# Numerisk løysing av varmelikninga i to dimensjonar

## 1 Introduksjon

Det er ofte nyttig å vite korleis varme sprer seg gjennom eit plan når det leggast varme legeme langs eller på planet. Då blir ein nødd til å løyse varmelikninga numerisk i to romslege dimensjonar.

## 2 Teori

Varmelikninga er gitt ved

$$u_t(x,t) = \alpha u_{xx}(x,t) \tag{2.1}$$

eit uttrykk som kan omformulerast til å omfatte to romslege koordinater og ein tidskoordinat, altså $x, y$ og $t$. Her, og ovenfor, er $\alpha$ ein konstant.

$$u_t(x,y,t) = \alpha(u_{xx}(x,y,t) + u_{yy}(x,y,t)) \tag{2.2}$$

For denne simuleringa er det derimot ønskjeleg å diskretisere likninga, slik den kan behandlast som eit felt med gitterpunkt. Vi veljer då at avstanden mellom eit kvart gitterpunkt er representert med $\Delta x, \Delta y$ og $\Delta t$, og at total avstand frå origo langs ei akse då blir $x_i = i\Delta x$, $y_j = j\Delta y$ og $t_k = k\Delta t$.

For å faktisk løyse varmelikninga numerisk, trengs det den deriverte og dobbelderiverte, som vi tek med hensyn til $x$ i denne omgang. Vi veljer da $h$ som verdien for eit steg, som i dette tilfellet er $\Delta x$. Her nyttar vi sentraldifferansen både for den deriverte og den dobbeltderiverte seinare.

$$u_x(x,t) = \frac{\partial u}{\partial x} = \frac{u(x_{i+h_x}, y_j, t_k) - u(x_{i-h_x}, y_j, t_k)}{2h_x} \tag{2.3}$$

Den dobbelderiverte blir så uttrykt ved

$$u_{xx}(x,t) = \frac{\partial^2 u}{\partial x^2} = \frac{u(x_{i+h_x}, y_j, t_k) - 2u(x_i, y_j, t_k) + u(x_{i-h_x}, y_j, t_k)}{h_x^2} \tag{2.4}$$

Videre kombinerast denne likninga med den generelle løysinga i 2 romslege dimensjonar, Likning (2.2), som gir oss den eksplisitte numeriske løysinga for varmelikninga i eit plan.

$$\frac{u_{i,j,k+h_t} - u_{i,j,k}}{h_t} = \alpha\left(\frac{u_{i+h_x,j,k} - 2u_{i,j,k} + u_{i-h_x,j,k}}{h_x^2} + \frac{u_{i,j+h_y,k} - 2u_{i,j,k} + u_{i,j-h_y,k}}{h_y^2}\right) \tag{2.5}$$

Her kan eventuelt alle $h_z$ og representerast som $\Delta z$, for klarheits skyld. Det er denne likninga som alltid tillet ein å vite kva temperaturen i eit punkt i gitteret over planet vil vere, i det neste tidssteget.

# 3 Metode

## 3.1 Python

Det fyrste steget for å simulere den to-dimensjonelle varmelikninga i Python vil vere å definere dei fysiske verdiane for objektet vi simulerar. Dei eksakte verdiane blir bestemt seinare, då ein berre treng variblane for termisk konduktivitet og lengde, $\alpha$ og *lengde*. Vidare, definerast verdiane nytta i sjølve simuleringa, som antallet punkt som skal undersøkjast på objektet, og for kor mange sekund det skal simulerest, *punkt* og $t$.

Ein byrjar så å utleie prosessen for simulering ved å etablere tids- og rom-stegene, som tillet oss å approksimere alle punkt i gitteret som er defineret. $dx$, $dy$ og $dt$.

Definer så initial- og randbetingelsane, og definer plottet ved hjelp av Matplotlib.pyplot. Så definerast ei while-løkke, som itererar så mange tidssteg vi ønskjar. Så definerast den dobbeltderiverte likninga i både x- og y-retning, før den leggast inn i Likning (2.2) for den endelege løysinga. Vidare, så gjenstår det berre å plotte denne i grafen.

