OBLIG. TMA4106

1. $f'(1{,}5) \approx \dfrac{f(1{,}5+h) - f(1{,}5)}{h}$

Den eksakte deriverte til $f(x) = e^x$ i punktet $x = 1{,}5$ er $f'(1{,}5) = e^{1{,}5} \approx 4{,}4817$

Numerisk kan vi tilnærme den ved

$$\frac{f(1{,}5+h) - f(1{,}5)}{h} = \frac{e^{1{,}5+h} - e^{1{,}5}}{h} = D_h$$

- $h = 0{,}1 \Rightarrow D_{0{,}1} = \dfrac{e^{1{,}6} - e^{1{,}5}}{0{,}1} = 4{,}7134$

↳ Tilnærmingen bommer med ca. 0,23

- $h = 0{,}01 \Rightarrow D_{0{,}01} = \dfrac{e^{1{,}51} - e^{1{,}5}}{0{,}01} \approx 4{,}5011$

↳ Tilnærmingen bommer med ca. 0,0194

- $h = 0{,}001 \Rightarrow D_{0{,}001} = \dfrac{e^{1{,}501} - e^{1{,}5}}{0{,}001} \approx 4{,}4859$

↳ Tilnæmingen bommer med 0,00424

- $h = 0{,}0001 \Rightarrow D_{0{,}0001} = \dfrac{e^{1{,}5001} - e^{1{,}5}}{0{,}0001} = 4{,}4819$

↳ Tilnærmingen bommer med $2{,}24 \cdot 10^{-4}$

Hvis vi fortsetter på samme måte, vil vi se at feilen initialt går nedover når h reduseres, fordi vi da får en bedre "sekant-tilnærming" til tangenten. Men fra et visst punkt vil flyttalls-avrunding gjøre at differansen $e^{1{,}5+h} - e^{1{,}5}$ blir så liten at den "drukner" i maskinpresisjonen. Da begynner feilen å vokse igjen.

For rundt $h = 10^{-13}$ begynner feilen å øke igjen. Her vil ikke den numeriske deriverte lenger nærme seg den sanne verdien, og resultatet blir upålitelig.

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

- For $h = 0{,}1$: $f'(1{,}5) = \frac{e^{1,6} - e^{1,4}}{2 \cdot 0{,}1} = 4{,}4892$

↳ Tilnærmingen bommer med $7{,}46 \cdot 10^{-3}$

- For $h = 0{,}01$: $f'(1{,}5) = \frac{e^{1,51} - e^{1,49}}{2 \cdot 0{,}01} = 4{,}4818$

↳ Tilnærmingen bommer med $7{,}47 \cdot 10^{-5}$

- For $h = 0{,}001$: $f'(1{,}5) = \frac{e^{1,501} - e^{1,499}}{2 \cdot 0{,}001} = 4{,}48169$

↳ Tilnærmingen bommer med $7{,}47 \cdot 10^{-7}$

For $h = 10^{-9}$ begynner feilen å øke igjen. Her blir de numeriske beregningene upålitelige.

Taylor-analysen sier at feilen i sentraldifferansen $\sim h^2$. Når $h \to 0$, blir denne delen av feilen veldig liten.

For veldig små h blir differansen $f(x+h) - f(x-h)$ svært liten. Når vi da deler på $2h$, kan feilen blåses opp.

Taylorutviklingen av $f(x+h)$ og $f(x-h)$ rund x:

$$f(x+h) = f(x) + f'(x)h + \frac{f''(x)}{2}h^2 + \frac{f'''(x)}{6}h^3 + \dots$$

$$f(x-h) = f(x) - f'(x)h + \frac{f''(x)}{2}h^2 - \frac{f'''(x)}{6}h^3 + \dots$$

$$f(x+h) - f(x-h) = 2f'(x)h + \frac{2f'''(x)}{6}h^3 + \dots$$

$\Rightarrow$ deler på $2h$:

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

↳ Feilen er nå proporsjonal med h^2 og ikke h. Noe som betyr at når vi deler h på 10, vil feilen omtrent bli 100 ganger mindre (feilen oppfører seg som h^2)

4 For å løse varmelikningen må vi diskretisere i rommet og i tiden:

Vi diskretiserer x-aksen med n+1 punkter

$\Rightarrow x_i = ih$, $i = 0, 1, \ldots, n$ $\Rightarrow h = \frac{1}{n}$

Tiden diskretiserer vi i steg Δt, og lar $m = 0, 1, \ldots, M$ være tidsindeks. Løsninge, lagres som et array $U[i, m]$ som tilnærmer $u(x_i, t_m)$.

Varmeligningen i 1D (med diffusjonskoeffisient $\alpha = 1$ for enkelthetsskyld) er: $u_t = u_{xx}$

$x \in [0,1]$ og $t > 0$ med randbetingelser $u(0,t) = 0$, $u(1,t) = 0$ og startbetingelser $u(t,0) = f(x)$

