NTNU

elsys
ELEKTRONISK
SYSTEMDESIGN
& INNOVASJON

Matte obligatorisk

Tittel: Standardprosjekt	
Forfattere: Joakim Enes	
Versjon: 1.0	Dato: 09.04.2025

Innhold

1 Oppgave 1 **1**

2 Oppgave 2 **2**

3 Oppgave 3 **4**

4 Oppgave 4 **6**

5 Oppgave 5 **8**

Referanser **10**

---

1 Oppgave 1

Vi skal undersøke hvor liten verdi h kan ha, basert på Ligning 1 som er stigningstallet til sekanten til f mellom punktene x og x+h

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{1}$$

Vi bruker dette i python og undersøker når hvor mange desiamler h kan ha. Dette vises i Figur 1

```python
import numpy as np

x = 1.5
def funksjon(parameter):
    return np.e**parameter

def derivert(funksjon, h):
    return (funksjon(x+(1/10**h))-funksjon(x))/(1/10**h)

for i in range(1, 20):
    print(f" indeks {i} : {derivert(funksjon, i)}")
```

[25] ✓ 0.0s

```
indeks 1 : 4.713433540570504
indeks 2 : 4.5041723976187775
indeks 3 : 4.483930662007474
indeks 4 : 4.481913162264206
indeks 5 : 4.4817114789097445
indeks 6 : 4.481691310509461
indeks 7 : 4.481689295232627
indeks 8 : 4.481689064306238
indeks 9 : 4.481689686031132
indeks 10 : 4.481686133317453
indeks 11 : 4.481659487964862
indeks 12 : 4.481748305806832
indeks 13 : 4.476419235288631
indeks 14 : 4.440892098500626
indeks 15 : 5.329070518200751
indeks 16 : 0.0
```

Figur 1: Kode for undersøkelse av når h feiler

Vi ser at den feiler ved indeks 11, hvor stigningstallet er høyere i neste indeks. Vi ser også at indeks 16 gir 0, som er en vanlig python feil, fordi vi opererer med for små tall iforhold til hva programmet takler.

2 Oppgave 2

Vi antar prøver nå å løse oppgaven på en litt annerledes måte. Vi antar at f er analytisk og taylorutvikler emphf om punktet x. Vi kan da bruke **??**.

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x-h)}{2h} \tag{2}$$

Vi bruker python og ser resultatene i Figur 2

```python
import numpy as np

x = 1.5
eksakt_derivert = np.exp(1.5)
def funksjon(parameter):
    return np.exp(parameter)

def derivert(funksjon, h):
    return (funksjon(x+(1/10**h))-funksjon(x - 1/10**h))/(2 * (1/10**h))

for i in range(1, 20):
    print(f" indeks {i} : {derivert(funksjon, i):.10f}     h = {1/10**i}     feil = {abs(derivert(funksjon, i) - eksakt_derivert)}")

