Numerisk løsning av varmelikningen: Eksplisitte og implisitte metoder

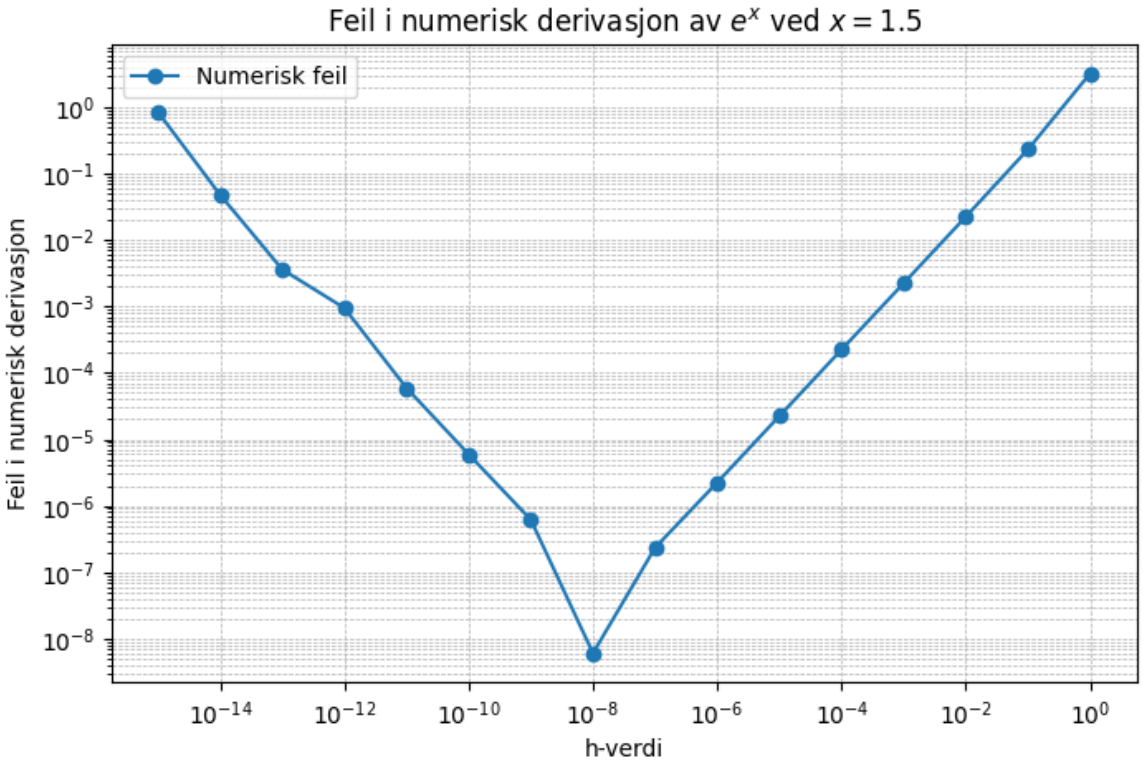
I denne notebooken skal jeg se på standardprosjektet i TMA4106.

In [ ]:
```
# Importering av nødvendige verktøy
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
from scipy.sparse import diags
from scipy.sparse.linalg import spsolve
```

Oppgave 1

- Bruk formelen: $f'(x) \approx \frac{f(x+h)-f(x)}{h}$
- $f(x) = e^x$
- Se hva som skjer når h nærmer seg null

In [67]:
```
def f(x):
    return np.exp(x)

def df(x): # dervierte funksjon
    return np.exp(x)

def df_numerisk(x, h):
    return (f(x + h) - f(x)) / h

x_val = 1.5
h_values = np.logspace(0, -15, 16)  # h-verdier fra 10^0 til 10^-15
errors1 = np.abs(df_numerisk(x_val, h_values) - df(x_val))  # Feil

# Plotter feilen
plt.figure(figsize=(8, 5))
plt.loglog(h_values, errors1, marker='o', linestyle='-', label="Num
plt.xlabel("h-verdi")
plt.ylabel("Feil i numerisk derivasjon")
plt.title("Feil i numerisk derivasjon av $e^x$ ved $x=1.5$")
plt.legend()
plt.grid(True, which="both", linestyle="--", linewidth=0.5)
plt.show()
```



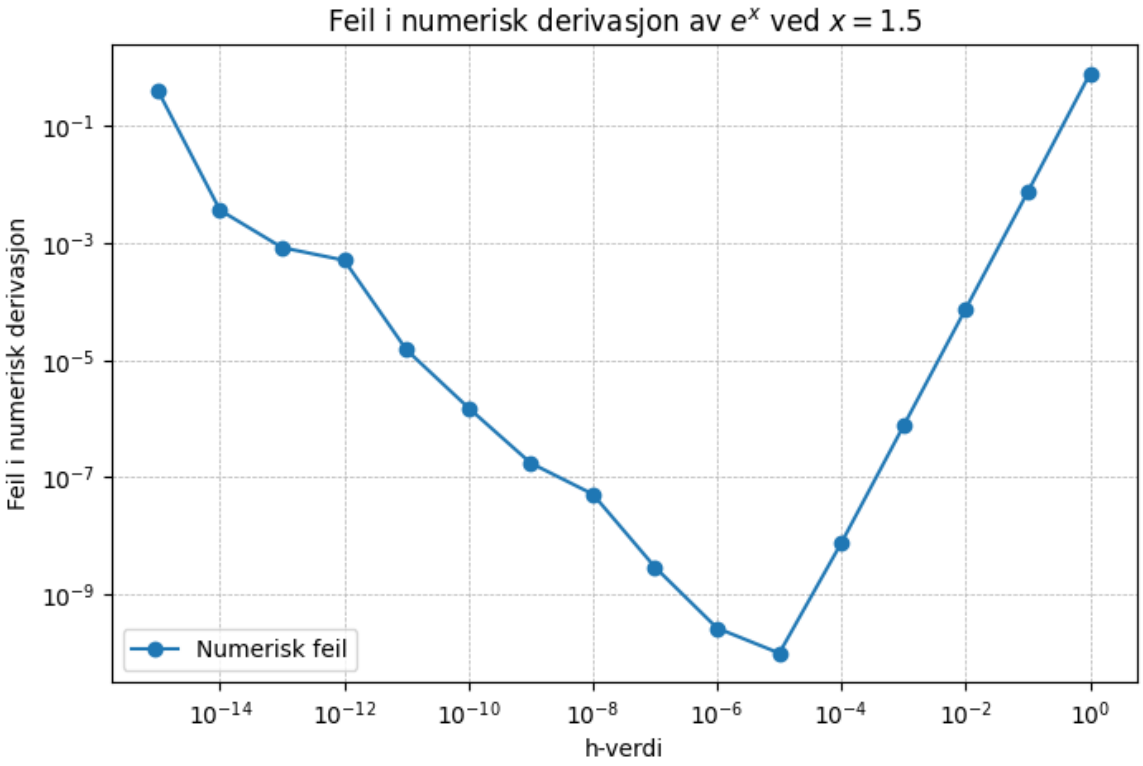
Kommentar til oppgave 1

Kan se at "det går åt skogen" når h-verdien er: $< 10^{-8}$, og man får en U (eller V) formet feilkurve.

