Matte 2 oblig Leah Kavli Mærde, gjort med Jenny Fonstad, 23.03.2025

Har prøvd på programmeringen, men veldig vanskelig!

Oblig Matte 2.

$\boxed{1}$ $$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

$$f(x) = e^x$$

$h = 0{,}01$

$$f'(1{,}5) \approx \frac{e^{1{,}51} - e^{1{,}5}}{0{,}01} \approx 4{,}5042$$

Bommer med ca $2 \cdot 10^{-2}$

$h = 0{,}001$

$$f'(1{,}5) \approx \frac{e^{1{,}501} - e^{1{,}5}}{0{,}001} \approx 4{,}4839$$

Bommer med ca $2 \cdot 10^{-3}$

$h = 0{,}0001$

$$f'(1{,}5) \approx \frac{e^{1{,}5001} - e^{1{,}5}}{0{,}0001} \approx 4{,}4819$$

Bommer med ca $2 \cdot 10^{-4}$

$h = 0{,}00001$

$$f'(1{,}5) \approx \frac{e^{1{,}50001} - e^{1{,}5}}{0{,}00001} \approx 4{,}4817$$

Lik $f'(1{,}5) = e^{1{,}5} = 4{,}4817$

$h = 0{,}000001$

$$f'(1{,}5) \approx \frac{e^{1{,}500001} - e^{1{,}5}}{0{,}000001} \approx 4{,}4817$$

Igjen lik

Matte 2 oblig Leah Kavli Mærde, gjort med Jenny Fonstad, 23.03.2025

$h = 0{,}0000001$

$$f'(1{,}5) \approx \frac{e^{1{,}5000001} - e^{1{,}5}}{0{,}0000001} \approx 4{,}4817$$

Lik igjen

$h = 0{,}00000001$

$$f'(1{,}5) \approx \frac{e^{1{,}50000001} - e^{1{,}5}}{0{,}00000001} \approx 4{,}4817$$

Lik igjen

$h = 0{,}000000001$

$$f'(1{,}5) \approx \frac{e^{1{,}500000001} - e^{1{,}5}}{0{,}000000001} \approx 4{,}4817$$

Lik igjen

$h = 0{,}0000000001$

$$f'(1{,}5) \approx \frac{e^{1{,}5000000001} - e^{1{,}5}}{0{,}0000000001} \approx 4{,}4817$$

Lik igjen

$h = 0{,}00000000001$

$$f'(1{,}5) \approx \frac{e^{1{,}50000000001} - e^{1{,}5}}{0{,}00000000001} \approx 4{,}482$$

Kalkulator vil ikke va med flere desimaler

$h = 0{,}000000000001$

$$f'(1{,}5) \approx \frac{e^{1{,}500000000001} - e^{1{,}5}}{0{,}000000000001} \approx 4{,}49$$

Kalkulatoren kutter enda et desimal

$h = 0{,}0000000000001$

$$f'(1{,}5) \approx \frac{e^{1{,}5000000000001} - e^{1{,}5}}{0{,}0000000000001} \approx 0$$

Her går det til skogs

Matte 2 oblig Leah Kavli Mærde, gjort med Jenny Fonstad, 23.03.2025

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

h=0.0000000000000001
x=1.5

f_der=(np.exp(x-2*h)-8*np.exp(x-h)+8*np.exp(x+h)-np.exp(x+2*h))/(12*h)