```

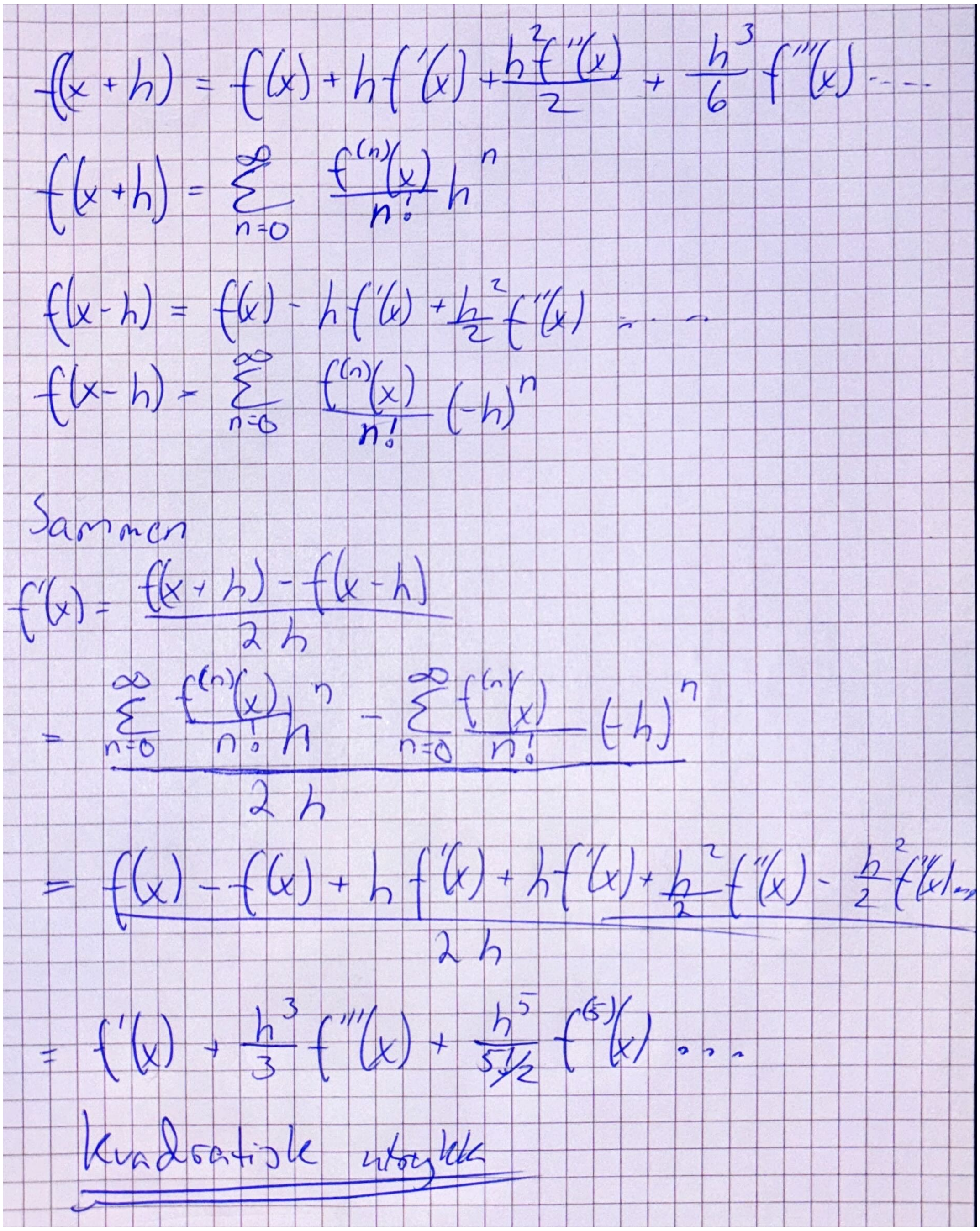
```
✓ 0.0s
```

```
indeks 1 : 4.4891622878     h = 0.1     feil = 0.007473217414137423
indeks 2 : 4.4817637655     h = 0.01     feil = 7.469519133618263e-05
indeks 3 : 4.4816898173     h = 0.001     feil = 7.469480740596168e-07
indeks 4 : 4.4816890778     h = 0.0001     feil = 7.468485385686563e-09
indeks 5 : 4.4816890704     h = 1e-05     feil = 9.660450217552352e-11
indeks 6 : 4.4816890701     h = 1e-06     feil = 2.5866686570452657e-10
indeks 7 : 4.4816890732     h = 1e-07     feil = 2.8499576032459117e-09
indeks 8 : 4.4816890199     h = 1e-08     feil = 5.04407475787616e-08
indeks 9 : 4.4816892419     h = 1e-09     feil = 1.716038573462697e-07
indeks 10 : 4.4816905742     h = 1e-10     feil = 1.5038714868964576e-06
indeks 11 : 4.4817038969     h = 1e-11     feil = 1.4826547782398336e-05
indeks 12 : 4.4821923950     h = 1e-12     feil = 0.0005033246786174672
indeks 13 : 4.4808601274     h = 1e-13     feil = 0.0008289429509327206
indeks 14 : 4.4853010195     h = 1e-14     feil = 0.0036119491475679055
indeks 15 : 4.8849813084     h = 1e-15     feil = 0.40329223801262426
indeks 16 : 0.0000000000     h = 1e-16     feil = 4.4816890703380645
indeks 17 : 0.0000000000     h = 1e-17     feil = 4.4816890703380645
indeks 18 : 0.0000000000     h = 1e-18     feil = 4.4816890703380645
indeks 19 : 0.0000000000     h = 1e-19     feil = 4.4816890703380645
```

Figur 2: Kode for undersøkelse av når h feiler

Vi observerer igjen at ved indeks 15 at det går skeis. Dette er trolig på grunn av datamaskinen.

Vi ser sammenhengen med taylor rekke i feilmarginen som ser kvadratisk ut, frem til indeks 5. Dette kan vi bevise ved følgene utregning Figur 3:



Figur 3: Utledning av hvorfor feilmarginen er kvadratisk

Vi ser at Ligning 2 sin utledning blir kvadratisk med oddetalls potenser.