## 3.2 Reell anvendelse

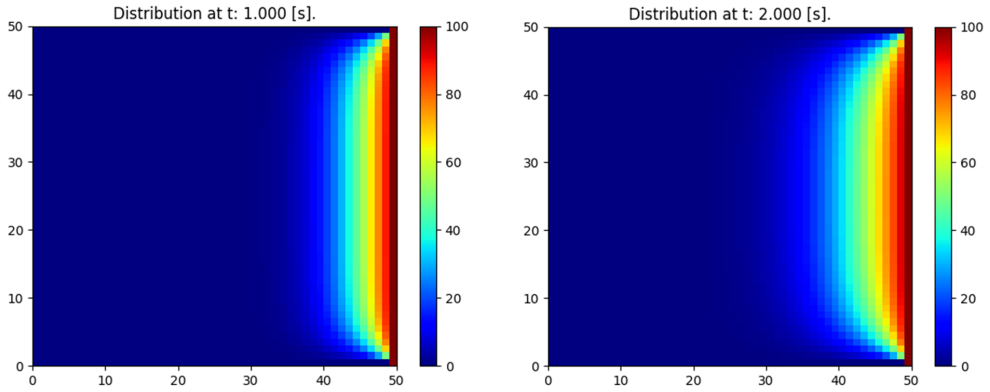
Vi veljer då ei reell problemstilling, hovudsakleg for å gjere det lettare of visualisere det som foregår. Vi ser for oss ei kvadratisk jarnplate, 5x5 cm, som har ein romtemperatur på 25 °C. Vi held så ein varm stang, 100 °C, langs eine sida av plata, og observerar varmeendringa gjennom plata.

Jarnplata har då ein termisk diffusivitet, $\alpha$, på 23 $\text{mm}^2/\text{s}$, og eit areal på 25 cm. Vi har ovenfor definert initial- og randbetingelsar, kvar den initielle temperaturen er 0 °C, uniformt over heile plata, og alle randbetingelsar er ingenting, bortsett frå den eine sida, kvar vi held den varme stanga inntil.

Desse fysiske verdiane som areal og termisk diffusivitet skal vi ikkje endre på. Initial- og randbetingelsar derimot, kan lett endrast på for å modellere ulike situasjonar av varmelikninga. Dette har blitt gjort i vedlagte figurar i seksjonen under.

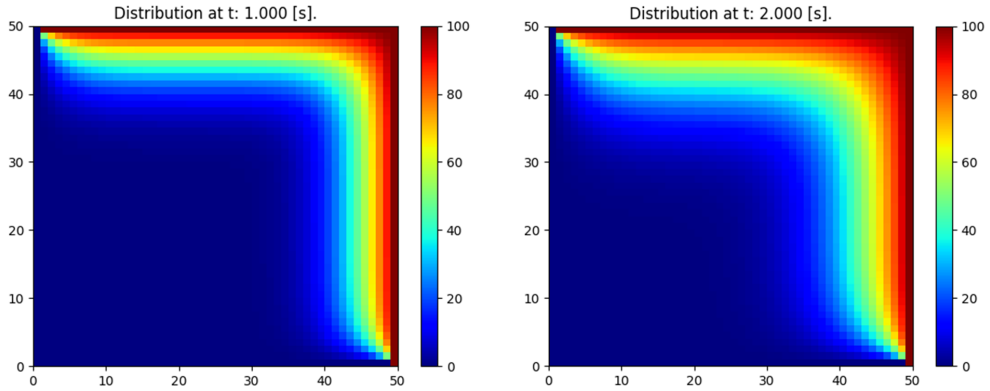
# 4 Resultat

Først simulerast situasjonen beskrive ovenfor, under 'Reell Anvendelse.'