print(f_der)
```

Matte 2 oblig Leah Kavli Mærde, gjort med Jenny Fonstad, 23.03.2025

Endret på uttrykket i koden for oppgave 2 og 3

$\boxed{2}$

$$\frac{f(x+h)-f(x-h)}{2h} = f'(x)$$

$h = 0{,}1$

$$f'(1{,}5) = \frac{f(1{,}5+0{,}1) - f(1{,}5-0{,}1)}{2 \cdot 0{,}1} \approx 4{,}4892$$

Beregninger i python

$h = 0{,}1 \approx 4{,}4892$

$h = 0{,}01 \approx 4{,}4818$

$h = 0{,}001 \approx 4{,}4817$

$h = 0{,}0001 \approx 4{,}4817$

$h = 0{,}00001 \approx 4{,}4817$

$h = 0{,}000001 \approx 4{,}4817$

$h = 0{,}0000001 \approx 4{,}4817$

$h = 0{,}00000001 \approx 4{,}4817$

$h = 0{,}000000001 \approx 4{,}4817$

$h = 0{,}0000000001 \approx 4{,}4817$

$h = 0{,}00000000001 \approx 4{,}4817$

$h = 0{,}000000000001 \approx 4{,}4822$

$h = 0{,}0000000000001 \approx 4{,}4809$

BRUNNEN

$h = 0{,}00000000000001 \approx 4{,}4853$

Matte 2 oblig Leah Kavli Mærde, gjort med Jenny Fonstad, 23.03.2025

$h = 0{,}0000000000000001 \approx 4{,}8850$

$h = 0{,}00000000000000001 \approx 0{,}0$

Går til skauer ved $h = 0{,}00000000000000001$.

Tilnærmingen blir bedre ved formlen

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

Dette kan forklares med taylor og feilleddet.

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$ gir taylor:

$$f'(x) = f'(x) + \frac{f''(x)}{2} h + \frac{f'''(x)}{6} h^2 + \dots$$

Det dominerende feilleddet

$\frac{f''(x)}{2} h$ er proposjonal med h og gir oss at feilen blir 10 ganger mindre når h reduseres med 10

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$ gir taylor:

$$f'(x) = f'(x) + \frac{f'''(x)}{6} h^2$$

Det dominerende feilleddet $\frac{f'''(x)}{6} h^2$ er proposjonal med h^2 altså blir feilen 100 ganger mindre når h reduseres med 10

Denne er derne bedre.

Matte 2 oblig Leah Kavli Mærde, gjort med Jenny Fonstad, 23.03.2025

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

Beregnet verdier i python

$h = 0,1 \approx 4,4817$

$h = 0,01 \approx 4,4817$

$h = 0,001 \approx 4,4817$

$h = 0,0001 \approx 4,4817$

$h = 0,00001 \approx 4,4817$

$h = 0,000001 \approx 4,4817$

$h = 0,0000001 \approx 4,4817$

$h = 0,00000001 \approx 4,4817$

