Standardprosjekt i TMA4106

Forfattere:
Gavin Minjae Kim, Andreas Hovland Skaane
Vincent Vågeskar, Athusan Kailainathan

Semester: Våren 2025

Innledning

Vi har valgt standardoppgaven, siden vi ikke kom på noe mer interessant oppgave. Dette semesteret har vært mer travelt enn det forrige, men vi er motivert til å gjøre denne oppgaven så godt vi kan.

Denne oppgaven handler om derivasjon av funksjoner med to variabler og partielle differensialligninger. Vi skal også løse varmelikningen numerisk basert på metodene som vi lærte i TMA4101.

Oppgave 1

Vi må først snakke litt om eksponentialfunksjonen. Formelen er: $f(x) = e^x$. Denne funksjonens spesielle egenskap er at den er identisk med sin deriverte og integrerte (når kjernen er 1). Dette kan bevises ved å derivere e^x ved hjelp av definisjonen av den deriverte.

$$f'(x) = \lim_{h\to 0} \frac{f(x+h) - f(x)}{h} = \lim_{h\to 0} \frac{e^{x+h} - e^x}{h} = \lim_{h\to 0} \frac{e^x \cdot e^h - e^x}{h} = \lim_{h\to 0} \frac{e^x \cdot (e^h - 1)}{h}$$

Vi vet at

$$e = \lim_{n\to\infty} (1 + \frac{1}{n})^n$$

Vi kan substituere $n = 1/h$. Da blir uttrykket:

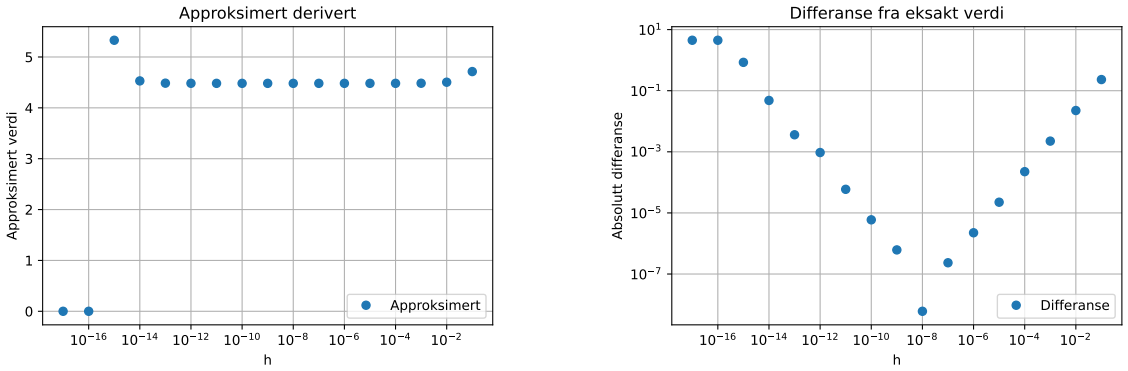
$$e = \lim_{h\to 0^+} (1+h)^{1/h} \Rightarrow e^h = \lim_{h\to 0^+} (1+h)$$

Da kan vi gjøre følgende:

$$f'(x) = \lim_{h\to 0} \frac{e^x \cdot (e^h - 1)}{h} = e^x \cdot \lim_{h\to 0} \frac{(1+h) - 1}{h} = e^x \cdot \lim_{h\to 0} \frac{h}{h} = e^x$$

Vi kan gjøre en tilnærming av den deriverte ved å bruke definisjonen av den deriverte. Vi skal lage en pythonscript som bruker forskjellige verdier av h.

Dette er plottene som vi fikk fra pythonscriptet.



(a) Approksimert derivert

(b) Differanse fra eksakt verdi

Figure 1: Sammenligning av resultater

I denne koden blir tilnærmingen uttrykt slik:

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

Det minste h kan bli før koden kollapser er når $h \leq 10^{-16}$. Her ser vi at differansen mellom den approksimerte deriverte og den eksakte er lavest når $h = 10^{-8}$. Dette skyldes at float-tall i Python ikke kan uttrykkes nøyaktig, men har en begrenset presisjon på ca. 15-16 desimaler. Når h blir for liten ($< 10^{-8}$), begynner avrundingsfeilene å dominere beregningen.