Oppgave 2

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

In [54]:
```
def df_numerisk_v2(x, h):
    return (f(x_val+h_values)-f(x-h_values))/(2*h)


errors2 = np.abs(df_numerisk_v2(x_val, h_values) - df(x_val))  # Fe

# Plotter feilen
plt.figure(figsize=(8, 5))
plt.loglog(h_values, errors2, marker='o', linestyle='-', label="Num
plt.xlabel("h-verdi")
plt.ylabel("Feil i numerisk derivasjon")
plt.title("Feil i numerisk derivasjon av $e^x$ ved $x=1.5$")
plt.legend()
plt.grid(True, which="both", linestyle="--", linewidth=0.5)
plt.show()
```



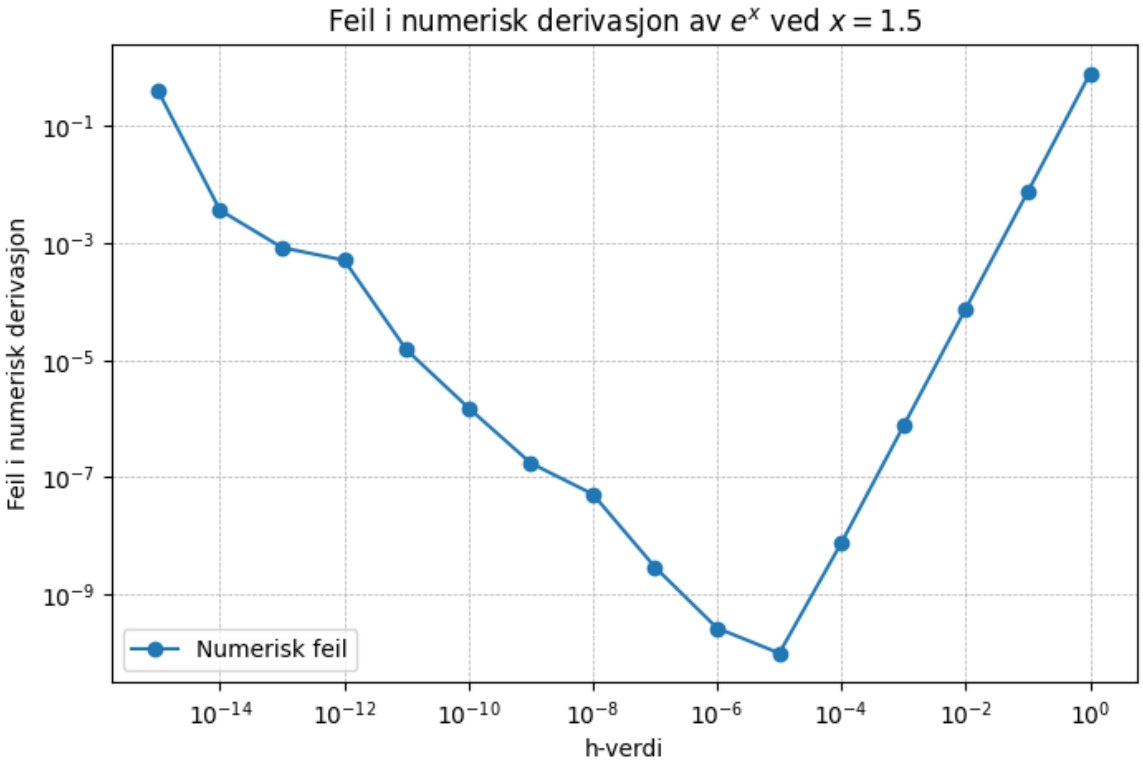
Kommentar til oppgave 2

I gjennomsnitt er funksjonen fra oppgave 2 ca. 3,6436 mer nøyaktig enn funksjonen fra oppgave 1