$h = 0,000000001 \approx 4,4817$

$h = 0,0000000001 \approx 4,4817$

$h = 0,00000000001 \approx 4,4817$

$h = 0,000000000001 \approx 4,4826$

$h = 0,0000000000001 \approx 4,4801$

$h = 0,00000000000001 \approx 4,4853$

$h = 0,000000000000001 \approx 5,1070$

$h = 0,0000000000000001 \approx -1,4803$

Her gikk det til skauen

Matte 2 oblig Leah Kavli Mærde, gjort med Jenny Fonstad, 23.03.2025

$\boxed{4}$ Vi er gitt formler

Eksplisitt $\dfrac{u_{i,j+1} - u_{i,j}}{k} = \dfrac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}$

For å beregne neste temeratur og dette er gitt ved leddet $u_{i,j+1}$

$u_{i,j+1} = u_{i,j} + \lambda(u_{i+1,j} - 2u_{i,j} + u_{i-1,j})$

der $\lambda = \frac{k}{h^2}$

Her er

$u_{i,j}$ nåværende temperatur

$u_{i+1,j}$ og $u_{i-1,j}$ neste og forrige punkt i griden

h = romsteg k = tidsteg

$u(x,0) = \sin(\pi)$ = initialbetingelse

$u(0,t) = 0$ og $u(1,t) = 0$ = Randbetingelsene

For stabilitet må

$\lambda = \frac{k}{h^2} \leq 0{,}5$

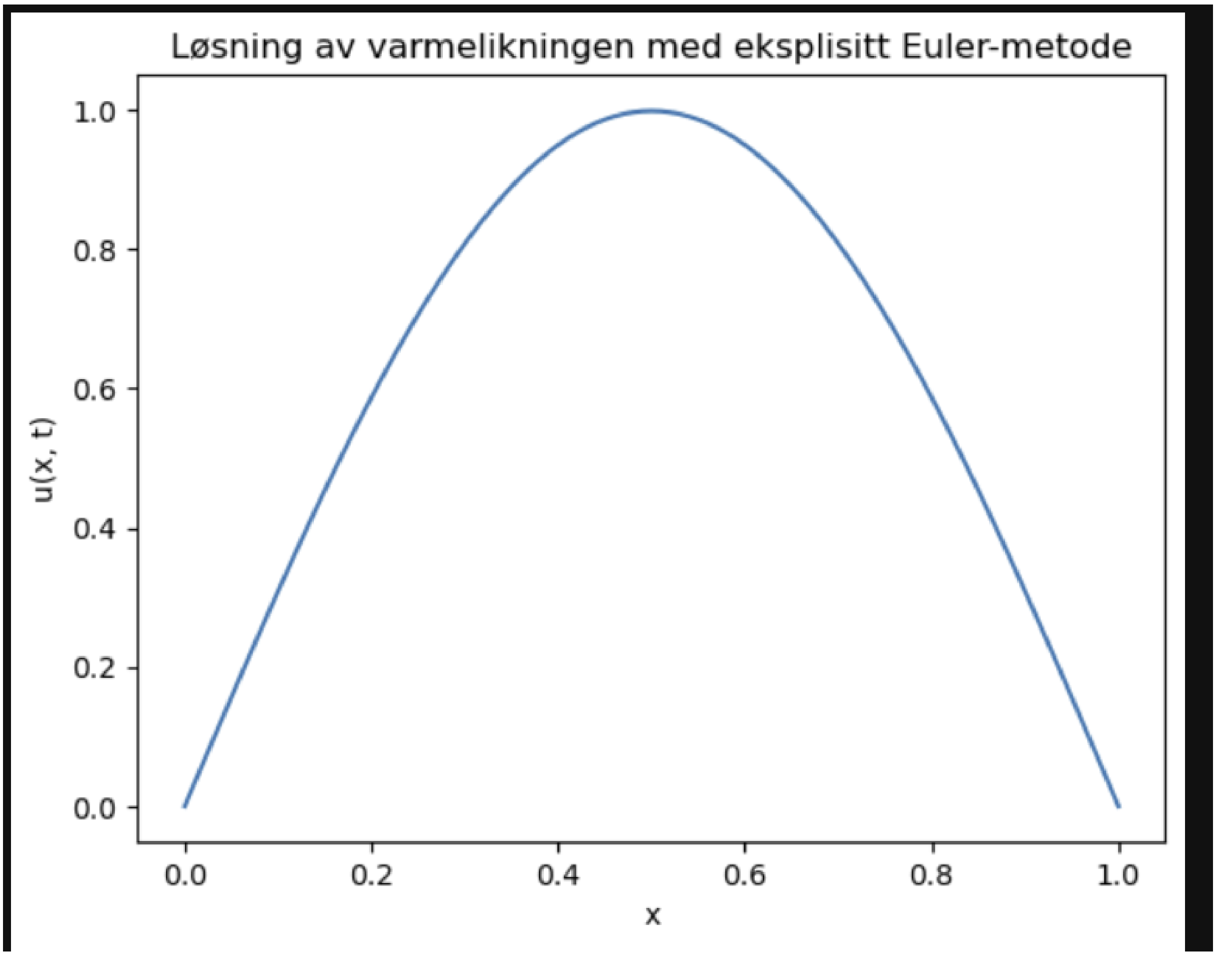
Programmerer i python.

prøvde litt forskjellige koder men usikker på animeringa.

Matte 2 oblig Leah Kavli Mærde, gjort med Jenny Fonstad, 23.03.2025

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

# Parametre
L = 1.0          # Lengde på stangen
T_max = 0.2      # Maksimal tid
h = 0.01        # Gitteravstand i x-retningen
k = 0.0001        # Gitteravstand i t-retningen
x = np.arange(0, L + h, h)  # Romgitterpunkter
n = len(x)       # Antall punkter i x
t = np.arange(0, T_max + k, k)  # Tidsgitterpunkter
m = len(t)       # Antall tidspunkter

# Initialbetingelse u(x, 0) = sin(x)
u = np.sin(np.pi * x)

# Animasjon
fig, ax = plt.subplots()
line, = ax.plot(x, u)

def update(frame):
    global u
    u_new = np.copy(u)
    for i in range(1, n - 1):
        u_new[i] = u[i] + (k / h**2) * (u[i + 1] - 2 * u[i] + u[i - 1])
    u = np.copy(u_new)
    line.set_ydata(u)  # Oppdaterer kurven
    return line,

ani = FuncAnimation(fig, update, frames=m, interval=50, blit=True)
plt.xlabel('x')
plt.ylabel('u(x, t)')
plt.title('Løsning av varmelikningen med eksplisitt Euler-metode')
plt.show()
```