Oppgave 2

Av det obligen forteller oss er at vi kan uttrykke $f(x+h)$ som en taylorrekke utviklet ved punktet $a = x$. Taylorrekke utviklet ved punktet a er formulert som dette:.

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)(x-a)^2}{2!} + \frac{f'''(a)(x-a)^3}{3!} + ... = \sum_{n=0}^{\infty} \frac{f^n(a)(x-a)^n}{n!}$$

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

$$\Rightarrow f(x) + f'(x)h + \frac{f''(x)h^2}{2!} + \frac{f'''(x)h^3}{3!}...$$

Vi skal nå bruke en bedre tilnærming. Den nye deriverte er utrykt slik:

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

For å utlede den foregående formelen må vi først uttrykke $f(x-h)$ som en taylorrekke.

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

$$\Rightarrow f(x) - f'(x)h + \frac{f''(x)h^2}{2!} - \frac{f''(x)h^3}{3!} + ...$$

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

Her ser vi at for leddene som ikke er derivert eller er derivert med partall antall ganger kansellerer hverandre. Leddene som er derivert oddetall antall ganger blir summert.

$$f(x+h) - f(x-h) = 2f'(x)h + \frac{2f'''(x)h^3}{3!} + \frac{2f'''''(x)h^5}{5!} + ...$$

$$\frac{f(x+h) - f(x-h)}{2h} = f'(x) + \frac{f'''(x)h^2}{3!} + \frac{f'''''(x)h^4}{5!} + ...$$

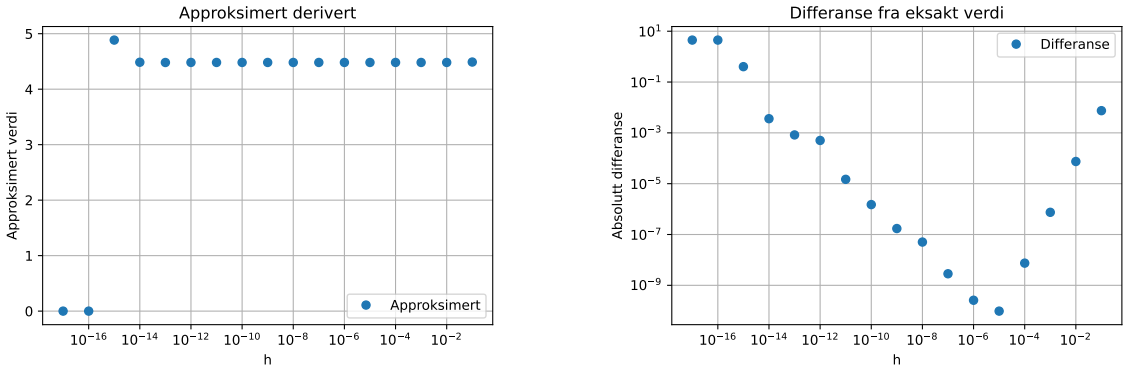
Det betyr at usikkerhet til den deriverte kan uttrykkes slikt:

$$O(h) = \frac{f'''(x)h^2}{3!} + \frac{f'''''(x)h^4}{5!} + \frac{f'''''''(x)h^6}{7!} + ...$$

$$\Rightarrow \sum_{n=3,5,7,...}^{\infty} \frac{f^n(x)h^{n-1}}{n!}$$

Man kan se på formelen at den nye usikkerheten mye mindre enn usikkerheten til $f'(x) = \frac{f(x+h)-f(x)}{h}$. Det er uttrykt slikt:

$$O(h) = \frac{f''(x)h}{2!} + \frac{f'''(x)h^2}{3!} + \frac{f''''(x)h^3}{4!} + ... = \sum_{n=2}^{\infty} \frac{f^n(x)h^{n-1}}{n!}$$



