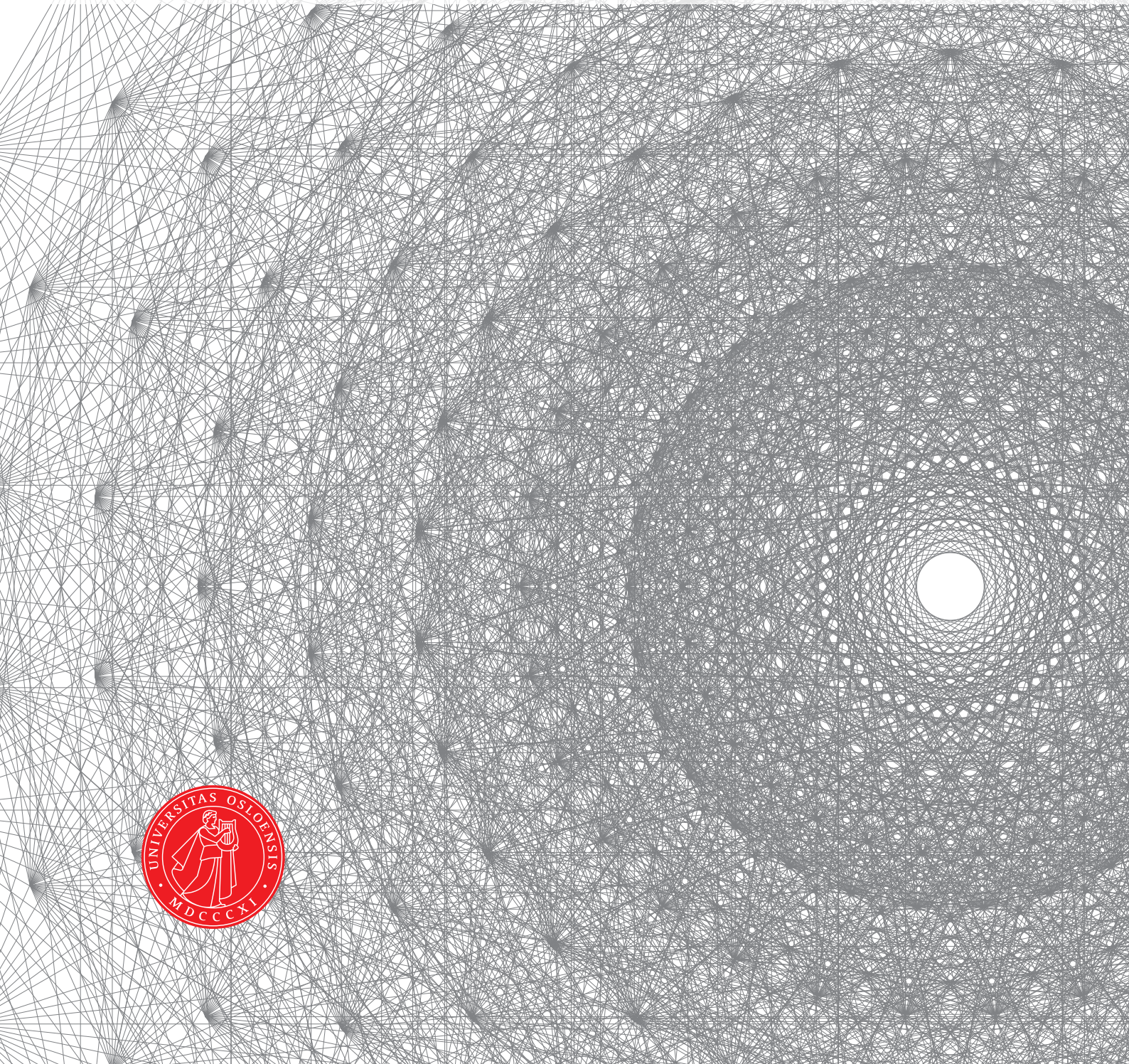
UiO : Matematisk institutt
Det matematisk-naturvitenskapelige fakultet

Positivitetsbevarende numeriske metoder

for parabolske differensiallikninger

Preben Olsen

Masteroppgave, våren 2023



Denne masteroppgaven er levert inn som en del av programspesialiseringen *Matematikk* under *Lektorprogrammet* ved Universitetet i Oslo. Oppgaven er normert til 30 studiepoeng.