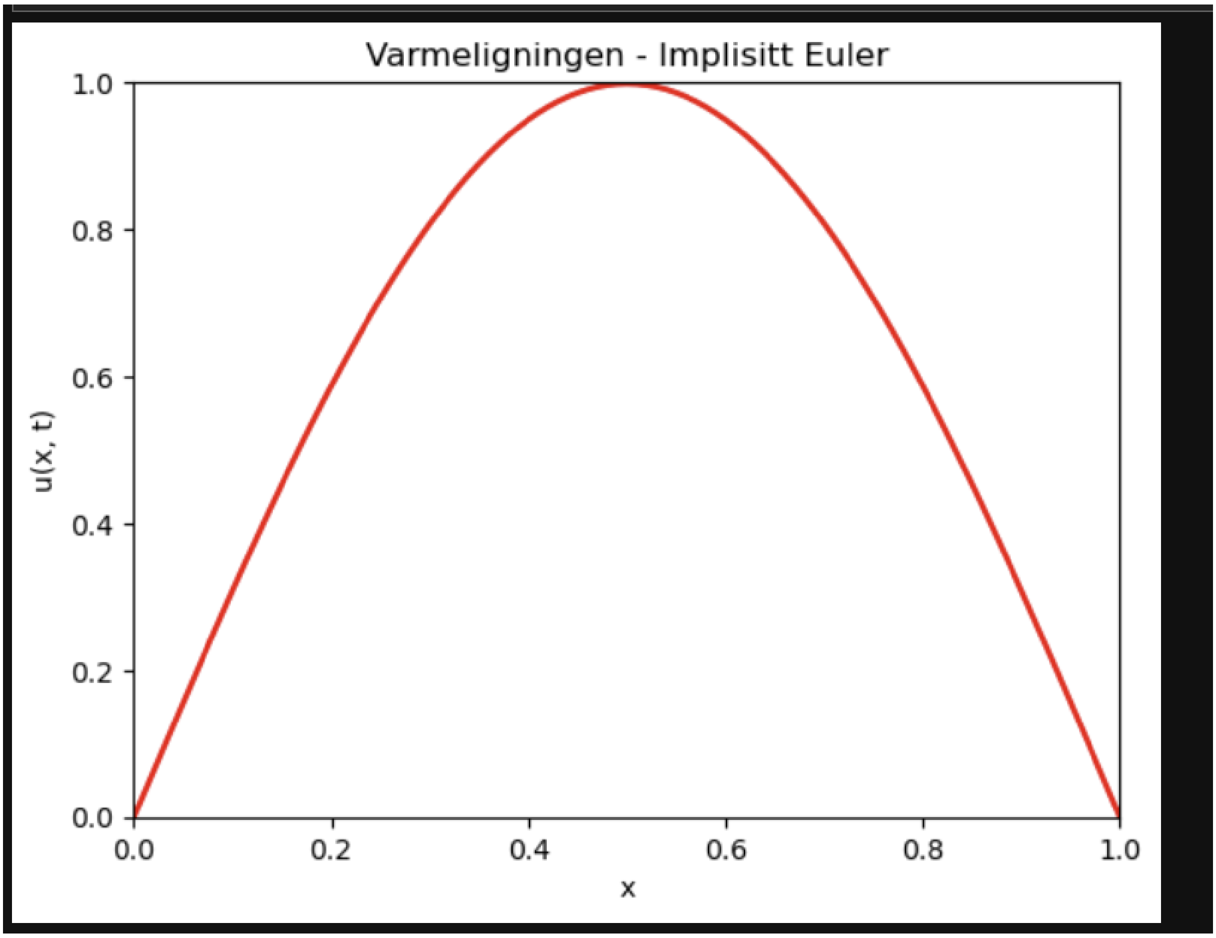
Matte 2 oblig Leah Kavli Mærde, gjort med Jenny Fonstad, 23.03.2025