3 Oppgave 3

For å gå litt overkill for samme problemstilling, kan man bruke Ligning 3

$$
f'(x) = \frac{f(x-2h) - 8f(x-h) + 8f(x+h) - f(x+2h)}{12h} \tag{3}
$$

Denne tilnærmingen vil være mye mer presis, for samme h enn tilnærmingene av *f'(x)* fra oppgave 1 og 2. Koden for denne ligge i Figur 4

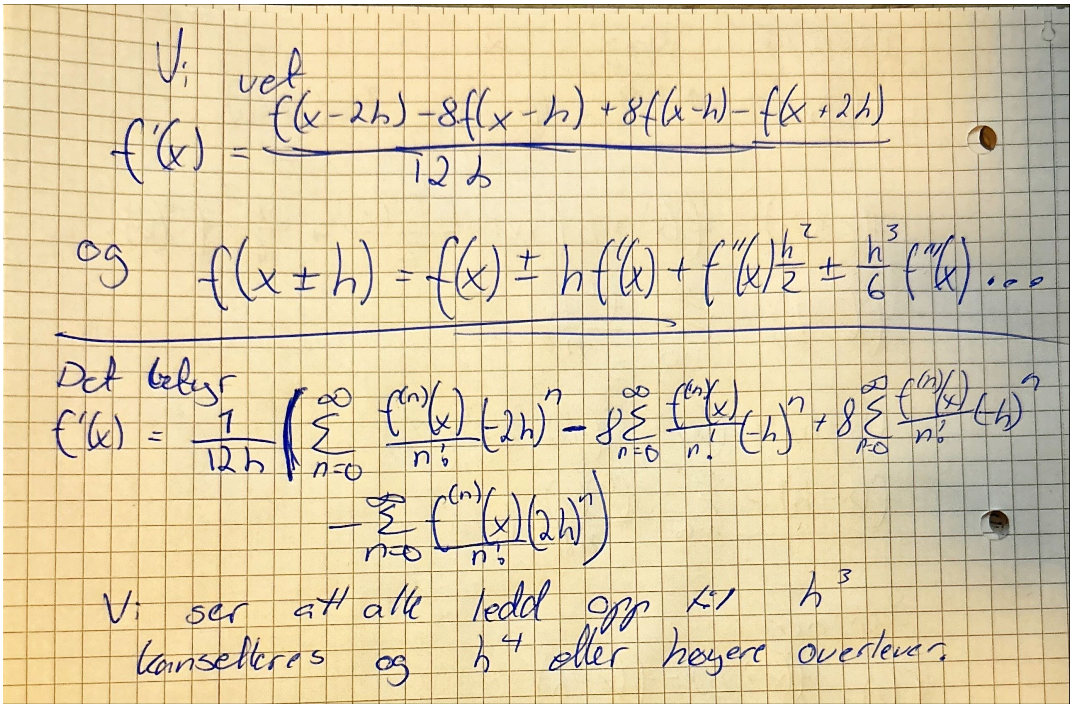
```python
import numpy as np

x = 1.5
eksakt_derivert = np.exp(1.5)
def f(parameter):
    return np.exp(parameter)

def derivert(f, inp):
    h = 1/10**inp
    return (f(x - 2*h) - 8*f(x-h) +8*f(x+h) - f(x + 2 * h))/(12 * h)

for i in range(1, 20):
    print(f" indeks {i} : {derivert(f, i):.10f}     h = {1/10**i}     feil = {abs(derivert(f, i) - eksakt_derivert)}")

```

```
[3]  ✓ 0.0s
```

```
indeks 1 : 4.4816741136     h = 0.1     feil = 1.4956758427331351e-05
indeks 2 : 4.4816890688     h = 0.01     feil = 1.4938787984419832e-09
indeks 3 : 4.4816890703     h = 0.001     feil = 3.552713678800501e-13
indeks 4 : 4.4816890703     h = 0.0001     feil = 3.8458125573015423e-13
indeks 5 : 4.4816890704     h = 1e-05     feil = 5.219469301209756e-11
indeks 6 : 4.4816890700     h = 1e-06     feil = 3.326814379533971e-10
indeks 7 : 4.4816890739     h = 1e-07     feil = 3.590106878448296e-09
indeks 8 : 4.4816889977     h = 1e-08     feil = 7.264520895944315e-08
indeks 9 : 4.4816893900     h = 1e-09     feil = 3.196335933708383e-07
indeks 10 : 4.4816920545     h = 1e-10     feil = 2.9841688533593924e-06
indeks 11 : 4.4817186999     h = 1e-11     feil = 2.962952144436315e-05
indeks 12 : 4.4825624694     h = 1e-12     feil = 0.0008733990201585939
indeks 13 : 4.4801199787     h = 1e-13     feil = 0.0015690916340167504
indeks 14 : 4.4853010195     h = 1e-14     feil = 0.0036119491475679055
indeks 15 : 5.1070259133     h = 1e-15     feil = 0.6253368429376556
indeks 16 : -1.4802973662     h = 1e-16     feil = 5.96198643650494
indeks 17 : -7.4014868308     h = 1e-17     feil = 11.883175901172441
indeks 18 : -74.0148683083     h = 1e-18     feil = 78.49655737868183
indeks 19 : -740.1486830834     h = 1e-19     feil = 744.6303721537759
```