Den romlige andregeleen $\frac{\partial^2 u}{\partial x^2}$ approksimeres

av: $u_{xx}(x_i, t_m) = \dfrac{U[i+1, m] - 2U[i, m] + U[i-1, m]}{h^2}$

La $\lambda = \dfrac{\Delta t}{h^2}$

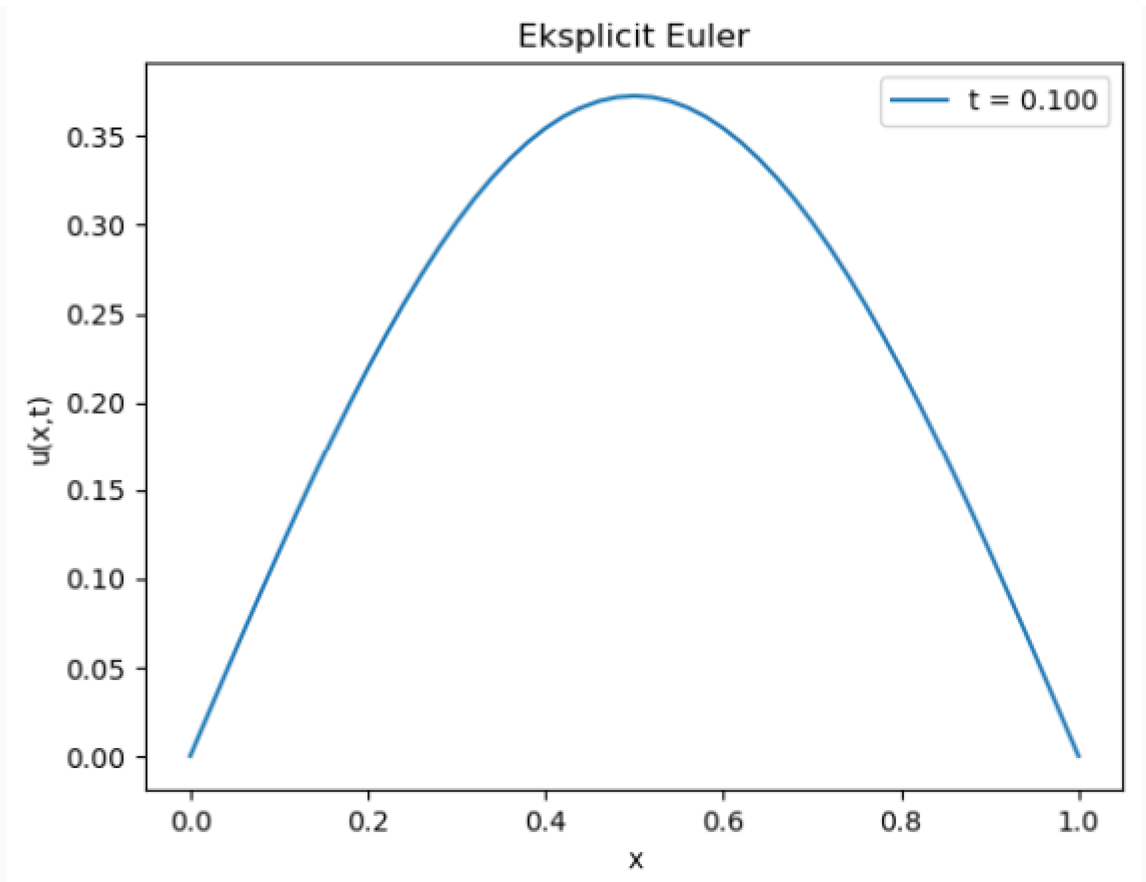
Eksplisitt : $\dfrac{u_{i,j+1} - u_{ij}}{k} = \dfrac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}$

Den eksplisitte eulermetoden i tid gir per steg:

$U[i,m+1] = U[i,m] + \lambda(U[i+1,m] - 2U[i,m] + U[i-1,m])$,

for $i = 1, 2, \dots, n-1$. Randpunktene $i=0$ og $i=n$ settes til 0 i hvert steg (gitt randbetingelsene).

Resultater fra python-script :