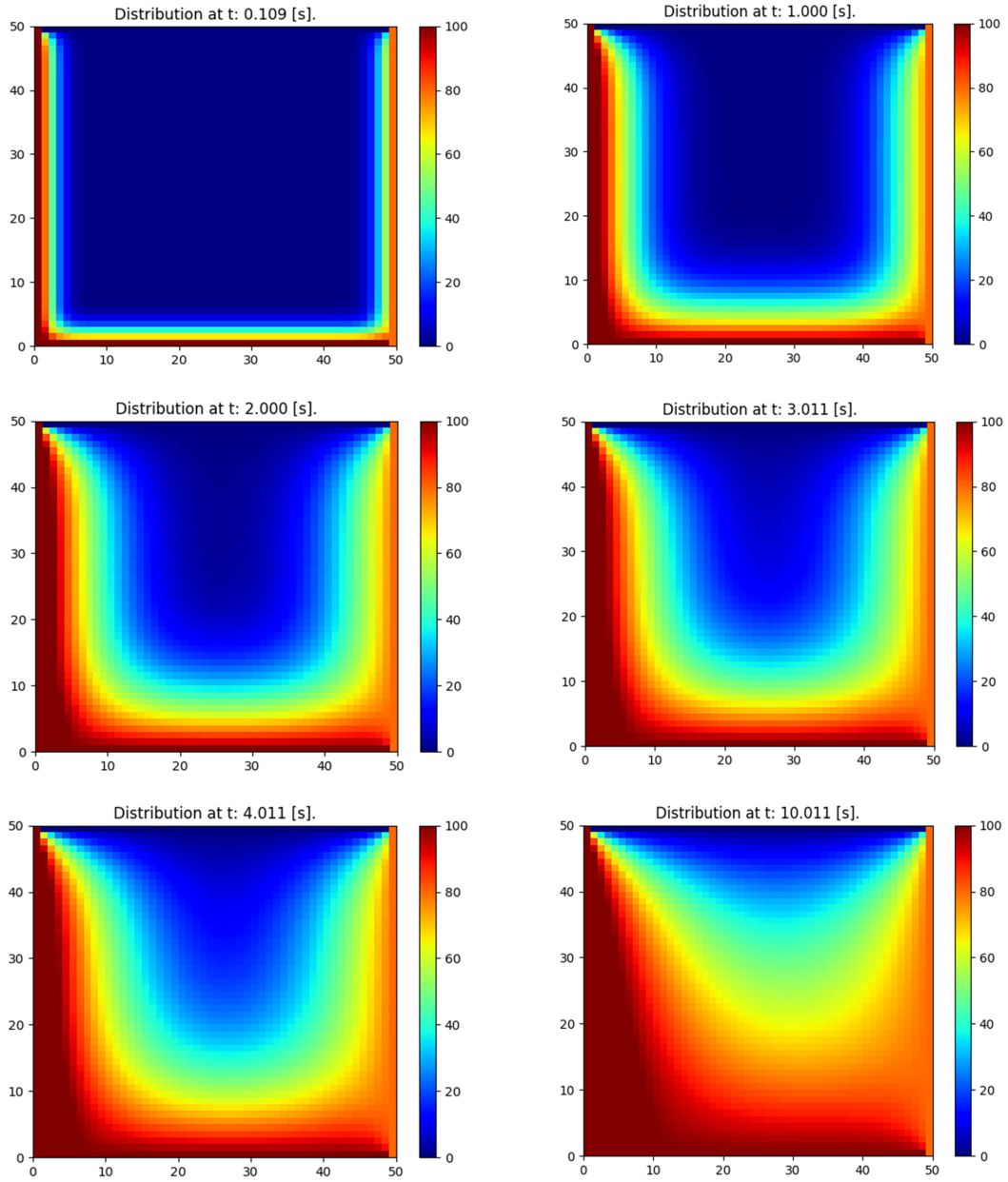

**Figur 4.1:** Varmediffusjon til jarnplate beskrive under 'Reell Anvendelse.' Figuren viser utvikling ved både 1 og 2 sekund etter initialtilstanden.

For den neste simulering blir ei tili varm stang lagt langs planet, vedsidesliggande av den originale stanga.




**Figur 4.2:** Varmediffusjon til jarnplate beskrive over. Figuren viser utvikling ved både 1 og 2 sekund etter initialtilstanden.

Vidare kan det betemmast eit litt meir variert sett randbetingelsar til jarnplata. Her har vi ei side som er ein uniform 120 °C, ei vedliggande side med 100 °C, og den siste sida med 80 °C. Det observerast at varmeflyten framstår som realistisk i alle simuleringar hittil. For å fastslå at resultata gjenspeglar røynda krevast truleg ein praktisk gjennomførsel.




**Figur 4.3:** Varmediffusjon til jarnplate beskrive over. Figuren viser utvikling ved eit breiddt utval tidspunkt etter initialtilstanden. Dei ulike tidspunkta er oppgitt i figuren.

# 5 Vedlegg A

```
"""
Forfattar : Alex Aasen
Dato : 23.04.24
Formål : Numerisk løysing av Varmelikninga i 2-dimensjonelt rom
"""

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

```
# Fysikalske verdiar

alfa = 23 # jarn
lengd = 50 # mm

# Simuleringsverdiar

punkt = 50
tid = 1 # sek

# Steg og Initialbetingelsar

dx = lengd / punkt
dy = lengd / punkt

dt = min( dx**2 / (4 * alfa), dy**2 / (4 * alfa))

t_punkt = int(tid/dt)

u = np.zeros((punkt, punkt)) # initialtemp. 0 grader C

# Randbetingelsar

u[0, :] = np.linspace(100, 100, punkt)
u[-1, :] = np.linspace(0, 0, punkt)

u[:, 0] = np.linspace(120, 120, punkt)
u[:, -1] = np.linspace(80, 80, punkt)

# definerar plot

fig, axis = plt.subplots()

pcm = axis.pcolormesh(u, cmap=plt.cm.jet, vmin=0, vmax=100)
plt.colorbar(pcm, ax=axis)

# simulering, nyttar eksplisitt metode

tid_løpt = 0

while tid_løpt < tid :

    w = u.copy()

    for i in range(1, (punkt - 1)):
        for j in range(1, (punkt - 1)):

            dd_ux = (w[i+1, j] - 2*w[i, j] + w[i-1, j]) / dx**2
            dd_uy = (w[i, j+1] - 2*w[i, j] + w[i, j-1]) / dy**2

            u[i, j] = dt * alfa * (dd_ux + dd_uy) + w[i, j]

    tid_løpt += dt
```

```
    #plotting

    pcm.set_array(u)
    axis.set_title("Varmediffusjon ved: {:.3f} [s].".format(tid_løpt))
    plt.pause(0.01)


plt.show()
```

Trondheim, 23. april 2024

Alex Aasen

## Referanser