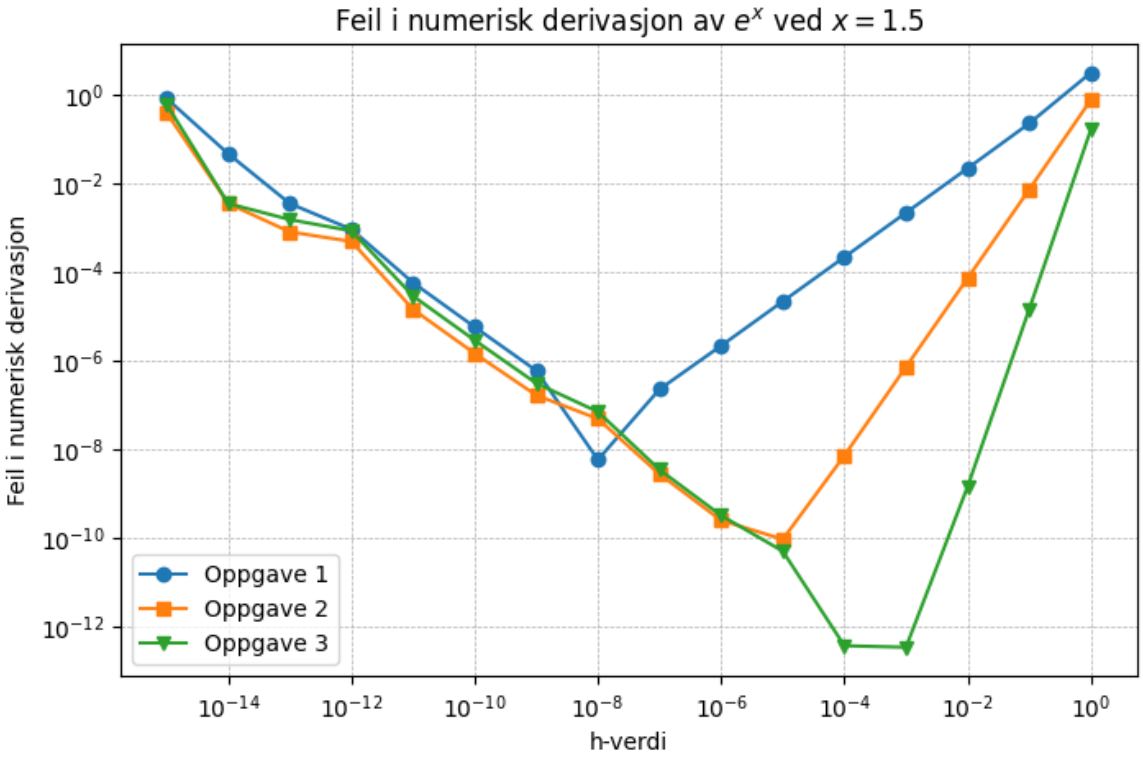
Oppgave 3

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

In [68]:
```
def df_numerisk_v3(x, h):
    return (f(x-2*h)-8*f(x-h)+8*f(x+h)-f(x+2*h))/(12*h)


errors3 = np.abs(df_numerisk_v3(x_val, h_values) - df(x_val))  # Fe

# Plotter feilen
plt.figure(figsize=(8, 5))
plt.loglog(h_values, errors2, marker='o', linestyle='-', label="Num
plt.xlabel("h-verdi")
plt.ylabel("Feil i numerisk derivasjon")
plt.title("Feil i numerisk derivasjon av $e^x$ ved $x=1.5$")
plt.legend()
plt.grid(True, which="both", linestyle="--", linewidth=0.5)
plt.show()
```



In [77]:
```
plt.figure(figsize=(8, 5))
plt.loglog(h_values, errors1, marker='o', linestyle='-', label="Opp
plt.loglog(h_values, errors2, marker='s', linestyle='-', label="Opp
plt.loglog(h_values, errors3, marker='v', linestyle='-', label="Opp

plt.xlabel("h-verdi")
plt.ylabel("Feil i numerisk derivasjon")
plt.title("Feil i numerisk derivasjon av $e^x$ ved $x=1.5$")
plt.legend()
plt.grid(True, which="both", linestyle="--", linewidth=0.5)
plt.show()

print(f"Gjennomsnittlig avvik (Oppgave 1): {np.average(errors1)}")
print(f"Gjennomsnittlig avvik (Oppgave 2): {np.average(errors2)}")
print(f"Gjennomsnittlig avvik (Oppgave 3): {np.average(errors3)}")
```




```
Gjennomsnittlig avvik (Oppgave 1): 0.27349135422081755
Gjennomsnittlig avvik (Oppgave 2): 0.07506243450903044
Gjennomsnittlig avvik (Oppgave 3): 0.049980623095041754
```

Kommentar til oppgave 3

Vi kan se at funksjonen i oppgave 3 har er nesten dobbelt så nøyaktig som funksjon fra oppgave 2, og vil derfor fungere best som en estimator

Oppgave 4, 5 og 6

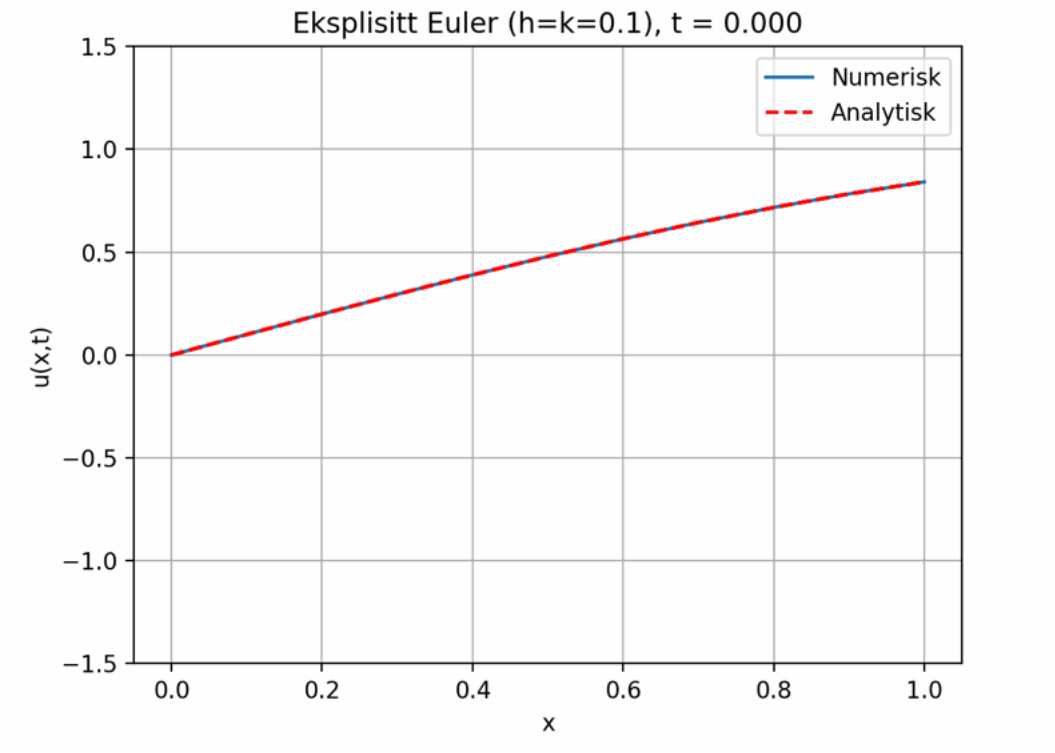
In [ ]:
```python
# funksjon for eksplisitt euler
def explicit_euler(N, M, h, k):
    x = np.linspace(0, 1, N+1)
    t = np.linspace(0, 1, M+1)
    u = np.zeros((M+1, N+1))
    u[0, :] = np.sin(x)
    r = k / h**2
    if r > 0.5:
        print(f"Ustabilitet ved r = {r:.2f} > 0.5")

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

# funksjon for implisitt euler
def implicit_euler(N, M, h, k):
    x = np.linspace(0, 1, N+1)
    t = np.linspace(0, 1, M+1)
```