Figur 4: Kode for mer presis *f'(x)*

Vi observerer fra indeks 1 - 3, at h er 4 desiamler mer presis enn fra oppgave 1. Dette kan vi utlede her Figur 5.



Figur 5: Utledning for mer presis *f'(x)*

4 Oppgave 4

Løs varmeligningen med eksplisitt skjema og initial krav $u(x, 0) = \sin(x)$. Prøv forskjellige kombinasjoner av k og h. Lag animasjon for å se om koden fungerer.

Viktig informasjon for denne oppgaven finnes i [1].

Fra den eksplisitte omformer vi den slik at vi får $u_{i,j+1}$ på venstre side. Dette gir oss Ligning 4

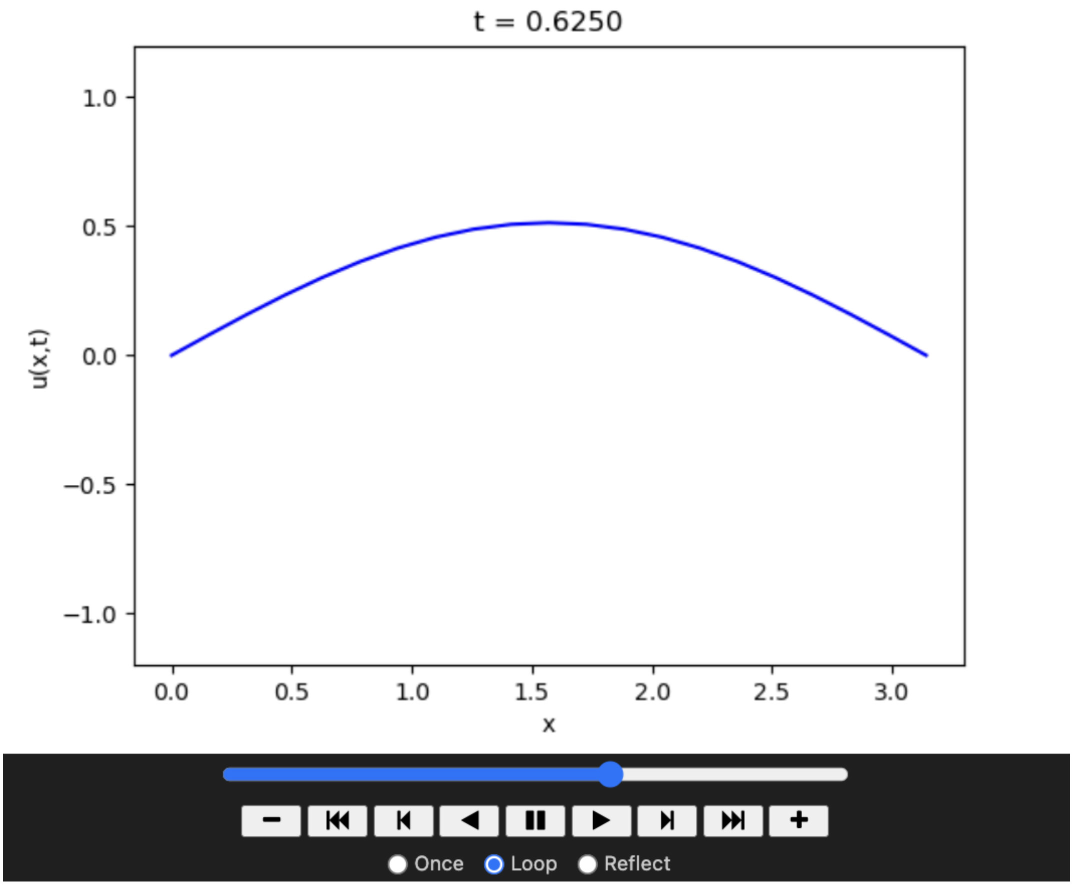
$$u_{i,j+1} = \frac{k(u_{i+1,j} - 2u_{i,j} + u_{i-1,j})}{h^2} + u_{i,j} \tag{4}$$

Vi lager et skript for å plotte dette som en animasjon gitt oppgave beskrivelsen. Dette vises i Figur 6.