Matte 2 oblig Leah Kavli Mærde, gjort med Jenny Fonstad, 23.03.2025

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

# Parametre
L = 1.0          # Lengde på stangen
T_max = 0.2      # Maksimal tid
h = 0.01         # Gitteravstand i x-retningen
k = 0.0001         # Tidssteg (ingen stabilitetsrestriksjon for implisitt metode)
x = np.arange(0, L + h, h)  # Romgitterpunkter
n = len(x)       # Antall punkter i x
t = np.arange(0, T_max + k, k)  # Tidsgitterpunkter
m = len(t)       # Antall tidspunkter

r = k / h**2  # Parameter for implisitt metode

# Initialbetingelse u(x, 0) = sin(pi*x), men setter negative verdier til 0
u = np.maximum(np.sin(np.pi * x), 0)
u[0] = u[-1] = 0  # Randbetingelser: u(0,t) = u(L,t) = 0

# === Opprett tridiagonalmatrisen A ===
A = np.zeros((n-2, n-2))
np.fill_diagonal(A, 1 + 2 * r)  # Hoveddiagonal
np.fill_diagonal(A[:-1, 1:], -r)  # Øvre diagonal
np.fill_diagonal(A[1:, :-1], -r)  # Nedre diagonal

# Animasjon
fig, ax = plt.subplots()
line, = ax.plot(x, u, 'r-', lw=2)
ax.set_xlim(0, L)
ax.set_ylim(0, 1.0)  # Setter minimum til 0

def update(frame):
    global u
    b = u[1:-1]  # Høyreside (u fra forrige tidssteg)
    u_new = np.zeros_like(u)
    u_new[1:-1] = np.linalg.solve(A, b)  # Løs ligningssystemet
    u_new = np.maximum(u_new, 0)  # Setter negative verdier til 0
    u = u_new
    line.set_ydata(u)
    return line,

ani = FuncAnimation(fig, update, frames=m, interval=50, blit=True)
plt.xlabel('x')
plt.ylabel('u(x, t)')
plt.title('Varmeligningen - Implisitt Euler')
plt.show()
```

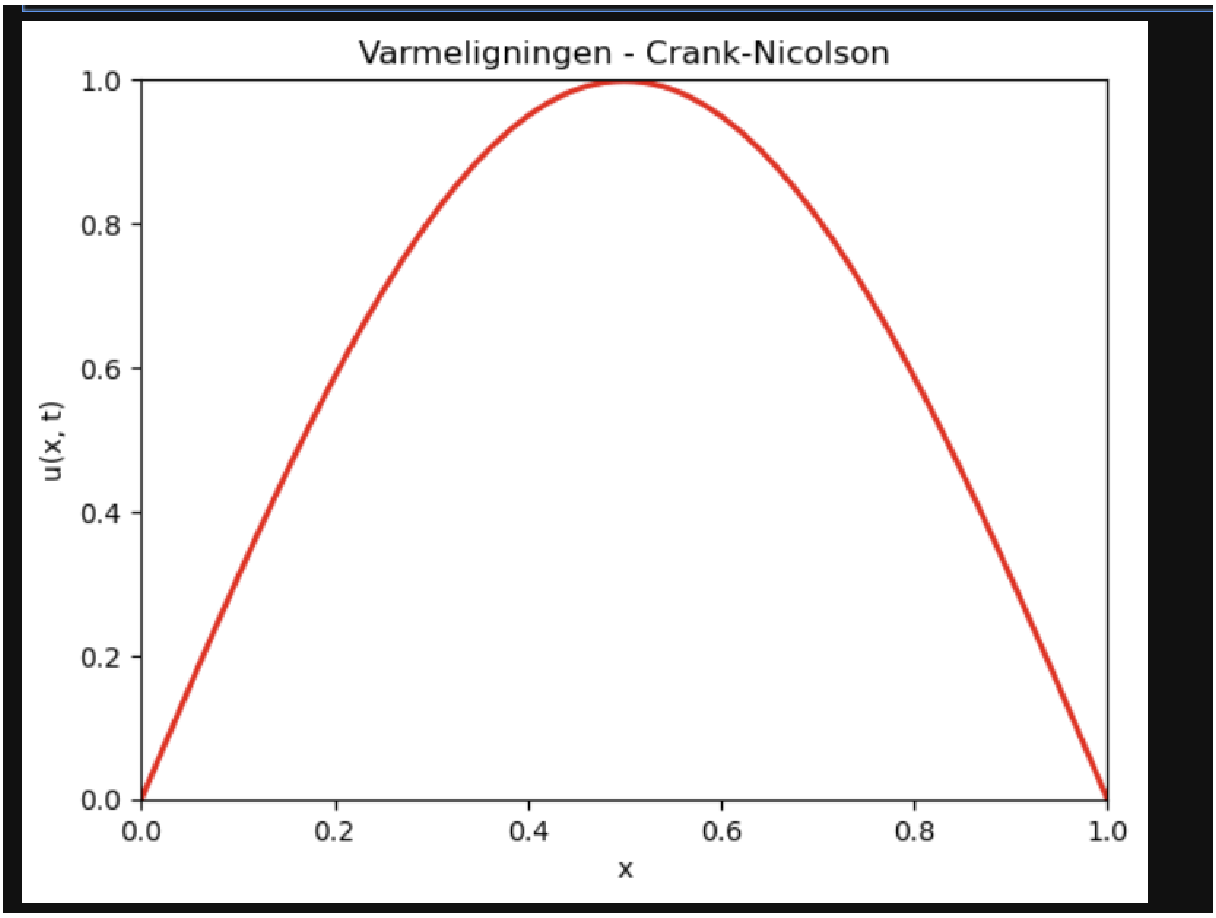
Matte 2 oblig Leah Kavli Mærde, gjort med Jenny Fonstad, 23.03.2025



Matte 2 oblig Leah Kavli Mærde, gjort med Jenny Fonstad, 23.03.2025

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

# Parametre
L = 1.0          # Lengde på stangen
T_max = 0.2      # Maksimal tid
h = 0.01        # Gitteravstand i x-retningen
k = 0.0001        # Tidssteg
x = np.arange(0, L + h, h)  # Romgitterpunkter
n = len(x)       # Antall punkter i x
t = np.arange(0, T_max + k, k)  # Tidsgitterpunkter
m = len(t)       # Antall tidspunkter

r = k / (2 * h**2)  # Crank-Nicolson parameter

# Initialbetingelse u(x, 0) = sin(pi*x), men setter negative verdier til 0
u = np.maximum(np.sin(np.pi * x), 0)
u[0] = u[-1] = 0  # Randbetingelser: u(0,t) = u(L,t) = 0

# === Konstruer matriser A og B for Crank-Nicolson ===
A = np.zeros((n-2, n-2))
B = np.zeros((n-2, n-2))

np.fill_diagonal(A, 1 + 2*r)  # Hoveddiagonal A
np.fill_diagonal(A[:-1, 1:], -r)  # Øvre diagonal A
np.fill_diagonal(A[1:, :-1], -r)  # Nedre diagonal A

np.fill_diagonal(B, 1 - 2*r)  # Hoveddiagonal B
np.fill_diagonal(B[:-1, 1:], r)  # Øvre diagonal B
np.fill_diagonal(B[1:, :-1], r)  # Nedre diagonal B

# Animasjon
fig, ax = plt.subplots()
line, = ax.plot(x, u, 'r-', lw=2)
ax.set_xlim(0, L)
ax.set_ylim(0, 1.0)  # Setter minimum til 0

def update(frame):
    global u
    b = B @ u[1:-1]  # Høyreside av ligningen
    u_new = np.zeros_like(u)
    u_new[1:-1] = np.linalg.solve(A, b)  # Løs tridiagonalt system
    u_new = np.maximum(u_new, 0)  # Fjerner negative verdier
    u = u_new
    line.set_ydata(u)
    return line,

ani = FuncAnimation(fig, update, frames=m, interval=50, blit=True)
plt.xlabel('x')
plt.ylabel('u(x, t)')
plt.title('Varmeligningen - Crank-Nicolson')
plt.show()
```