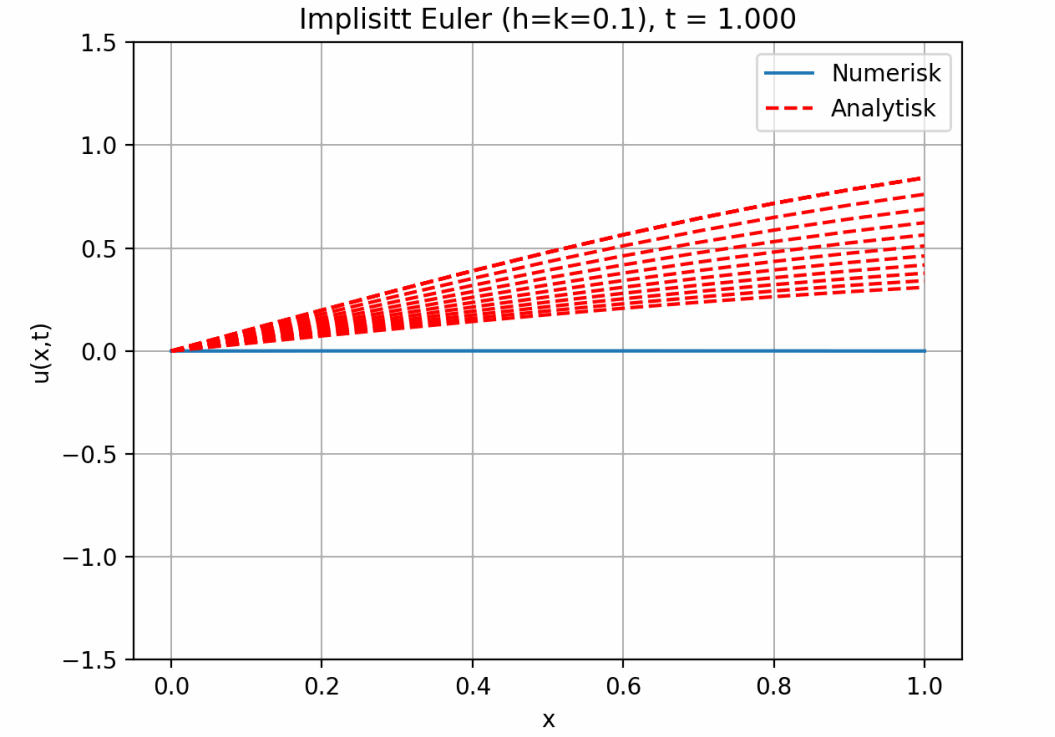
```
    u = np.zeros((M+1, N+1))
    u[0, :] = np.sin(x)
    r = k / h**2

    diagonals = [r * np.ones(N-2), -(1 + 2*r) * np.ones(N-1), r * n
    A = diags(diagonals, [-1, 0, 1]).tocsc()

    for j in range(M):
        b = u[j, 1:N]
        u[j+1, 1:N] = spsolve(A, b)
        u[j+1, 0] = 0
        u[j+1, N] = 0
    return x, t, u

# funksjon for crank-nicolson
def crank_nicolson(N, M, h, k):
    x = np.linspace(0, 1, N+1)
    t = np.linspace(0, 1, M+1)
    u = np.zeros((M+1, N+1))
    u[0, :] = np.sin(x)
    r = k / (2 * h**2)

    diag_A = (1 + 2*r) * np.ones(N-1)
    offdiag_A = -r * np.ones(N-2)
    A = diags([offdiag_A, diag_A, offdiag_A], [-1, 0, 1]).tocsc()

    diag_B = (1 - 2*r) * np.ones(N-1)
    offdiag_B = r * np.ones(N-2)
    B = diags([offdiag_B, diag_B, offdiag_B], [-1, 0, 1]).tocsc()

    for j in range(M):
        b = B.dot(u[j, 1:N])
        u[j+1, 1:N] = spsolve(A, b)
        u[j+1, 0] = 0
        u[j+1, N] = 0
    return x, t, u

# funksjon for å feilberegning
def compute_error(u, x, t):
    error = np.zeros(len(t))
    for j in range(len(t)):
        u_analytical = analytical_solution(x, t[j])
        error[j] = np.max(np.abs(u[j, :] - u_analytical))
    return error

test_cases = [
    {'h': 0.1, 'k': 0.1, 'label': 'h=k=0.1'},
    {'h': 0.1, 'k': 0.01, 'label': 'h=0.1, k=0.01'},
    {'h': 0.01, 'k': 0.1, 'label': 'h=0.01, k=0.1'},
]

for case in test_cases:
    h = case['h']
    k = case['k']
    N = int(1 / h)
    M = int(1 / k)
```

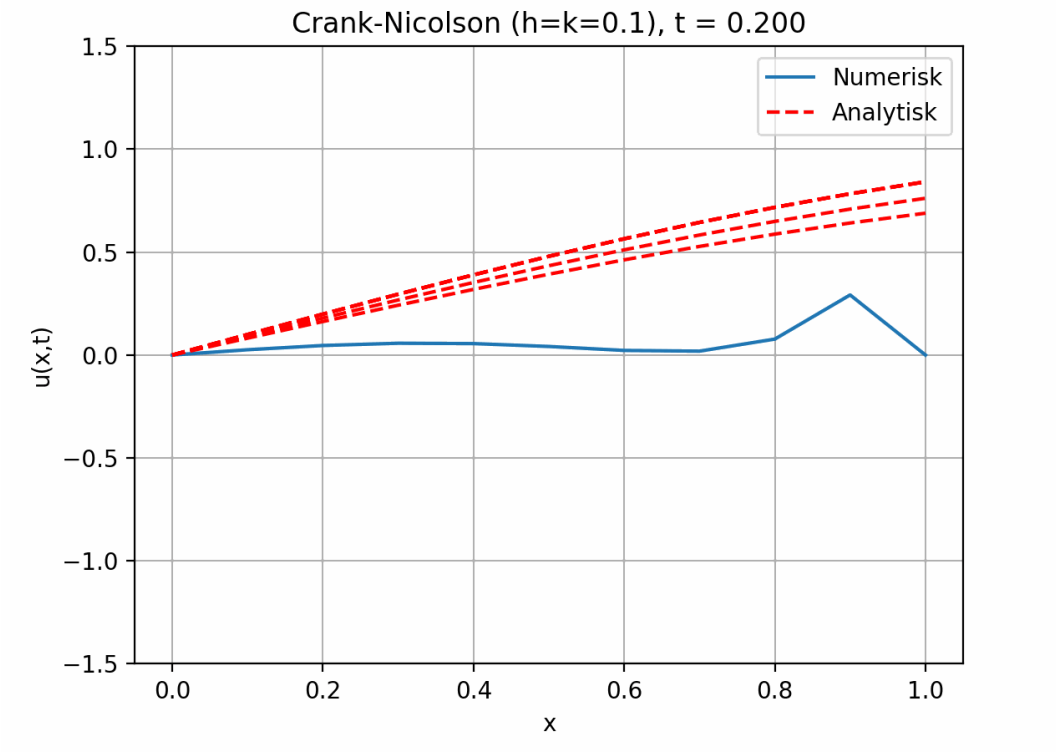
```
    label = case['label']

    print(f"Kjører testtilfelle: {label}")

    # eksplisitt euler
    x, t, u_explicit = explicit_euler(N, M, h, k)
    explicit_filename = os.path.join(SAVE_DIR, f"explicit_{label.re
    animate_solution(x, u_explicit, t, f"Eksplisitt Euler ({label})
    error_explicit = compute_error(u_explicit, x, t)

    # implisitt euler
    x, t, u_implicit = implicit_euler(N, M, h, k)
    implicit_filename = os.path.join(SAVE_DIR, f"implicit_{label.re
    animate_solution(x, u_implicit, t, f"Implisitt Euler ({label})"
    error_implicit = compute_error(u_implicit, x, t)

    # crank-nicolson
    x, t, u_cn = crank_nicolson(N, M, h, k)
    cn_filename = os.path.join(SAVE_DIR, f"cn_{label.replace('=', '
    animate_solution(x, u_cn, t, f"Crank-Nicolson ({label})", cn_fi
    error_cn = compute_error(u_cn, x, t)

    plt.figure()
    plt.plot(t, error_explicit, label='Eksplisitt Euler')
    plt.plot(t, error_implicit, label='Implisitt Euler')
    plt.plot(t, error_cn, label='Crank-Nicolson')
    plt.xlabel('t')
    plt.ylabel('Maksimal absolutt feil')
    plt.title(f'Feil for {label}')
    plt.legend()
    plt.grid(True)

    error_filename = os.path.join(SAVE_DIR, f"error_{label.replace(
    plt.savefig(error_filename)
    print(f"Lagret feilplott som {error_filename}")
    plt.close()
```