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

def varmeligningen(N, M):
    tid = 1
    avstand = np.pi
    h = avstand / N
    k = tid / M
    alpha = k / h**2

    x_verdier = np.linspace(0, avstand, N+1)
    t_verdier = np.linspace(0, tid, M + 1)

    u = np.zeros((N+1, M+1))
    u[:, 0] = np.sin(x_verdier)

    for j in range(0, M):
        for i in range(1, N):
            u[i, j +1] = alpha * (u[i+1, j] - 2*u[i, j] + u[i -1, j]) + u[i, j]

            u[0, j + 1] = 0
            u[N, j + 1] = 0
    return u, x_verdier, t_verdier, k

u, x_verdier, t_verdier, k = varmeligningen(20, 8)

fig, ax = plt.subplots()
line, = ax.plot(x_verdier, u[:, 0], color='blue')
ax.set_ylim(-1.2, 1.2)
ax.set_title("Varmelikningen – eksplisitt løsning")
ax.set_xlabel("x")
ax.set_ylabel("u(x,t)")

def oppdater(bilde):
    line.set_ydata(u[:, bilde])
    ax.set_title(f"t = {t_verdier[bilde]:.4f}")
    return line,


ani = animation.FuncAnimation(fig, oppdater, frames=len(t_verdier), interval=200, blit=True)
plt.close()

ani
from IPython.display import HTML
HTML(ani.to_jshtml())
```

Figur 6: Eksplisitt kode for varmeligningsoppgave

Dette gir oss et animasjons vindu som viser hvordan ligningen utvikler seg over et sekund. For å se dette, må du kjøre koden selv, men legger til bilde av vinduet under: Figur 7



Figur 7: Eksplisitt animasjon

Ved testing av flere ulike alfa eller $\frac{k}{h^2}$ finner vi ut at det er en kritisk verdi som må overholdes for å unngå en eksplosjon i grafen. det er $\frac{k}{h^2} < \frac{1}{2}$.

5 Oppgave 5

For den implisitte løsningen må vi formulere utrykket på en litt annen måte, som gjør at den blir slik Ligning 5.

$$u_{i,j+1} = \frac{u_{i+1,j+1} + u_{i-1,j+1} + \alpha u_{i,j}}{1 + 2\alpha} \tag{5}$$

Vi bruker denne ligningen og skriver kode for å lage animasjonen nokså likt som fra oppgave 4.

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

def varmeligningen_implisitt(N, M):
    tid = 1
    avstand = np.pi
    h = avstand / N
    k = tid / M
    alpha = k / h**2

    x_verdier = np.linspace(0, avstand, N+1)
    t_verdier = np.linspace(0, tid, M + 1)

    u = np.zeros((N+1, M+1))
    u[:, 0] = np.sin(x_verdier)

    for j in range(0, M):
        ny = u[:, j].copy()
        for i in range(1, N):
            ny[i] = (u[i + 1, j] + u[i - 1, j] + alpha * u[i, j]) / (1 + 2 * alpha)

        ny[0] = 0
        ny[N] = 0

        u[:, j + 1] = ny

    return u, x_verdier, t_verdier, k

u, x_verdier, t_verdier, k = varmeligningen_implisitt(20, 8)

fig, ax = plt.subplots()
line, = ax.plot(x_verdier, u[:, 0], color='blue')
ax.set_ylim(-1.2, 1.2)
ax.set_title("Varmelikningen □ implisitt løsning")
ax.set_xlabel("x")
ax.set_ylabel("u(x,t)")

def oppdater(bilde):
    line.set_ydata(u[:, bilde])
    ax.set_title(f"t = {t_verdier[bilde]:.4f}")
    return line,


ani = animation.FuncAnimation(fig, oppdater, frames=len(t_verdier), interval=200, blit=True)
plt.close()

ani
from IPython.display import HTML
HTML(ani.to_jshtml())
```

Figur 8: Implisitt kode

Vi får tilnærmet lik animasjon som fra oppgave 4.

Referanser

[1] NTNU, *OBLIG - TMA4106*, Kursmateriell til emne TMA4106, 2025.