Prosjekt Matte 4106

Prosjektet: Plotter varmeligningen i Python, kjører et eksperiment i virkelige liv og sammenligner verdier og grafer.

Fremgangsmåte. Varmer opp en kjele med vann, putter en av sidene av vår stang inn i vannet og ser hvordan varmen endrer seg på forskjellige plasser (25, 50, 75, 100cm) på stangen. Bruker en lasertemperaturmåler til å måle temperatur.

1 Introduksjon

Denne oppgaven så skal vi sammenligne hvordan en simulering av varmeligningen ser ut i forhold til den virkelige verden. Du vil i løpet av vår rapport se hvordan våre fremgangsmåter endres ut ifra hvor mye alkohol vi har innabords. Vi er tross alt ingeniører ved NTH.

2 Koding og morro

Kode for numerisk løsning av varmelikningen:

```
length = 0.75
alfa = 97E-6
temp_left = 25
temp_right = 100
total_time = 1000
dx = 0.05
x_vec = np.linspace(0, length, int(length/dx))
dt = 0.01
t_vec = np.linspace(0, total_time, int(total_time/dt))
u = np.zeros([len(t_vec), len(x_vec)])
```

Denne delen av koden er initieleverdier og konstanter

Length er lengen på stanga i meter

Alfa er alfa fra varmelikningen, og heter termisk diffusivitet. Verdien ble funnet på wikipedia

Temp_left er varme til stangen ved t=0

Temp_right er varme til delen av stangen som er i det kokende vannet med en konstant temp på 100c

dx er steg lengden i posisjon

x_vec er et array med som deler lengden på dx

dt er steg lengden i tid

t_vec er et array med som deler tid på dt

u er et tomt 2d array med lengden til x_vec og t_vec

```
u[0, :] = 25
u[0,0]=25
u[:,-1]=100
```

her defineres tempraturen til alle posisjoner ved tid=0 til 25C

og den siste posisjonen, posisjonen lengst til høyre, har temperaturen 100c for alle t

```
for t in range(1, len(t_vec)):
        for x in range(1, len(x_vec)-1):
                u[t, x] = alfa * (dt / dx**2) * (u[t-1, x+1] - 2*u[t-1, x] + u[t-1, x-1]) + u[t-1, x]
#oppdater venstre grenseverdi
        #u[t, 0] = 0.5 * (u[t, 1] + u[t-1, 0])
#oppdater høyre grenseverdi
        u[t, 0]=temp_left
        u[t, -1] = temp_right
```

her iterer vi over t og x og oppdater u for hver verdi u(t,x). I stedet for å bruke u(t+i,x) bruker vi u(x,t) og juster alle verdiene i utrykket. Dette er fordi vi har verdier for t=0, men ikke for t+1, som gjør at den har start verdier for å t.

```
        #u[t, 0] = 0.5 * (u[t, 1] + u[t-1, 0])
```

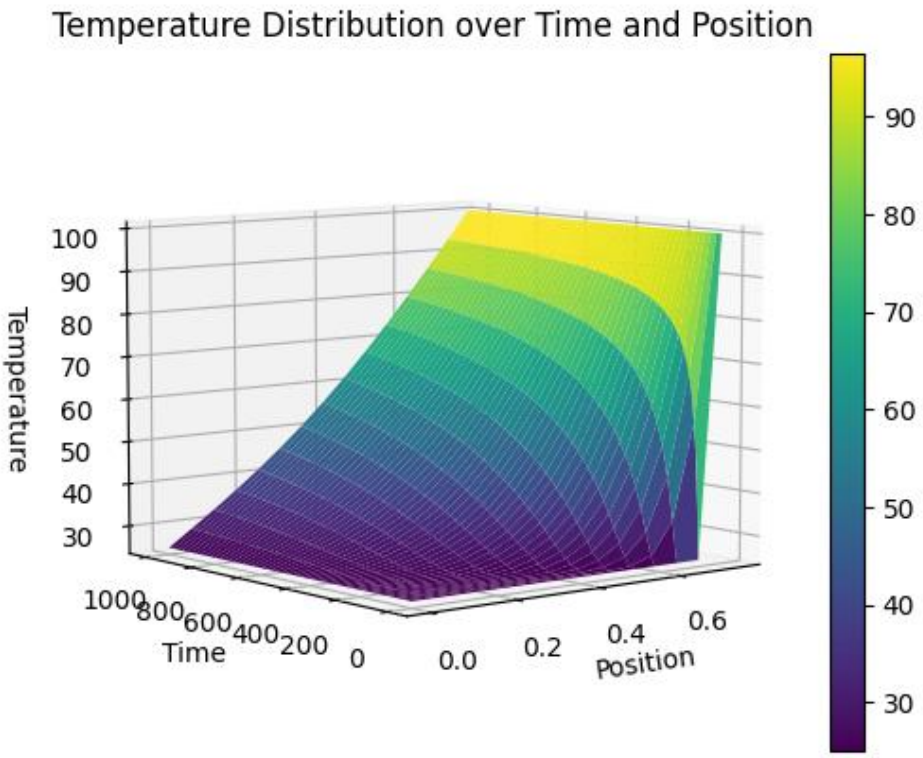
For at verdiene på venstregrensen ikke skal være konstant, må den oppdateres for seg selv. Dette gjøres ved å ta gjennomsnittsverdien til det mellom den forrige temperaturen til venstregrensen og den nåværende tempen til punkt en til høyre (punkt 1)

```
        u[t, 0]=temp_left
```