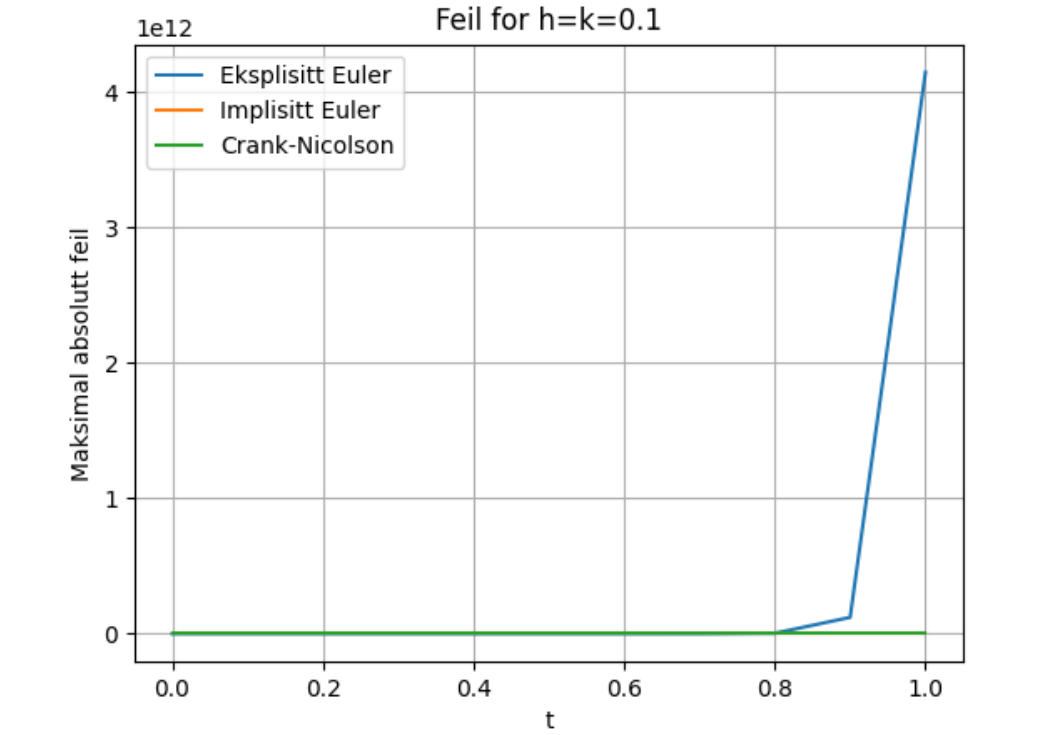
Animasjoner og feilplott

Testtilfelle: h=k=0.1

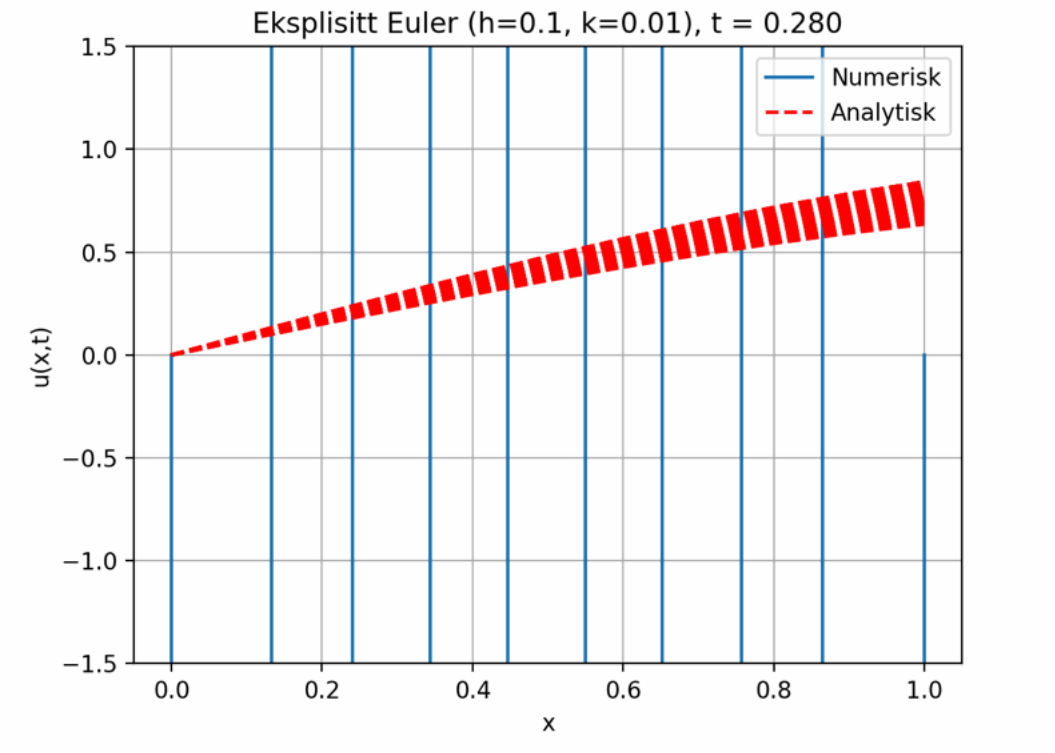

Eksplisitt Euler (h=k=0.1), t = 0.000
Numerisk
Analytisk
u(x,t)
x
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0.0
0.2
0.4
0.6
0.8
1.0