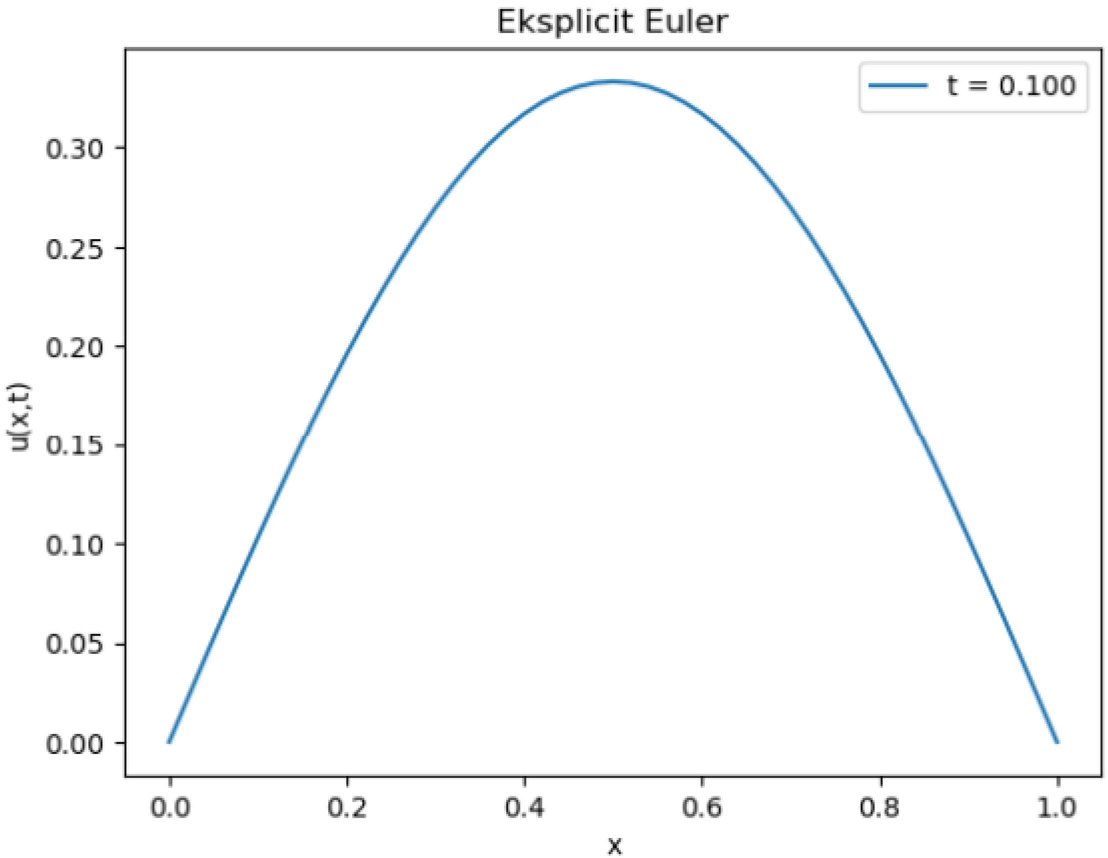
$n = 50$

$dt = 0,0001$

$T = 0,1$

$h = \frac{1}{n} = 0,02$

$\Rightarrow$ stabil og jevn kurve

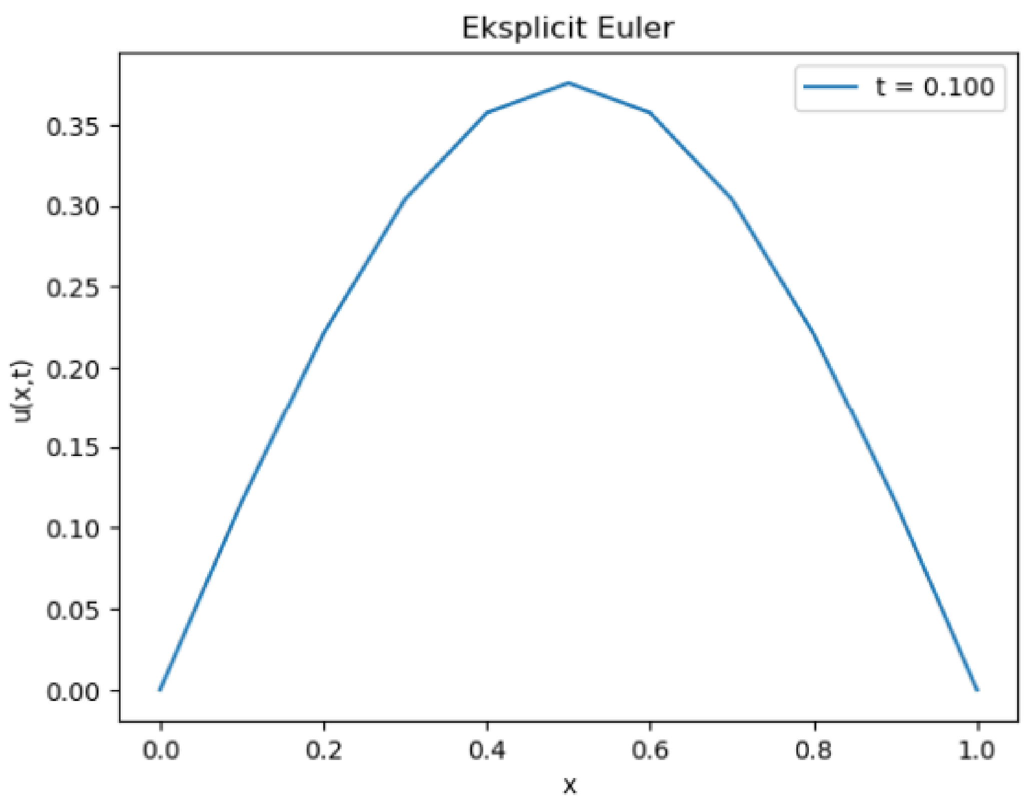


n = 50

dt = 0,02

T = 0,1

h = dt = 0,02 => toppunkt lengre ned på y-aksen.

