(t_{\text{start}}, t_{\text{slutt}})$.
C: En drone mister motorkraften, og faller fritt. Vi vet at $\frac{dv}{dt} = g - \frac{kv^2}{m}$. Vi er interesserte i å finne $v(t)$. (g, k og m er konstante tall vi vet.)
D: Vi er gitt at en drone har følgende høyde målt fra hustaket ved tiden t:
$h(t) = e^{-t} - 0.001t^2$. Vi er interesserte i ved hvilket tidspunkt høyden blir null, altså for hvilket tidpunkt t man får $h(t) = 0$.

Finn de som passer sammen:

	C	A	D	B
Newtons metode	○	○	○	○
Foroverdifferanse	○	○	○	○
Simpsons metode eller Eulers metode	○	○	○	○
Vi har ikke nok informasjon	○	○	○	○

Maks poeng: 4

9 Newtons metode

Bruk to iterasjoner av Newtons metode til å løse ligningen

$$\sin(x) + x = 2.0$$

Det kan være nyttig at $\frac{d}{dx}(\sin(x)) = \cos(x)$.

Bruk startgjetning $x_0 = 0.5$

Ett steg av Newtons metode gir : ☐

To steg av Newtons metode gir: ☐

Maks poeng: 6

10 Differensialligning

En sten faller med hastigheten $v(t)$, der $v(t)$ følger differensialligningen:

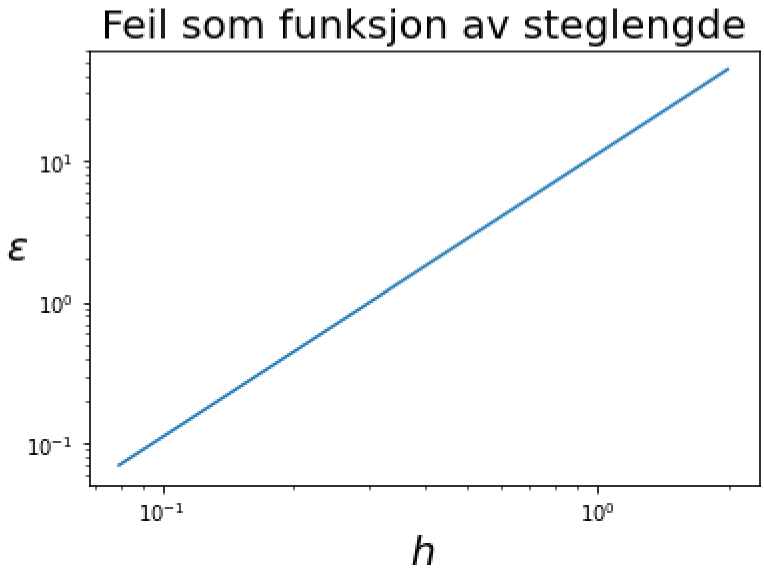
$$\frac{dv}{dt} = g - kv^2$$

Vi ser bort fra enheter og går ut fra $g = 9.81$ og $k = 0.15$. Ved $t = 0.0$ er hastigheten 3.0 det vil si $v(0.0) = 3.0$. I numerikkurset har dere lært en metode for å løse differensialligninger numerisk. Denne metoden står på formelarket. Bruk ett eller flere steg i denne metoden til å beregne en tilnærming for hastigheten ett tidels sekund senere. Det vi si, finn en tilnærming til $v(0.1)$.

$v(0.1) \approx$ ☐

Maks poeng: 5

11 Konvergensrate



Over ser dere et logaritmisk plot av feilen ϵ som en funksjon av steglengden h for en numerisk metode. Vi ser at $\epsilon = k \cdot h^a$, der k og a er konstanter. Det er oppgitt at a er et heltall.

Dere skal bestemme a ut fra grafen.

$a =$ ☐

Dersom vi bruker steglengde $h = 0.09$ får vi feilen $\epsilon = 0.09$. Hvilken steglengde h må vi velge for at feilen skal bli $\epsilon = 0.01$?

$h =$ ☐

Maks poeng: 5

12 Integrasjon

Vi ser på funksjonen:

$$g(x) = \sin(x)$$

Bruk Trapesmetoden med to delintervaller (tre punkter) til å finne en tilnærming til $I = \int_0^\pi g(x)\,dx$.

Bruk så Simpsons metode med to delintervaller (tre punkter) til å tilnærme I. Hva er avviket fra eksakt verdi? (Eksakt verdi er 2.) Hvilken metode er den mest nøyaktige?

Vi ser så på

$$h(x) = 123.45x^2 - 4.34526x + 325$$

Dersom du skal beregne $\int_0^\pi h(x)\,dx$ med tre punkter, vil trapesmetoden eller Simpsons metode gi minst feil? Trenger du å regne ut integralet for å svare på spørsmålet? Begrunn svaret.