(a) Approksimert derivert

(b) Differanse fra eksakt verdi

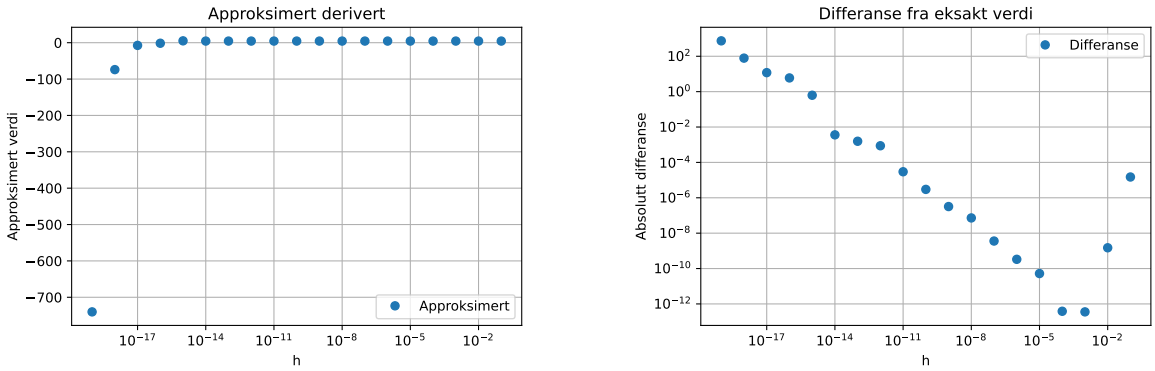
Figure 2: Sammenligning av resultater

I dette plottet ser vi at den minste differansen mellom tilnærmingen av den deriverte er mye mindre enn i den forrige tilnærmingen. Her er h-verdien som gir minst differanse større enn den tilsvarende verdien i det forrige plottet. Tilnærmingsfeilen øker også tidligere sammenlignet med den forrige approksimasjonen av den deriverte.

Oppgave 3

Denne oppgaven introduserer en enda mer nøyaktig tilnærming av den deriverte. Formelen er som følger:

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



(a) Approksimert derivert

(b) Differanse fra eksakt verdi

Figure 3: Sammenligning av resultater

Denne tilnærmingen har minst differanse mellom den eksakte deriverte ved 10^{-3}, mens den forrige var 10^{-5}. Absoluttverdien til differansen begynner også å nærme seg maskinpresisjon til flyttall som er 10^{-16}.

Oppgave 4

Nå skal vi løse varmelikningen numerisk på tre forskjellige måter. Den første måten er Eulers eksplisitte metode. Før vi begynner skal h være definert som gitteravstand for posisjon og k være gitteravstand for tid. Varmelikningen er definert slik:

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}$$

Man kan bruke andre ordens sentraldifferanse for å kunne uttrykke en tilnærming til $\frac{\partial^2 u}{\partial x^2}$.

$$\frac{\partial^2 u}{\partial x^2} = \frac{u(x+h,t) - 2u(x,t) + u(x-h,t)}{h^2}$$

Vi setter inn denne likningen i den forrige. Dessuten skriver i $u(x,t)$ som $u_{i,j}$. I er indeks for posisjonen i en endimensjonal stang. J er tidsindeksen.

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

Før vi går videre skal vi klargjøre initialbetingelsene og randkravene for systemet:
Randkrav:

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

Initialbetngelser:

$$u(x,0) = \sin(\frac{\pi x}{L})$$

Å løse likningen for $u_{i,j+1}$ er enkelt siden man vet temperaturene for alle punktene i stangen:

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

Problemet med den eksplisitte metoden er at løsningen kan divergere hvis man velger feil $k = dt$. I pythonkoden er er k som er mindre enn en bestemt verdi som gjør at løsningen er stabil. Resultatene ser ganske lovende ut når lengden til stangen og antall punkter på stangen er like. Da er initialbetingelsen veldig bra og avkjølingen ser realistisk ut. Hvor raskt koden fullfører avhenger hvor mange punkter det er, altså hvor liten $dx = \frac{lengde}{punkter}$ er. Derfor kan man si at med mer punkter blir koden tregere, mens hvis lengden øker blir koden raskere. Det er fordi, hvis man øker oppløsningen i plottet, må man beregne temperaturen for flere punkter.