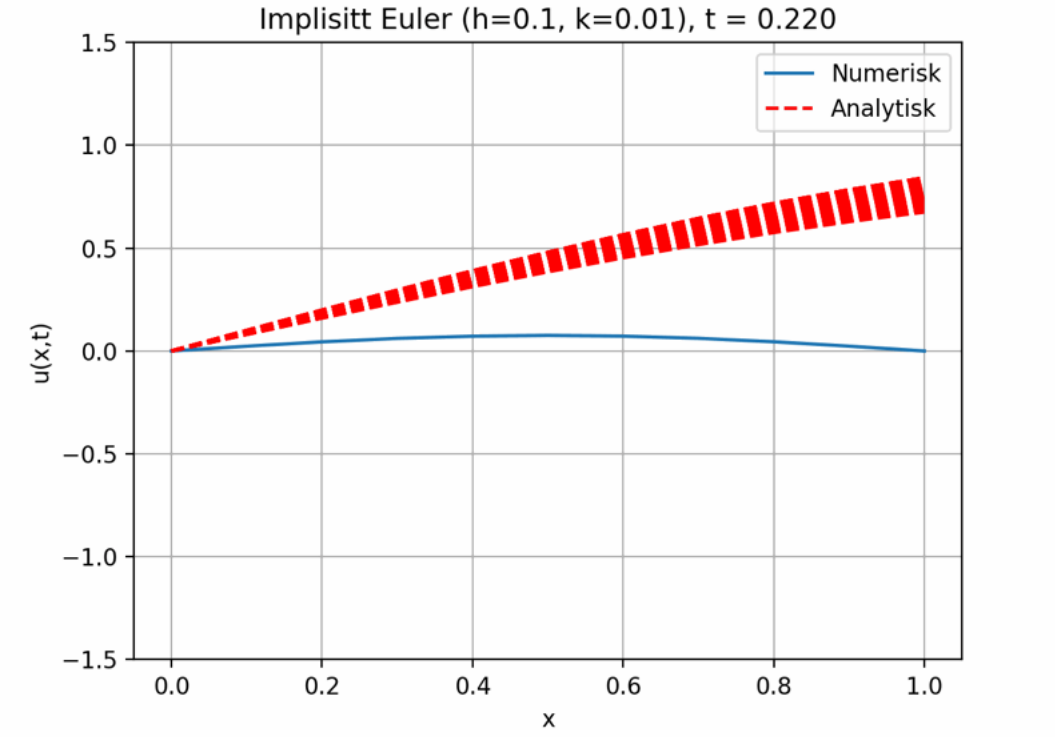

Implisitt Euler (h=k=0.1), t = 1.000
Numerisk
Analytisk
u(x,t)
x
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0.0
0.2
0.4
0.6
0.8
1.0