Skriv ditt svar her

Format

Words: 0

Maks poeng: 7

13 Numerisk derivasjon

Gitt en matematisk funksjon

$$f(x) = (x + 3.5)^3 + 2 \cdot x^2 - 10$$

Fullfør programmet nedenfor slik at det regner ut en tilnærming til funksjonens deriverte for intervallet
$x = 0$ t.o.m $x = 4$ med bruk av senterdifferanse med skrittlengde $h = 0.02$, og plotter $f'(x)$ som funksjon av x i intervallet $0 \leq x \leq 4$.

```
import numpy as np
import matplotlib.pyplot as plt

def f(x):

    Velg alternativ ((x + 3.5)**3 + 2*x**2 - 10, y = (x + 3.5)**3 + 2*x**2 - 10, f(x) = (x + 3.5)**3 + 2*x**2 - 10, y_der = (x + 3.5)**3 + 2*x**2 - 10)
    return y

h = 0.02

x = Velg alternativ (np.linspace(0, 4 + h, h), np.arange(0, 4, h), np.linspace(0, 4, h), np.arange(0, 4 + h, h))

f_der = Velg alternativ ((f(x+h)-f(x))/h*2, f(x+h) - f(x-h)/(h+h), (f(x+h)-f(x-h))/2*h, (f(x+h)-f(x-h))/(2*h))

Velg alternativ (plt.plot(f_der), plt.plot(f_der, x), plt.plot(f, f_der), plt.plot(x, f_der))
```

Maks poeng: 6

14 Numerisk Integrasjon med Trapesmetoden

Fullfør koden nedenfor slik at den bruker trapesmetoden til å beregne integralet av:

$f(x) = x^2$

fra 0 til 1 med $n = 10$ delintervall.

```
import numpy as np
def f(x):

    Velg alternativ (y = 2*x, y = x**2, y = 2*x**2, y = f(x))
    return y

a = 0
b = 1
n = 10
h = (b - a) / n

x = Velg alternativ (linspace(n+h/2, n-h/2), linspace(a, b, n),
np.linspace(a + h/2, b - h/2), np.linspace(a, b, n + 1))
y = f(x)

integral = Velg alternativ ((h / 2) * (y[0] + sum(y[1:-1]) + y[-1]),
sum(y[1:-1]), (h / 2) * (y[0] + 2 * sum(y[1:-1]), (h / 2) * (y[0] + 2 *
sum(y[1:-1]) + y[-1]))
```

Maks poeng: 6

15 # Eulers metode

Fullfør koden nedenfor slik at den bruker Eulers metode til å løse differensialligningen

$$y' = y - x^2 + 1$$

med initialverdien $y(0) = 0.5$ og skrittlengde $h = 0.5$ for x fra 0 til 1.

```
import numpy as np

def f(x, y):

    Velg alternativ (return y' + x**2 - 1, return y - x*2 + 1, return y - x**2 +
1, y - x**2 + 1)

h = 0.5

x = Velg alternativ (np.arange(0, 1 + h, h), np.arange(h, 1 + h, h),
arange(0, 1 + h, h), np.arange(0, 1, h))
y = np.zeros(len(x))
y[0] = 0.5

for i in range(1, len(x)):

    y[i] = Velg alternativ (h * f(x[i-1], y[i-1]) - y[i-1], y[i+1] + h * f(x[i-1],
y[i+1]), y[i-1] + f(x[i-1], y[i-1])/h, y[i-1] + h * f(x[i-1], y[i-1]))
```

Maks poeng: 7.5

16 Fikspunktiterasjon

Gitt ligningen

$(4 - x)^2 = 4$

Fullfør programkoden nedenfor slik at den bruker *fikspunktiterasjon* med initialverdi $x_0 = 0$til å finne en løsning til ligningen med maksimalt avvik på 10^{-6}.

----

def g(x):

return Velg alternativ **((x**2+12)/8, (x**2+12), (x**2+8)/12, (x**2+16))**

x = 0

while Velg alternativ **(abs(g(x)) > 1e-6, abs(g(x)-x) > 1e-6, abs(x) < 1e-6, abs(g(x)-x) < 1e-6):**

Velg alternativ **(x = g(x), x = x - g(x)/g_deriv(x), x += g(x), x += 1e-6)**

print(f"Tilnærmet løsning funnet: x = {x}")

Maks poeng: 7.5

17 Absolutt og relativ feil

I numerikk bruker vi ofte *absolutt feil* og *relativ feil* til å evaluere nøyaktigheten til en numerisk beregning. Gitt en eksakt tallverdi y og en tilnærmet tallverdi x, vil den matematiske definisjonen på absolutt og relativ feil være:

$$\text{absolutt feil} = |x - y|$$

$$\text{relativ feil} = \left|\frac{x-y}{y}\right|$$

Funksjonen nedenfor har som oppgave å sjekke om to tall **x** og **y** er tilnærmet like gitt en absolutt toleranse **atol** og relativ toleranse **rtol**. Funksjonen skal returnere **True** dersom $\text{absolutt feil} \leq \texttt{atol}$ *eller* dersom $\text{relativ feil} \leq \texttt{rtol}$, og returnere **False** for alle andre tilfeller.

Fullfør koden til funksjonen **is_approx_equal** slik at den fungerer som beskrevet over, samtidig som den unngår "ulovlige" matematiske regneoperasjoner (dvs. regneoperasjoner som gir feilmelding i Python).

def is_approx_equal(x, y, rtol, atol):

if Velg alternativ **(abs(x - y) <= max(atol, abs(y)*rtol), abs((x - y)/y) <= rtol and abs(x - y) <= atol, [(x - y)/y] >= rtol and [x - y] >= atol, abs(x - y) <= min(atol, abs(y)*rtol)):**
return True
else:
return False

Maks poeng: 7