oppdaterer verdien til høyre grense til å alltid være 100

```
X, T = np.meshgrid(x_vec, t_vec)
# Plot 3D surface
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
surf = ax.plot_surface(X, T, u, cmap='viridis')
ax.set_xlabel('Position')
ax.set_ylabel('Time')
ax.set_zlabel('Temperature')
ax.set_title('Temperature Distribution over Time and Position')
fig.colorbar(surf)
plt.show()
```

alt er kode for å plotte i et 3d plot



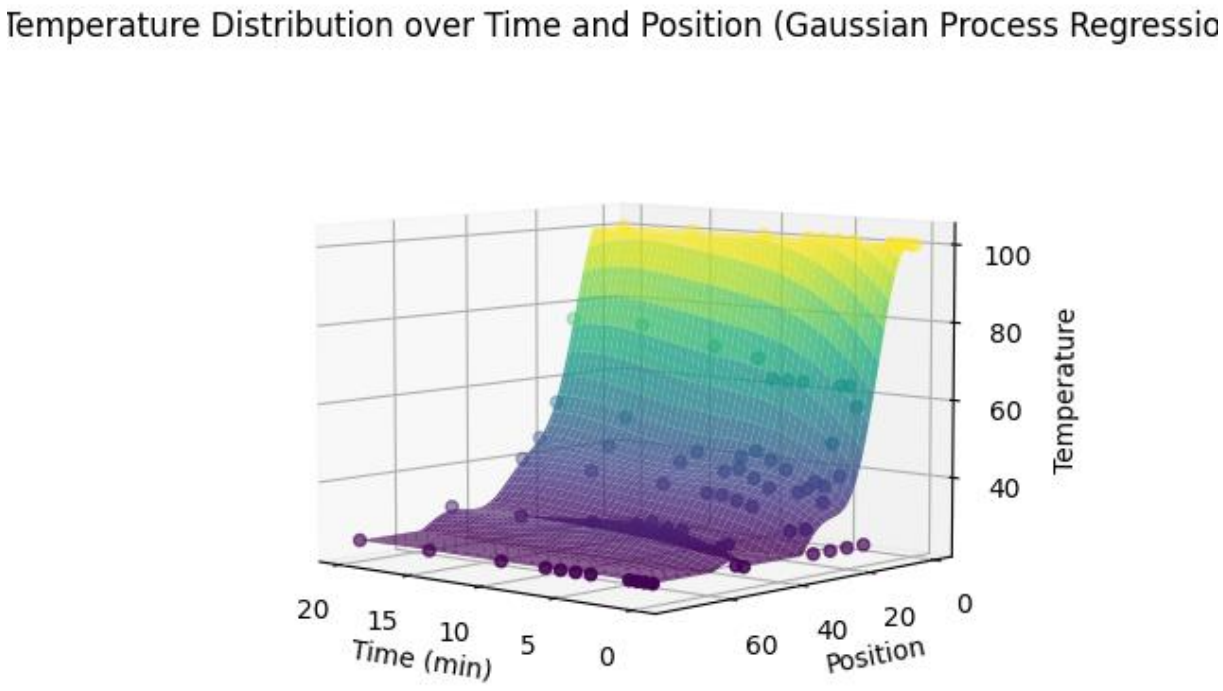
Kode for plotting av punkter med Gaussian prosess regresjon:

Denne koden lager et plan ved hjelp av Gaussian prosess regresjon for de punktene vi fant. Denne metoden blei foreslått av ChatGPT for regresjon av plan. Med hjelp av ChatGPT fikk vi koden til å funke, så vi kunne sammen likne planene våres.

