Jeg har valgt å gjøre standardprosjektet.

1.

Jeg tester formelen for mindre og mindre verdier for h. Resultatet er i denne tabellen:

|  |  |  |
| --- | --- | --- |
| *h* | Tilnærmet | Feil |
| 10-1 | 4.7134 |  |
| 10-2 | 4.5042 |  |
| 10-3 | 4.4839 |  |
| 10-4 | 4.4819 |  |
| 10-5 | 4.4817 |  |
| 10-6 | 4.4817 |  |
| 10-7 | 4.4817 |  |
| 10-8 | 4.4817 |  |
| 10-9 | 4.4817 |  |
| 10-10 | 4.4817 |  |
| 10-11 | 4.4817 |  |
| 10-12 | 4.4826 |  |
| 10-13 | 4.4853 |  |
| 10-14 | 4.5297 |  |
| 10-15 | 5.3291 |  |
| 10-16 | 0.0000 | 4.4817 |

Her kan man se at tilnærmingen er best med . I hvert fall er det sånn hvis jeg har skrevet Python-koden min riktig. Etter det blir feilen større, og når eller mindre, klarer ikke datamaskinen å skille fra 0.

2.

Jeg gjør samme eksperiment på nytt, men med formelen . Resultatet er i denne tabellen:

|  |  |  |
| --- | --- | --- |
| *h* | Tilnærmet | Feil |
| 10-1 | 4.4892 |  |
| 10-2 | 4.4818 |  |
| 10-3 | 4.4817 |  |
| 10-4 | 4.4817 |  |
| 10-5 | 4.4817 |  |
| 10-6 | 4.4817 |  |
| 10-7 | 4.4817 |  |
| 10-8 | 4.4817 |  |
| 10-9 | 4.4817 |  |
| 10-10 | 4.4817 |  |
| 10-11 | 4.4817 |  |
| 10-12 | 4.4822 |  |
| 10-13 | 4.4809 |  |
| 10-14 | 4.4853 |  |
| 10-15 | 4.8850 |  |
| 10-16 | 0.0000 | 4.4817 |

Her kan man se at feilen synker raskere enn i oppgave 1. Dette kan forklares med taylorrekker:

Når vi trekker disse fra hverandre, får vi .

Så deler vi på 2*h* som i formelen og får . Dette viser at feilen nå er proporsjonal med *h*2 istedenfor *h*.

Etter begynner feilen å bli større, og akkurat som i oppgave 1, går det helt galt når .

3.

Jeg har virkelig lyst til å slå på stortrommen, så jeg gjør eksperimentet nok en gang, men med formelen . Resultatet er i denne tabellen:

|  |  |  |
| --- | --- | --- |
| *h* | Tilnærmet | Feil |
| 10-1 | 4.4817 |  |
| 10-2 | 4.4817 |  |
| 10-3 | 4.4817 |  |
| 10-4 | 4.4817 |  |
| 10-5 | 4.4817 |  |
| 10-6 | 4.4817 |  |
| 10-7 | 4.4817 |  |
| 10-8 | 4.4817 |  |
| 10-9 | 4.4817 |  |
| 10-10 | 4.4817 |  |
| 10-11 | 4.4817 |  |
| 10-12 | 4.4826 |  |
| 10-13 | 4.4801 |  |
| 10-14 | 4.4853 |  |
| 10-15 | 5.1070 |  |
| 10-16 | -1.4803 | 5.9620 |
| 10-17 | -7.4015 | 11.8832 |

Her ser vi at feilen synker enda raskere enn for de to andre formlene. Allerede ved er feilen på sitt laveste, og deretter øker den igjen på samme måte som med de andre formlene. Denne gangen får vi ikke 0 ved , men vi får faktisk negative tall.

4.

Starter med å skrive om fra til der . Så var det å sette seg ned å finne ut av hvordan i alle dager man kan animere i Python. Det viste seg å være ganske greit å gjøre med matplotlib.animation.

Når jeg setter *h* og *k* lik hverandre, funker ikke koden hvis verdien er mindre enn 2. Det samme gjelder hvis jeg setter *k* større enn . Med andre ord virker det som om jeg trenger at . Det virker også som at jo større *h* er i forhold til *k*, jo saktere synker temperaturen. A screen shot of a computer program

AI-generated content may be incorrect.

A graph with a blue line

AI-generated content may be incorrect. A graph with a blue line

AI-generated content may be incorrect.

5.

Starter igjen med en omskrivning som er lettere å implementere i koden, men denne gangen blir det litt mer komplisert:

er fortsatt .

Når jeg setter *h* og *k* lik hverandre, funker koden for alle verdier jeg har prøvd (bortsett fra 0 selvfølgelig), og det virker ikke som om det er noen grenser for hva kan være med implisitt metode. På samme måte som med eksplisitt metode, vil temperaturen synke raskere jo høyere *h* er i forhold til *k*. Nå som *k* kan være større enn , kan man også se at temperaturen synker raskere og raskere jo større *k* er i forhold til *h*.

A screen shot of a computer program

AI-generated content may be incorrect.

A graph with a blue line

AI-generated content may be incorrect. A graph with a blue line

AI-generated content may be incorrect.

Animasjonen oppfører seg på samme måte som med eksplisitt metode

6.

Jeg starter nok en gang med å skrive om formelen:

er fremdeles .

Det virker som at når jeg setter *h* = *k*, funker ikke koden hvis denne verdien er mindre enn 1.5. Grensen for hvor stor *k* kan være i forhold til *h* er ikke den samme her som i eksplisitt. Jeg har ikke klart å finne ut hva grensen er, men det ser ut til at den er litt mindre vrang enn eksplisitt.

A screen shot of a computer program

AI-generated content may be incorrect.

A graph with a blue line

AI-generated content may be incorrect. A graph with a blue line

AI-generated content may be incorrect.

Den analytiske løsningen av varmelikningen er . Her har jeg plottet feil i de tre numeriske løsningene med *h* = 0.1 og *k* = 0.001 i forhold til den analytiske løsningen:

A graph with different colored lines

AI-generated content may be incorrect.

Det virker som at Crank-Nicolson er best med tanke på at den gir lavest feil. Selv om implisitt Euler er fin å bruke med tanke på at valg av *h* og *k* ikke er særlig begrenset, viser det seg at den gir størst feil av de tre metodene.