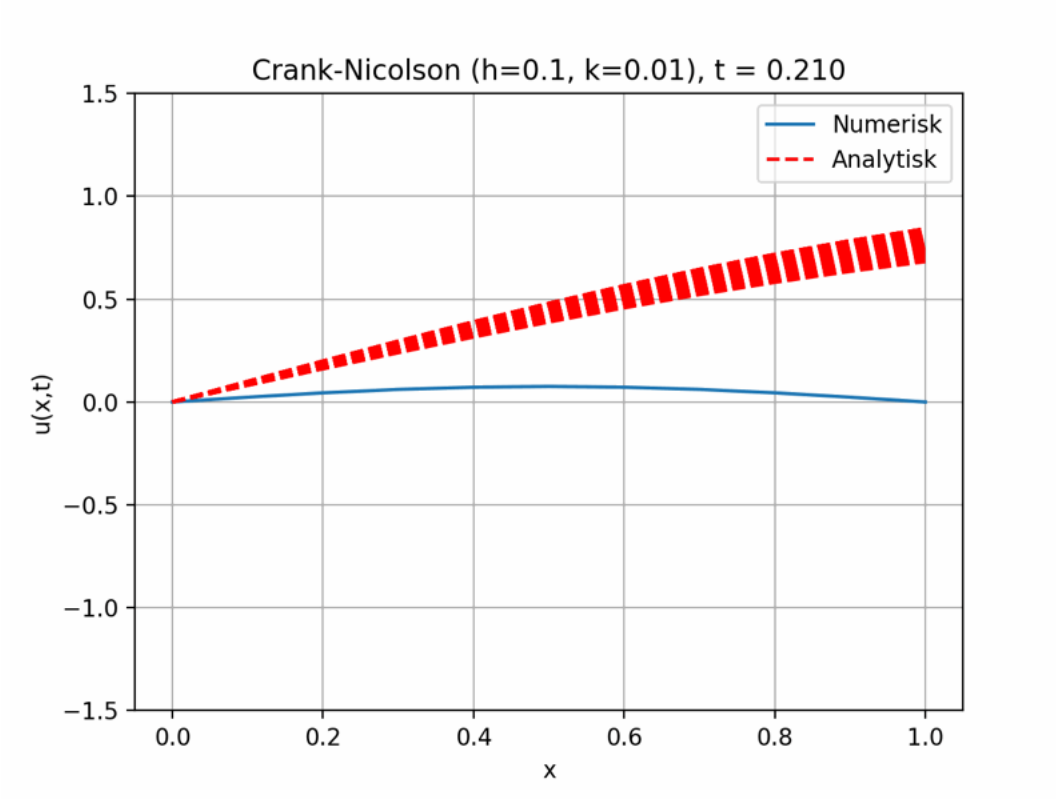


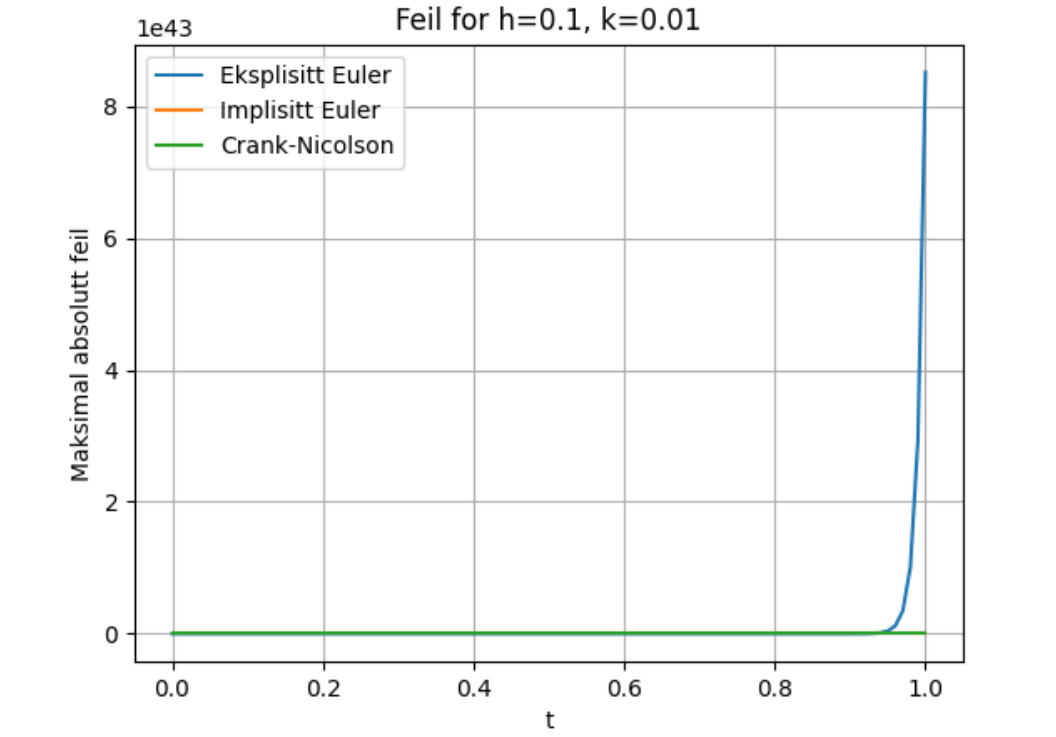


Testtilfelle: h=0.1, k=0.01

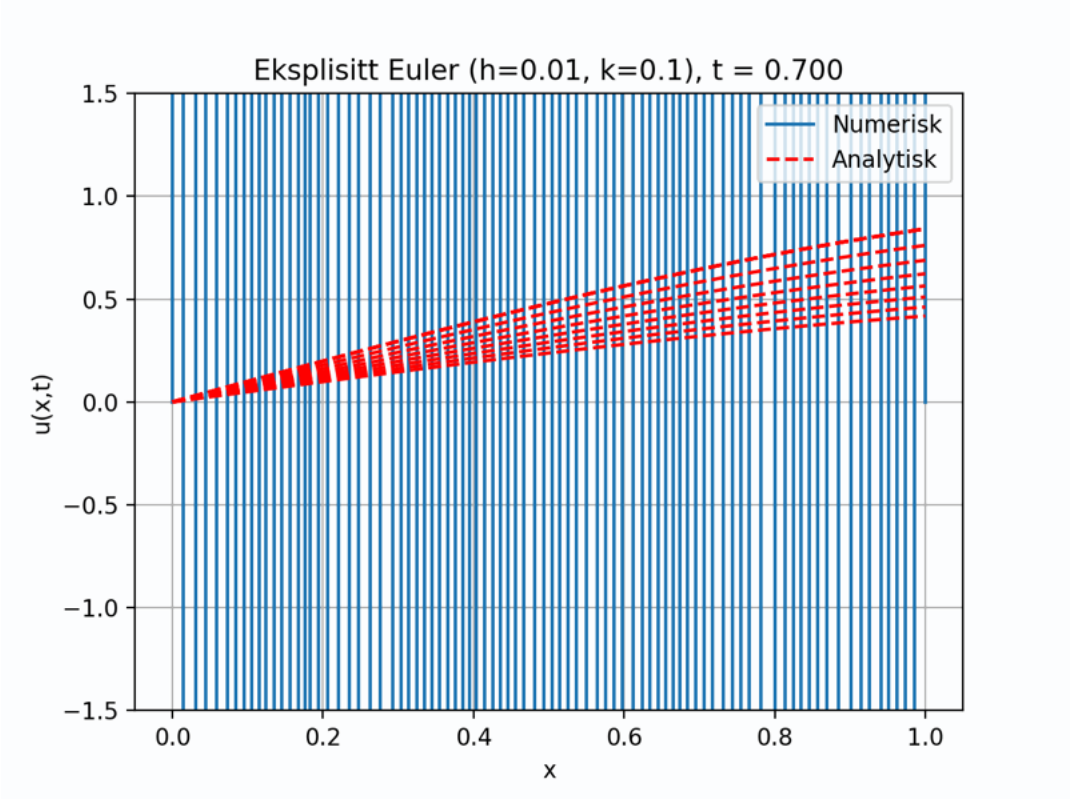

Eksplisitt Euler (h=0.1, k=0.01), t = 0.280
Numerisk
Analytisk
u(x,t)
x
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0.0
0.2
0.4
0.6
0.8
1.0

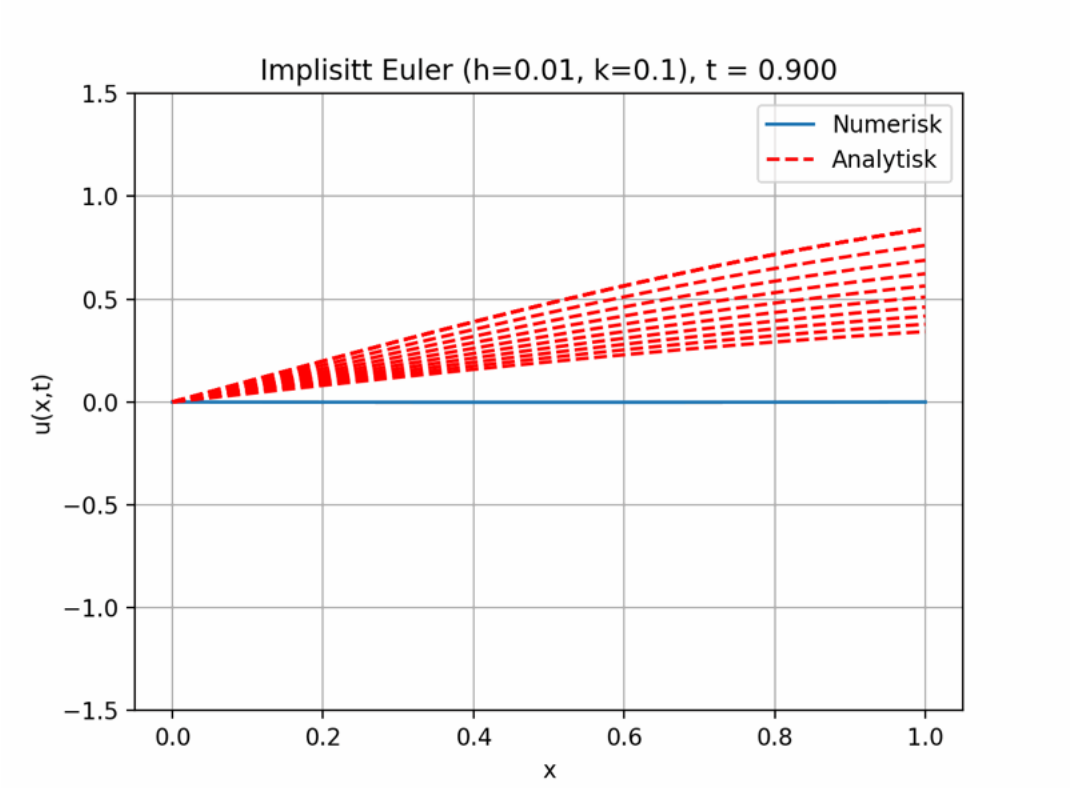

Implisitt Euler (h=0.1, k=0.01), t = 0.220
Numerisk
Analytisk
u(x,t)
x
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0.0
0.2
0.4
0.6
0.8
1.0