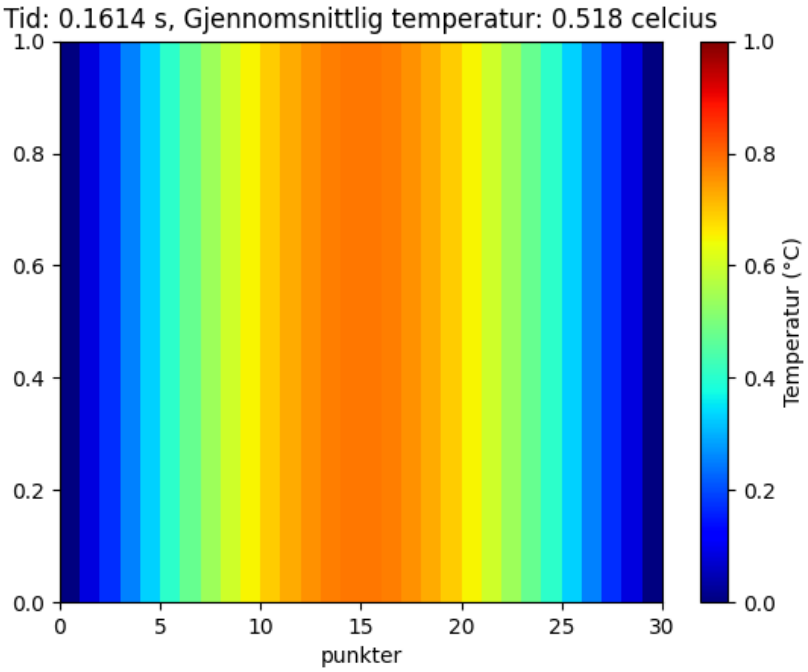


Figure 4: Et bilde av animasjonen

Oppgave 5

Den implisitte metoden er mer krevende å implementere. Det er på grunn av at det er mange flere ukjente i formelen.

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

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

Definerer

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

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

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

Alle disse verdiene på venstre side er ukjente. Det danner et likningssystem med N ukjente. Da er det en matrise med n x n dimensjoner.

$$\begin{bmatrix} 1+2\lambda & -\lambda & 0 & \cdots & 0 \\ -\lambda & 1+2\lambda & -\lambda & \cdots & 0 \\ 0 & -\lambda & 1+2\lambda & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & -\lambda \\ 0 & 0 & 0 & -\lambda & 1+2\lambda \end{bmatrix} \begin{bmatrix} u_1^{n+1} \\ u_2^{n+1} \\ u_3^{n+1} \\ \vdots \\ u_N^{n+1} \end{bmatrix} = \begin{bmatrix} u_1^n \\ u_2^n \\ u_3^n \\ \vdots \\ u_N^n \end{bmatrix}$$

Den implisitte løsningen har en veldig gunstig egenskap. Det er at vi trenger ikke å bekymre oss for at verdien av dt skal føre til at systemet divergerer. Det betyr at likningssystemet er ubetinget stabilt. Man kan bevise dette ved å bruke diskret maksimums prinsippet og trekantulikheten. (Shen, 2015)

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

Trekantulikhet:

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

$$\Rightarrow (1+2\lambda) \cdot \max_i |u_{i,j+1}| \leq \max_i |u_{i,j}| + \lambda \cdot \max_i |u_{i,j+1}| + \lambda \cdot \max_i |u_{i,j+1}|$$

$$\Rightarrow (1+2\lambda) \cdot \max_i |u_{i,j+1}| \leq \max_i |u_{i,j}| + 2\lambda \cdot \max_i |u_{i,j+1}|$$

$$\Rightarrow \max_i |u_{i,j+1}| \leq \max_i |u_{i,j}|$$

Ulikheten beviser at den implisitte metoden er stabil, fordi den neste verdien i tid er mindre enn den forrige. Det betyr at $u_{i,j+1}$ vil alltid være mindre enn $u_{i,j}$.