Matte 2 oblig Leah Kavli Mærde, gjort med Jenny Fonstad, 23.03.2025



Matte 2 oblig Leah Kavli Mærde, gjort med Jenny Fonstad, 23.03.2025

5 Med h verdi mye mindre enn k kræsjer programmet.
med lik verdi skjedde det ikke store endringer men topp punktet endret seg utifra hvor store verdiene var

Med k verdi mye mindre enn h er den ≈ uendret.

Forstår ikke helt hva som skjer men har prøvd.

5

Når h reduseres:
Vi får flere punkter i romretning og gir mer nøyaktighet
Mer stabil for større k-verdier.

Når k reduseres:
oppdaterer løsningen oftere, gir mer nøyaktig tidsutvikling av temp.
Ekstreme k-verdier er tidkrevende

Implisitt metode er stabil, men mer regne tung.

Matte 2 oblig Leah Kavli Mærde, gjort med Jenny Fonstad, 23.03.2025

6) H reduseres:

- Flere punkter i romretning, mer nøyaktig løsning
- Stabil

K reduseres:

- Mer detaljert tidsutvikling av løsningen, mer detaljert løsning over tid

Denne metoden gir god balanse mellom nøyaktighet og stabilitet, foretrukket. Bruker info fra nåværende og neste tidssteg og er mer nøyaktig.