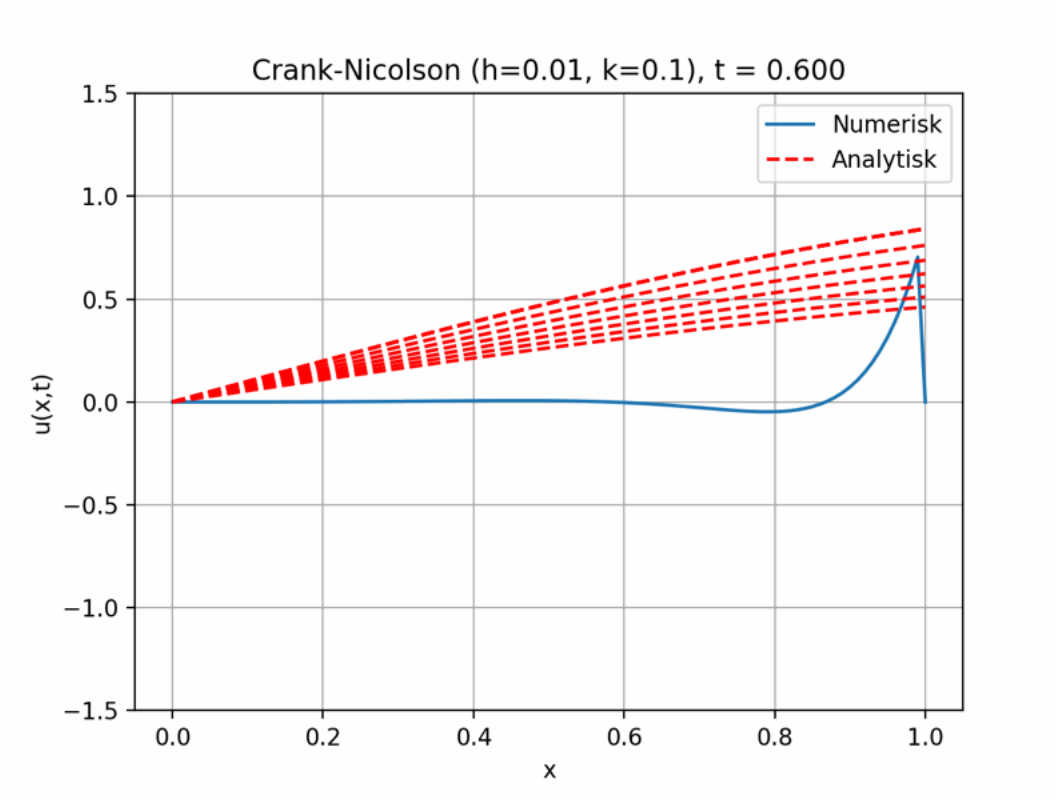


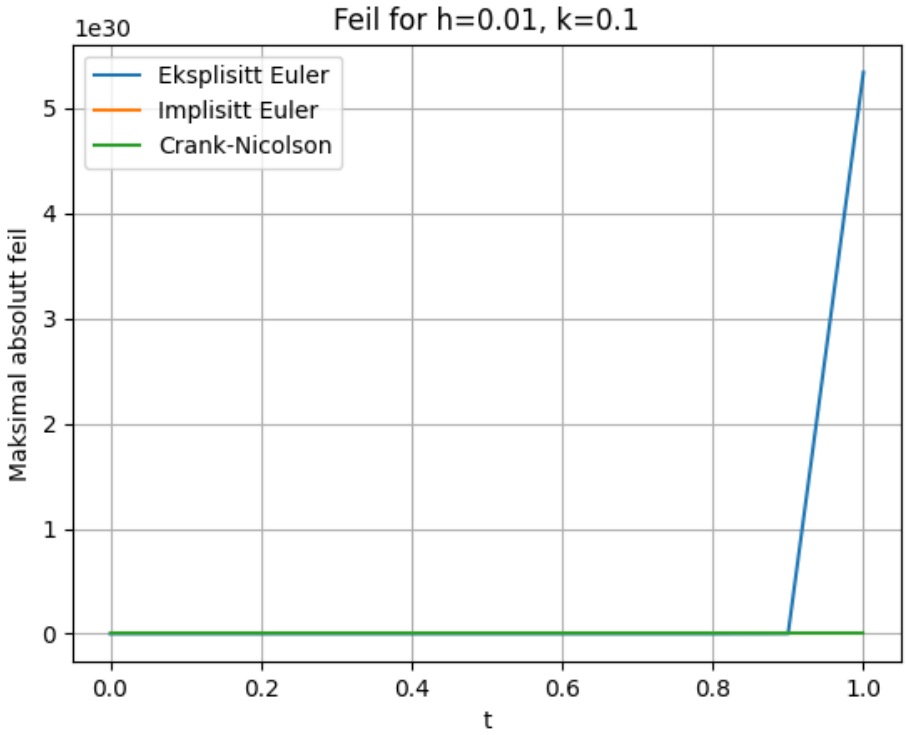


Testtilfelle: h=0.01, k=0.1


Eksplisitt Euler (h=0.01, k=0.1), t = 0.700
Numerisk
Analytisk
u(x,t)
x
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0.0
0.2
0.4
0.6
0.8
1.0


Implisitt Euler (h=0.01, k=0.1), t = 0.900
Numerisk
Analytisk
u(x,t)
x
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0.0
0.2
0.4
0.6
0.8
1.0





Kommentar til oppgave 4, 5 og 6

Koden ser noe rar ut fordi jeg utela noe av koden der jeg lagret bildene.

Jeg implementerte de tre numeriske metodene, eksplisitt Euler, implisitt Euler og Crank-Nicolson. Jeg testet metodene med paramenterkombinasjonene $h = k = 0.1$, $h = 0.1, k = 0.01$ og $h = 0.01, k = 0.1$, og sammenlignet resultatene med den analytiske løsningen $u(x,t) = e^{-t} * sin(x)$.

Man kunne se store oscillasjoner og feilplott som indikerer avvik for eksplisitt Euler. Dette gjelder alle parameterkombinasjonene. Implisitt Euler og Crank-Nicolson var relativt stabile i alle tilfeller, men implisitt Euler hadde større feil enn Crank-Nicolson, spesielt ved større k-verdier. Animasjonene visualiserte tydelig hvordan løsningene utviklet seg over tid, og feilplottene kvantifiserte avvikene. Crank-Nicolson holdte seg mest korrekt, mens eksplisitt Euler kan være noe gunstig for mindre k-verdier.

Bibliografi

- https://mortano.folk.ntnu.no/oblig/oblig-4106.pdf