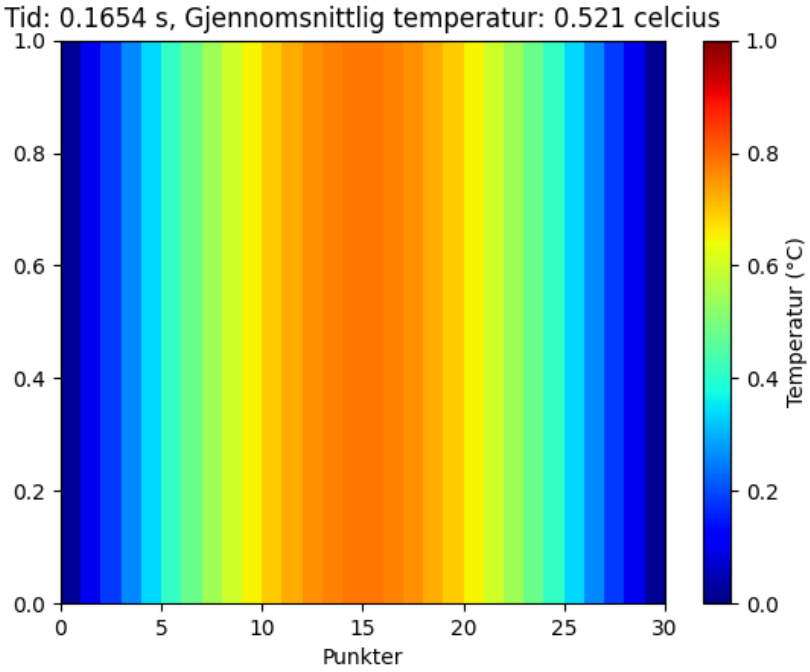


Figure 5: Et bilde av animasjonen

Koden for implisitt metode definerer dx som dette: $dx = \frac{lengde}{tidssteg}$. Akkurat som den implisitte koden vil økende lengde gjøre koden gå raskere, mens økende tidssteg gjøre koden gå tregere. Her kan man også gjøre dt så stort som en vil. Hvis man minker dt vil koden gå tregere siden det er flere tidspunkter der den må regne temperaturene. Hvis dt øker vil motsatt effekt skje.

Oppgave 6

Den siste numeriske metoden for å løse varmelikningen er crank-nicholson. Denne numeriske metoden kombinerer både eksplisitt og implisitt metode. Crank-nicholson blir skrevet slik:

$$\frac{u_{i,j+1} - u_{i,j}}{k} = \frac{\alpha}{2}\left(\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{u_{i+1,j+1} - 2u_{i,j+1} + u_{i-1,j+1}}{h^2}\right)$$

Der $k = dt$ og $h = dx$. Vi kan deretter gange k og definere $\lambda = \frac{k\alpha}{2h^2}$

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

Deretter sorterer vi u_{j+1} på venstre side og u_j på høyre side. Til slutt faktorisere leddene med felles faktor.

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

Dette blir en likningssystem der begge sidene har en matrise. $u_{i,j+1}$ er ukjente verdier og $u_{i,j}$ er kjente verdier.

$$\underbrace{\begin{bmatrix} 1+2\lambda & -\lambda & 0 & \cdots & 0 \\ -\lambda & 1+2\lambda & -\lambda & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -\lambda & 1+2\lambda & -\lambda \\ 0 & \cdots & 0 & -\lambda & 1+2\lambda \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} u_{1,j+1} \\ u_{2,j+1} \\ \vdots \\ u_{i-1,j+1} \\ u_{i,j+1} \end{bmatrix}}_{\mathbf{u}^{j+1}} = \underbrace{\begin{bmatrix} 1-2\lambda & \lambda & 0 & \cdots & 0 \\ \lambda & 1-2\lambda & \lambda & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \lambda & 1-2\lambda & \lambda \\ 0 & \cdots & 0 & \lambda & 1-2\lambda \end{bmatrix}}_{B} \underbrace{\begin{bmatrix} u_{1,j} \\ u_{2,j} \\ \vdots \\ u_{i-1,j} \\ u_{i,j} \end{bmatrix}}_{\mathbf{u}^{j}}$$