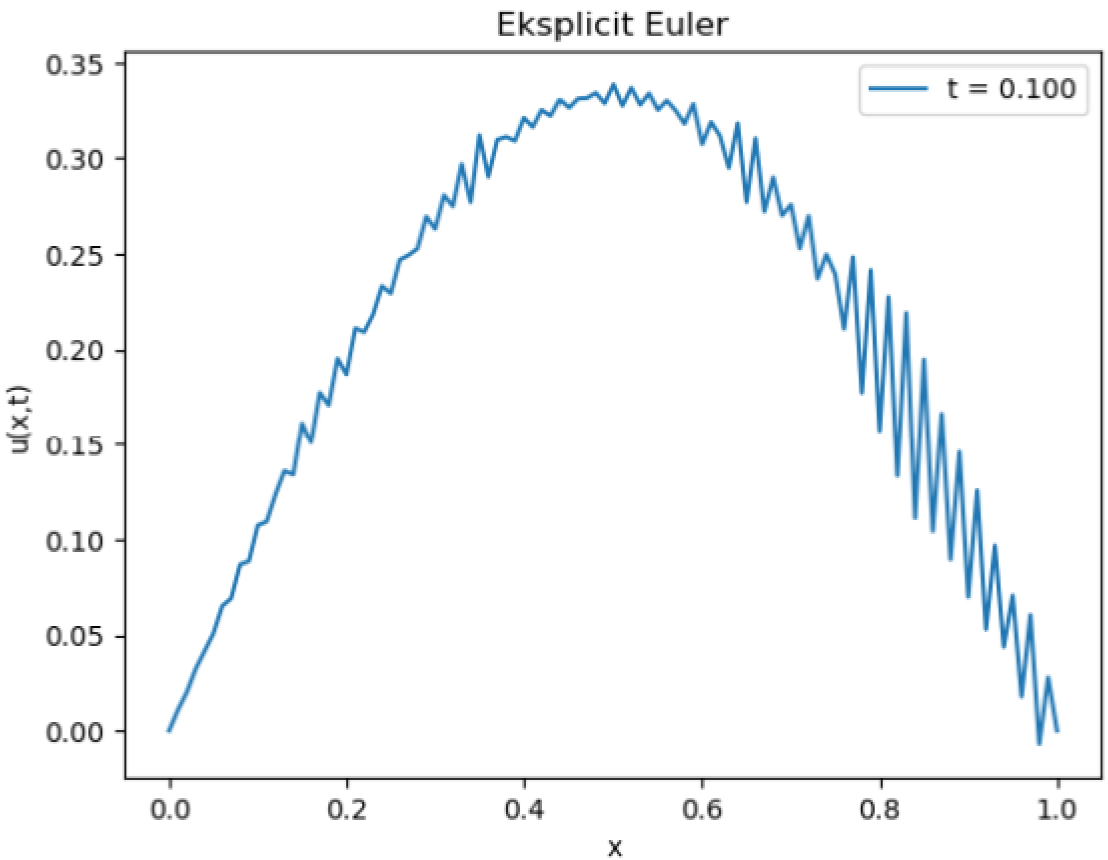


n = 10

dt = 0,00001

T = 0,1

h = 0,1 og k = 0,00001 h >> k => ujevn graf



$n = 100$

$dt = 0{,}02$

$T = 0{,}1$

$h = 0{,}01$ og $k = 0{,}02$

$k > h \Rightarrow$ hakkete graf, med større gap i hakkene for økende x.

5 Implisitt : $\frac{u_{i,j+1} - u_{ij}}{k} = \frac{u_{i+1,j+1} - 2u_{i,j+1} + u_{i-1,j+1}}{h^2}$

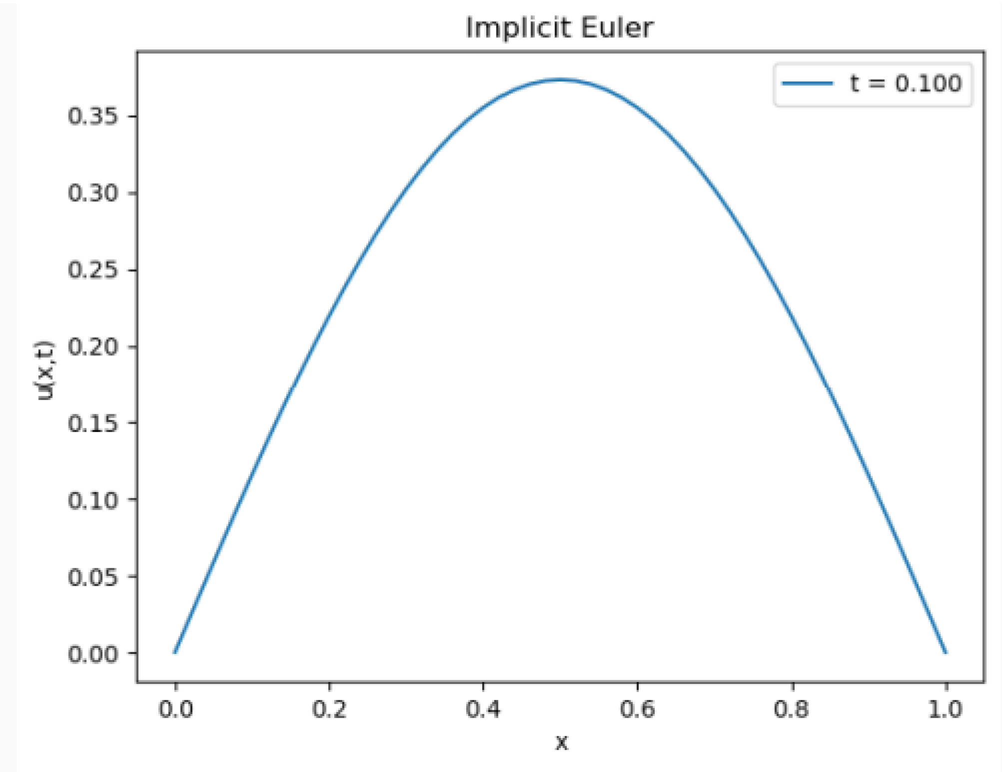
↳ Implisitt euler har vi tidssteget :

$U[i,m+1] - \lambda(U[i+1,m+1] - 2U[i,m+1] + U[i-1,m+1]) = U[i,m]$

På matriseform kan dette skrives :

$(I + \lambda A)U^{m+1} = U^m$, hvor A er den vanlige tridiagonale "andregradsderivasjons" - matrisen for indre punkter. Vi må dermed løse et lineært system for hver tidsiterasjon.

Resultat :