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import RBF, WhiteKernel
```

```
data = np.array([
[100, 25, 25, 25, 25, 25, 25],
[100, 60, 43, 37, 31, 25, 25],
[100, 65, 51, 42, 40, 30, 25],
[100, 65, 40, 40, 30, 29, 25],
[100, 65, 43, 39, 35, 32, 25],
[100, 65, 45, 41, 36, 32, 25],
[100, 65, 47, 43, 37, 33, 25],
[100, 70, 45, 42, 37, 32, 25],
[100, 72, 45, 43, 38, 31, 25],
[100, 76, 52, 45, 39, 30, 25],
[100, 76, 54, 45, 40, 30, 25]
])

tid = np.array([0, 0.5, 1, 1.5, 4, 5, 6, 7, 10, 15, 20])
pos = np.array([0, 15, 20, 25, 30, 50, 75])

# Create an array to store all points
points = []

# Iterate through data and create points
for i, t in enumerate(tid):
for j, p in enumerate(pos):
temp = data[i, j] # Access temperature value from data array
points.append([t, p, temp])

points = np.array(points)

# Perform Gaussian Process Regression
X = points[:, :2] # Features (time and position)
y = points[:, 2] # Target (temperature)

kernel = 1.0 * RBF(length_scale=1.0, length_scale_bounds=(1e-1, 10.0)) +
WhiteKernel(noise_level=1e-5)
model = GaussianProcessRegressor(kernel=kernel, n_restarts_optimizer=10)
model.fit(X, y)

# Plotting
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Generate grid of points for plotting surface
t_grid, p_grid = np.meshgrid(np.linspace(tid.min(), tid.max(), 100),
np.linspace(pos.min(), pos.max(), 100))
X_pred = np.column_stack((t_grid.ravel(), p_grid.ravel()))
y_pred, sigma = model.predict(X_pred, return_std=True)
y_pred = y_pred.reshape(t_grid.shape)

# Clip predicted temperatures to ensure they don't fall below 25 and exceed 100
y_pred_clipped = np.clip(y_pred, 25, 100)
```

```
# Plot surface
ax.plot_surface(t_grid, p_grid, y_pred_clipped, cmap='viridis', alpha=0.8)

# Scatter plot original data points
ax.scatter(points[:, 0], points[:, 1], points[:, 2], c=points[:, 2], cmap='viridis')

# Set labels and title
ax.set_xlabel('Time (min)')
ax.set_ylabel('Position')
ax.set_zlabel('Temperature')
ax.set_title('Temperature Distribution over Time and Position (Gaussian Process
Regression)')

plt.show()
```



3 Konklusjon

Vi fikk ikke helt likt plot eller verdier som vi hadde forventet og vi kom fram til at det er flere grunner til at utfallet ble slik. Røret vårt var veldig tynt i tillegg til at det var hult. Derfor var det veldig sensitivt for endring av temperaturen i rommet. Røret var festet i en kjøkkenvifte (som du Morten vil se på videoen) som var kald som videre da kjølte ned

deler av stangen. Dette vil jo da kjempe mot varmen som kommer fra vannet, og jo lengre opp på stangen vi kom jo mer avvikende temperaturer fikk vi. I tillegg til dette kom det varm damp fra det kokende vannet vårt (selvfølgelig, men dette tenkte vi ikke på før vi utførte forsøket og så en spennende røyk som steg opp fra kjelen), og dette påvirket også temperaturen vi målte på røret vårt. Vi gjorde tre forskjellige målinger der vi fikk tre forskjellige verdier, og bare for å understreke hvor sensitivt det røret er, så ble en dør åpnet under et av forsøkene og de verdiene vi fikk var helt ute å kjøre (Magnus åpnet døren.). Også er du jo godt kjent med Schrødinger-likningen (neste ukes matte). Derfor fastslo vi at ikke kan vi ikke kan vite rørets posisjon og temperatur samtidig. Derfor var det vanskelig og få en måling av temperaturen på en gjenstand som vi ikke helt vet hvor er, når vi ville vite dens faktiske temperatur. Og helt til slutt så vil jeg bare henvise til introduksjonen, og dermed bare dele at denne konklusjonen ble skrevet av 5 gutter med en gjennomsnitts-promille på 2.4. Deretter håper vi at du tar dette i betraktning og gir oss bestått selv om konklusjonen ble hemmet av et alkoholinntak som bare er sunt for stunder ved NTH

Med hjertelig hilsen: Peder Haagensen, Marcus Aslaksby Andersen, Martin Markussen, Magnus Hansen og Jakob Eidsaune <3

Kilde

https://en.wikipedia.org/wiki/Thermal_diffusivity

https://medium.com/@matiasortizdiez/beginners-introduction-to-natural-simulation-in-python-i-solving-the-heat-equation-bf0ae5d4c37f