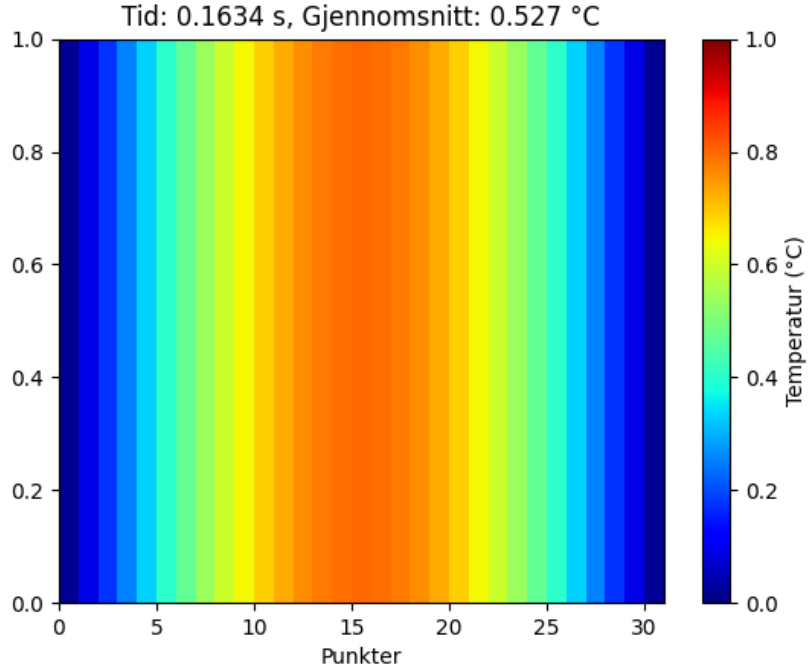


Figure 6: Et bilde av animasjonen

En måte å analytisk løse partielle differensiallikninger er å separere variablene til et produkt av to eller flere funksjoner. Da kan man definere varmelikningen slik: (Nome, 2015)

$$u(x,t) = z(t)y(x)$$

Fra definisjonen til varmelikningen kan vi løse dette både for $z(t)$ og $y(x)$.

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}$$

$$\Rightarrow \dot{z}(t)y(x) = z(t)y''(x) \Rightarrow \frac{y''(x)}{y(x)} = \frac{\dot{T}(t)}{a \cdot T(t)} = k$$

Når kan klarer å løse dette systemet, får man en formel på denne formen:

$$u_n(x,t) = b_n \cdot e^{-\alpha n^2 t} \cdot sin(nx)$$

der b_n er definert som

$$b_n = \frac{2}{L} \int_0^L f(x) sin(nx)\, dx$$

Det generaliserte uttrykket er en lineærkombinasjon for alle mulige løsninger:

$$u(x,t) = \sum_{n=1}^{\infty} b_n \cdot e^{-\alpha n^2 t} \cdot sin(nx)$$

Dette er fargeplottet som vi fikk.