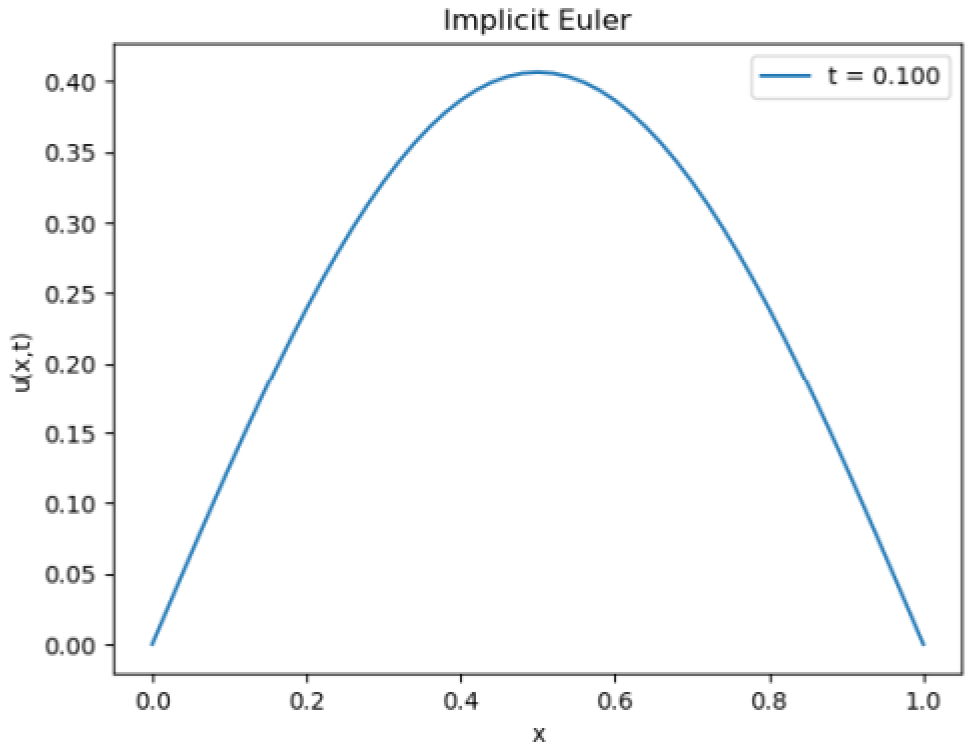
$n = 50$

$dt = 0{,}0001$

$T = 0{,}1$

$h = \frac{1}{n} = 0{,}02$ $dt = k = 0{,}0001$

stabil, jevn kurve

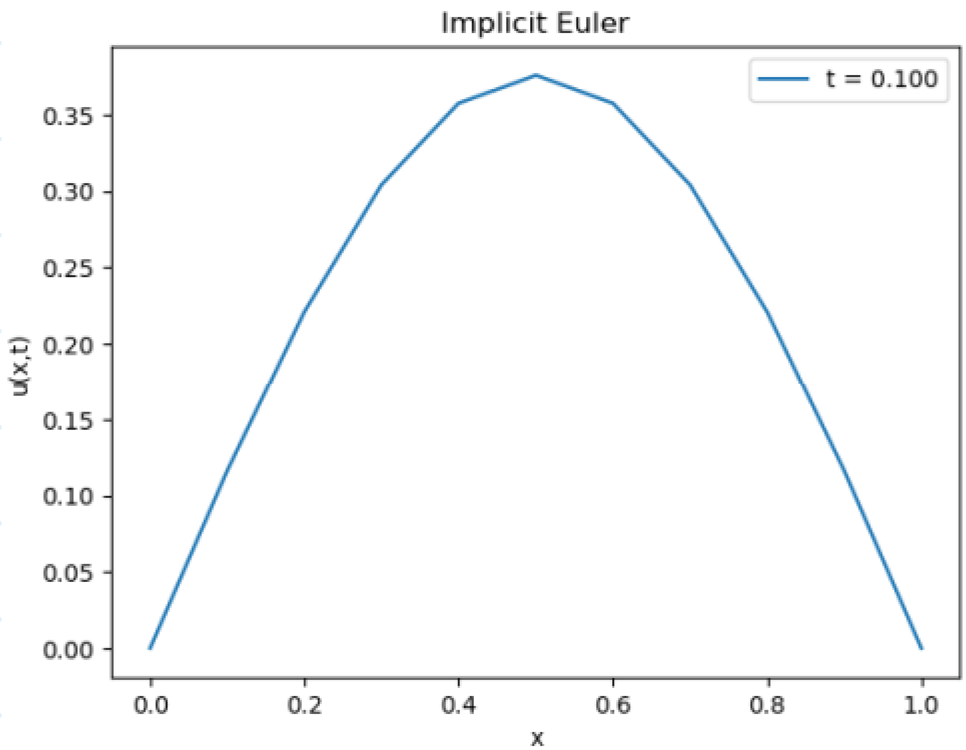


$n = 50$

$dt = 0{,}02$

$T = 0{,}1$

$h = k = 0{,}02 \Rightarrow$ høyere topp, motsatt for eksplisitt



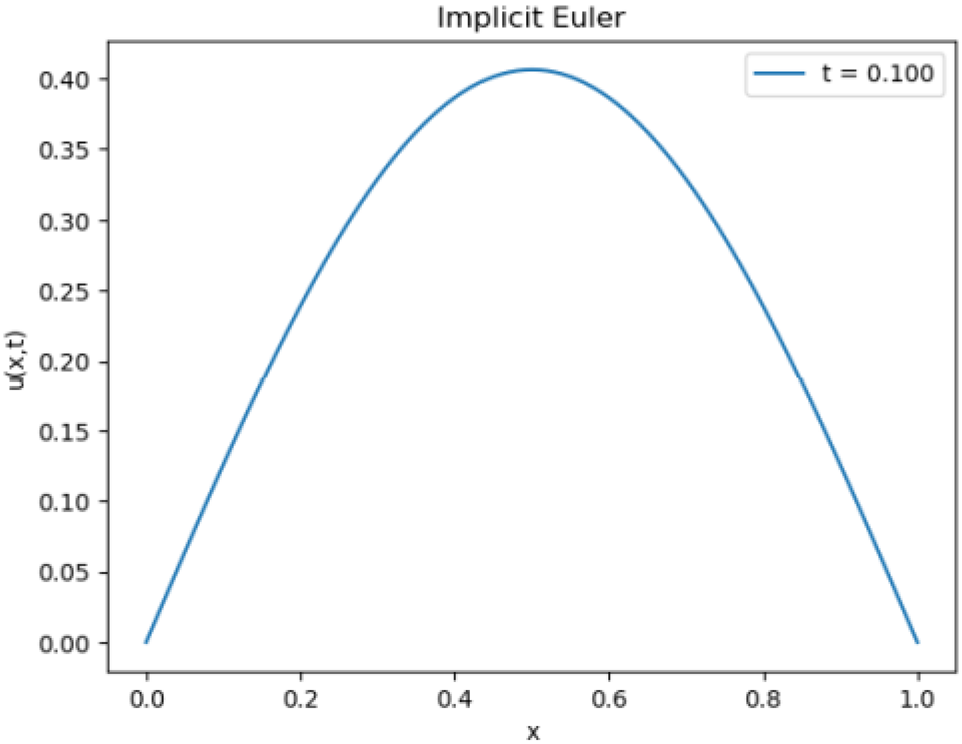
$n = 10$

$dt = 0{,}00001$

$T = 0{,}1$

$h = 0{,}1$, $k = 0{,}00001$, $h >> k$

$\Rightarrow$ mer hakkete kurve $\Rightarrow$ lik eksplisitt



$n = 100$

$dt = 0{,}02$

$T = 0{,}1$

$h = 0{,}01 \quad k = 0{,}02$

$k > h \Rightarrow$ jevn og glatt kurve

6

Crank-Nicolson: $\frac{u_{i,j+1} - u_{ij}}{k}$

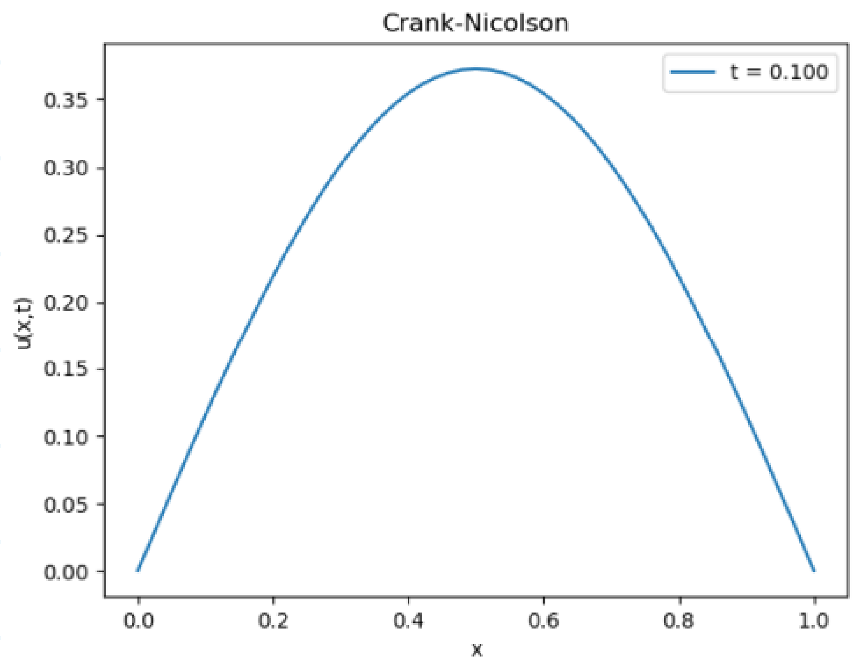
$$= \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{2h^2} + \frac{u_{i+1,j+1} - 2u_{i,j+1} + u_{i-1,j+1}}{2h^2}$$

↳ "halv eksplisitt og halv implisitt"

$$U[i,m+1] - \frac{\lambda}{2}\left(U[i+1,m+1] - 2U[i,m+1] + U[i-1,m+1]\right)$$
$$= U[i,m] + \frac{\lambda}{2}\left(U[i+1,m] - 2U[i,m] + U[i-1,m]\right)$$

I matriseform: $\left(I + \frac{\lambda}{2}A\right)U^{m+1}$
$$= \left(I - \frac{\lambda}{2}A\right)U^m$$

Resultat:



$n = 50$

$dt = 0{,}0001$

$T = 0{,}1$

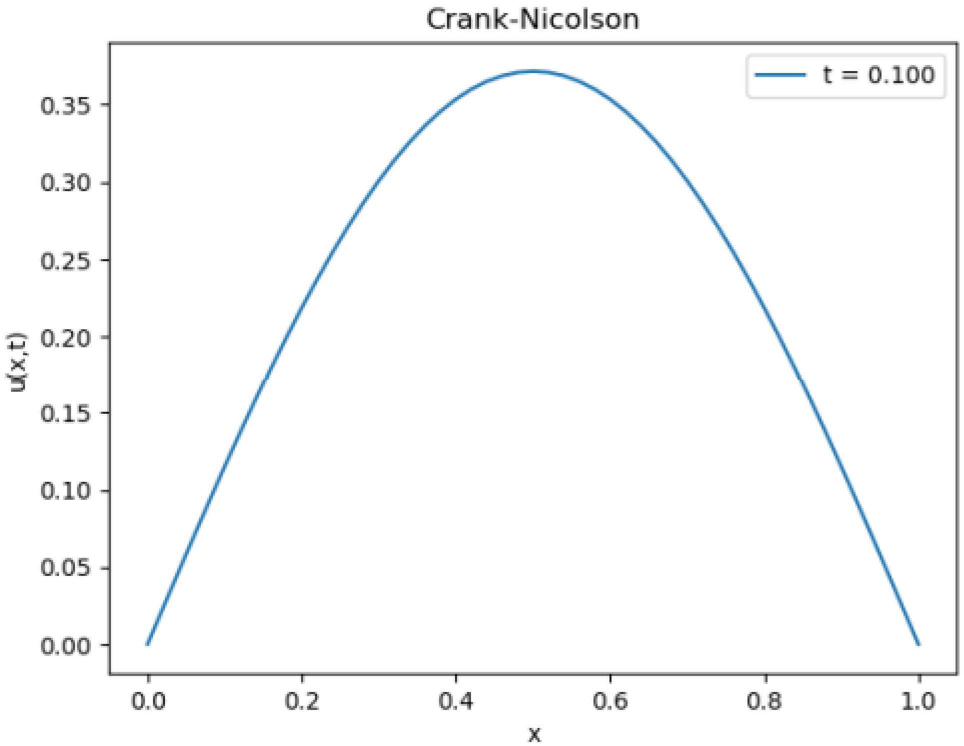
$h = \frac{1}{n} = 0{,}02$

$dt = k = 0{,}0001$

$h > k$

Jevn og glatt kurve

=> lik både eksplisitt og implisitt



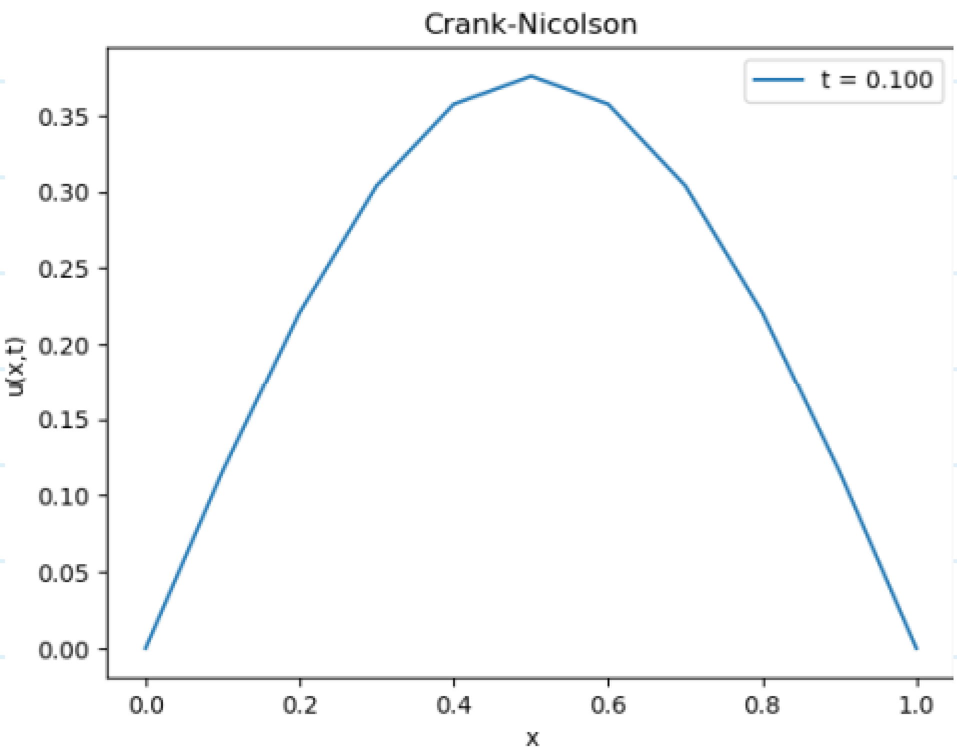
n = 50

dt = 0,02

T = 0,1

h = k = 0,02

Lik som over => forskjellen fra eksplisitt og implisitt er at toppunktet verken synker eller øker



n = 10

dt = 0,00001

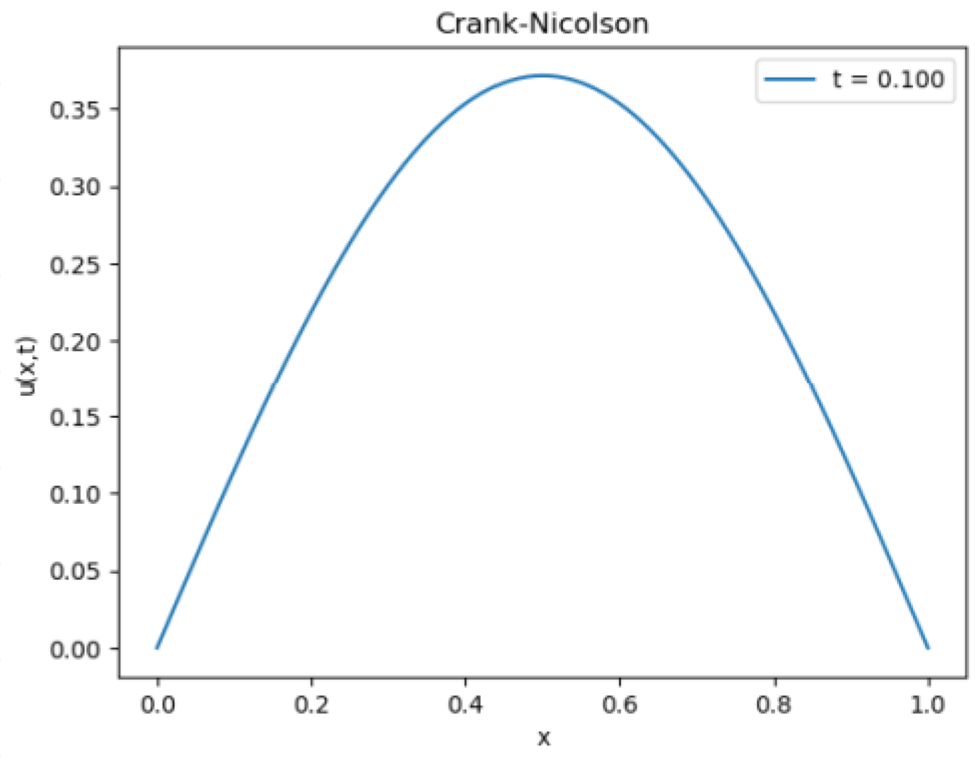
T = 0,1

h = 0,1

k = 0,00001

h >> k

Hakkete kurve => ligner eksplisitt og implisitt



$n = 100$

$dt = 0,02$

$T = 0,1$

$h = 0,01$

$k = 0,02$

Glatt og jevn kurve $\Rightarrow$ ligner implisitt

Konklusjon:

- Crank-Nicolson og implisitt: Disse metodene er mer stabile og tåler større tidssteg uten å bli helt hakkete. Mer praktisk i situasjoner der man ikke kan ta veldig små tidssteg
- Eksplisitt: Metoden er enklere å programmere men stabilitetsgrensen er "strengere".

```
#Oppgave 4

import numpy as np
import matplotlib.pyplot as plt

def f_initial(x):
    return np.sin(np.pi * x)

def heat_explicit_euler(n, dt, T):
    x = np.linspace(0, 1, n+1)
    h = 1.0 / n
    M = int(np.round(T/dt))
    t = np.linspace(0, M*dt, M+1)

    lam = dt / (h*h)
    U = np.zeros((n+1, M+1))

    for i in range(n+1):
        U[i,0] = f_initial(x[i])

    for m in range(M):
        for i in range(1, n):
            U[i, m+1] = U[i,m] + lam * (U[i+1, m] - 2*U[i,m] + U[i-1,m])

        U[0, m+1] = 0.0
        U[n, m+1] = 0.0
    return U, x, t

if __name__ == "__main__":
    n = 100
    dt = 0.02
    T = 0.1

    U_explicit, x, t = heat_explicit_euler(n, dt, T)

    plt.plot(x, U_explicit[:,-1], label = "t = %.3f" %t[-1])
    plt.xlabel("x")
    plt.ylabel("u(x,t)")
    plt.title("Eksplicit Euler")
    plt.legend()
    plt.show()
```

```python
#Oppgave 5
import numpy as np
import matplotlib.pyplot as plt

def f_initial(x):
    return np.sin(np.pi * x)

def heat_explicit_euler(n, dt, T):
    x = np.linspace(0, 1, n+1)
    h = 1.0 / n
    M = int(np.round(T/dt))
    t = np.linspace(0, M*dt, M+1)

    lam = dt / (h*h)
    U = np.zeros((n+1, M+1))

    for i in range(n+1):
        U[i,0] = f_initial(x[i])

        diag = np.ones(n-1)*(1 + 2*lam)
        off = np.ones(n-2)*(-lam)

    import scipy.linalg as la

    for m in range(M):
        b = U[1:n, m].copy()

        U_inner = la.solve_banded((1,1),
                                  np.vstack([np.hstack(([0],off)),
                                             diag,
                                             np.hstack((off, [0]))]),
                                  b)
        U[1:n, m+1] = U_inner


        U[0, m+1] = 0.0
        U[n, m+1] = 0.0
    return U, x, t

if __name__ == "__main__":
    n = 100
    dt = 0.02
    T = 0.1

    U_implicit, x, t = heat_explicit_euler(n, dt, T)

    plt.plot(x, U_implicit[:,-1], label = "t = %.3f" %t[-1])
    plt.xlabel("x")
    plt.ylabel("u(x,t)")
    plt.title("Implicit Euler")
    plt.legend()
    plt.show()
```

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.linalg as la

def f_initial(x):
    return np.sin(np.pi * x)

def heat_crank_nicolson(n, dt, T):
    x = np.linspace(0, 1, n+1)
    h = 1.0 / n
    M = int(np.round(T/dt))
    t = np.linspace(0, M*dt, M+1)

    lam = dt / (h*h)
    U = np.zeros((n+1, M+1))

    for i in range(n+1):
        U[i,0] = f_initial(x[i])

    main_diag_plus = np.ones(n-1)*(1 + lam)
    off_diag_plus = np.ones(n-2)*(-lam/2)

    main_diag_minus = np.ones(n-1)*(1 - lam)
    off_diag_minus = np.ones(n-2)*( lam/2)

    for m in range(M):
        b = np.zeros(n-1)

        for i_ind in range(n-1):
            i_global = i_ind + 1
            b[i_ind] = main_diag_minus[i_ind]*U[i_global,m]
            if i_ind+1 < (n-1):
                b[i_ind] += off_diag_minus[i_ind]*U[i_global +1, m]
            if i_ind-1 >= 0:
                b[i_ind]+=off_diag_minus[i_ind-1]*U[i_global-1,m]
        A_banded_plus = np.vstack([
            np.hstack(([0],off_diag_plus)),
            main_diag_plus,
            np.hstack((off_diag_plus, [0]))
            ])

        U_inner = la.solve_banded((1,1),A_banded_plus,b)

        U[1:n, m+1] = U_inner


        U[0, m+1] = 0.0
        U[n, m+1] = 0.0
    return U, x, t

if __name__ == "__main__":
    n = 100
    dt = 0.02
    T = 0.1

    U_cn, x, t = heat_crank_nicolson(n, dt, T)

    plt.plot(x, U_cn[:,-1], label = "t = %.3f" %t[-1])
    plt.xlabel("x")
    plt.ylabel("u(x,t)")
    plt.title("Crank-Nicolson")
    plt.legend()
    plt.show()
```