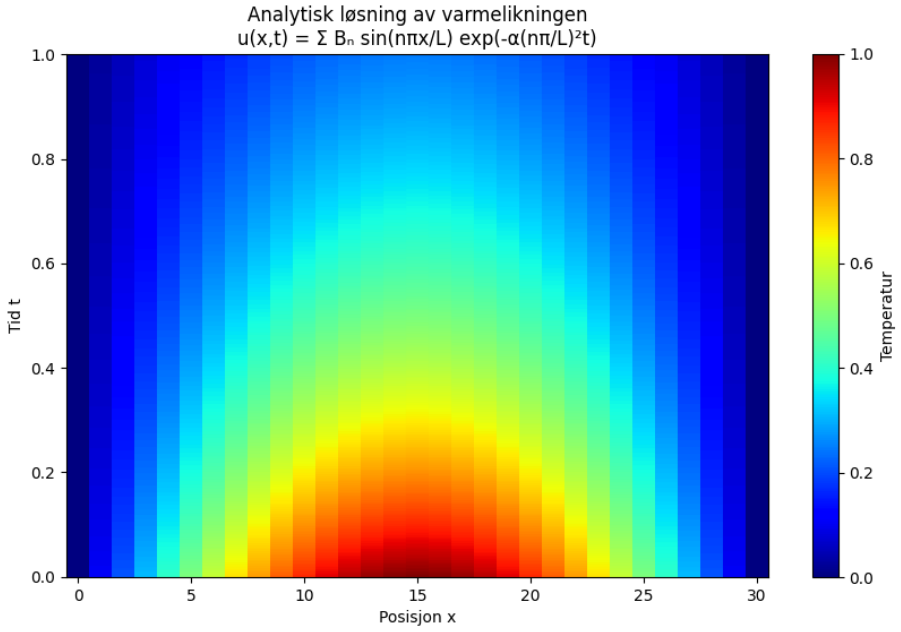


Figure 7: Det fullstendige bildet

Noe som vi la merke til er at alle tre numeriske løsningene avkjølte raskere enn den analytiske løsningen. Vi skal kopiere bilde til hvert numeriske metode når simuleringen er ferdig. Hvis du har lyst til å se hele simuleringen med dine egne øyne, kan du kopiere koden og kjøre den selv. Men vi tviler at du kommer til å gjøre det siden vi valgte standardoppgaven.
Vi skal nå sammenligne alle plotene ved 0.8 sekunder etter start.

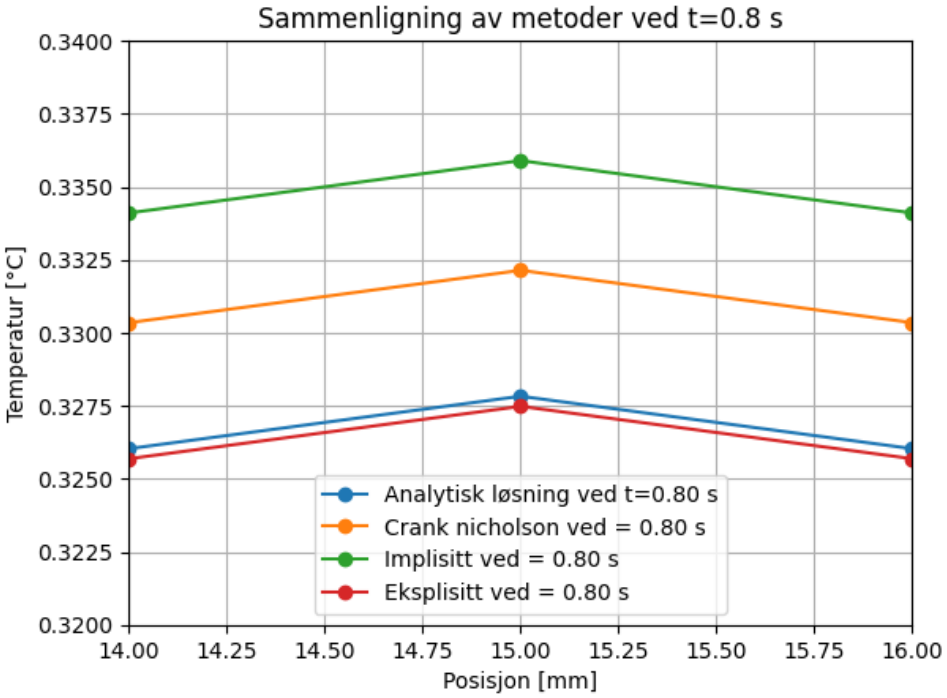


Figure 8: Sammenligning av alle metoder

Den mest ustabile metoden, eksplisitt er faktisk ved mest presise. Den implisitte som garanterer stabil løsning for alle dt er den mest upresise. Crank-nicholson som er en kombinasjon av eksplisitt og implisitt ligger midt i mellom eksplisitt og implisitt.

References

Shen, W. (2015, May 7). *ch11 8. Heat equation, implicit backward Euler step, unconditionally stable. Wen Shen* [Video]. YouTube.
Tilgjengelig på: `https://www.youtube.com/watch?v=GYVEScHVinI` (Funnet: 5 April 2025).

Nome, Morten A. (2025). *Tittelen på dokumentet: 2 - 7 - BØLGE- OG VARMELIKNINGEN*. NTNU. `https://mortano.folk.ntnu.no/stoff/2-7.pdf`