Forsiden viser et utsnitt av rotsystemet til den eksepsjonelle liegruppen E_8, projisert ned i planet. Liegrupper ble oppfunnet av den norske matematikeren Sophus Lie (1842–1899) for å uttrykke symmetriene til differensiallikninger og spiller i dag en sentral rolle i flere deler av matematikken.

Abstract

Differential equations are often difficult, or impossible to solve analytically, from this arises the need for good numerical schemes. A numerical scheme is deemed good by conserving structures of and converging quickly towards its analytical counterpart. This dissertation explores four different schemes, namely the forward Euler, Crank-Nicolson, finite element method (FEM), and a mass-lumped FEM. Explicit conditions for positivity and stability of the schemes are sought. FEM and mass-lumped FEM attain positivity by implicit conditions depending upon the initial data, with the exception of an explicit condition for mass lumped FEM when the stiffness matrix is an M-matrix. Explicit conditions are given for conservation of positivity for forward Euler and Crank-Nicolson. By applying von Neumann stability-analysis explicit conditions for stability is found for all four schemes. A centered finite element method is proposed as a natural scheme to explore after the standard FEM. Insight is given on the characteristics of terms in the finite element schemes. Explicit conditions for the stiffness-matrix being an M-matrix has been shown. This characteristic is of particular interest in modelling elliptic differential equations preserving positivity. A link between conditions for positivity of the right-hand side consisting of the known terms, and stability of the finite element method is indicated. All four schemes are implemented in python, and errors are analysed by use of a weighted l^2 norm.

Sammendrag

Differensiallikninger er ofte vanskelige, eller umulige å løse analytisk, derfor er gode numeriske skjemaer er i mange situasjoner en nødvendighet. Et godt numerisk skjema er strukturbevarende, og konvergerer raskt mot den analytiske løsningen. I denne avhandlingen utforskes fire ulike skjemaer for numerisk løsning av varmelikningen med konveksjonsledd, forlengs Euler, Crank-Nicolson, endelige elementer(FEM), og mass-lumped endelige elementer. Eksplisitte positivitet- og stabilitetskrav søkes for de fire skjemaene.
FEM-skjemaet oppnår implisitte positivitetskrav avhengige av initialbetingelsene. , Mass-lumped FEM oppnår eksplisitte positivitetskrav når stivhetsmatrisen er en M-matrise, ellers har denne metoden også implisitte positivitetskrav. Eksplisitte stabilitetskrav er utledet for forlengs Euler og Crank-Nicolson. Von Neumann stabilitetsanalyse anvendes på alle skjemaene for å finne stabilitetskrav, hvor alle skjemaene oppnår eksplisitte krav for stabilitet. Et sentrert endelige elementer skjema foreslås som et naturlig å utforske etter standard FEM.
Arbeidet med FEM er gir innsikt i egenskaper ved de ulike leddene i skjemaet. Det er funnet eksplisitte krav som sikrer at stivhetsmatrisen er en M-matrise. Denne egenskapen er spesielt nyttig for posititetsbevarende elliptiske differensiallikninger. Sammenheng mellom positivitet av de kjente leddene i et FEM-skjema og stabilitet er indikert.
Skjemaene er implementert i python, og feilestimater gitt ved en vektet l^2-norm.

Anerkjennelser

Takk til min veileder Snorre Harald Christiansen, for matematiske pekepinner, og hyggelige samtaler. Spesielt takk for rommet som er gitt til utforsking, og autonomi i oppgaven.
Takk til min samboer Jenny, for støtten gjennom hele semesteret. Jeg lover å bidra mer hjemme etter oppgaven er levert!
Takk til Mailand vgs, for god kaffe og lån av et fantastisk kontor.
Takk til medstuder på Lektorkroken. Deres selskap har vært givende i en periode med mye selvstendig arbeid.
Til slutt takk til min familie, som alltid har troen på meg.

Innhold

Figurer

KAPITTEL 1

Introduksjon

1.1 Disposisjon

Avhandlingen er organisert som følger:

Kapittel 1 Kort redgjørelse av problem og tidlig historisk utvikling av differensiallikner, med fokus på varmelikningen.

Kapittel 2 Matematisk redgjørelse for parabolske differensiallikninger

Kapittel 3 Analytiske maksimumprinsipler for parabolske differensiallikninger.

Kapittel 4 Beskrivelse av oppbygning til fire ulike skjema som løser parabolske differensiallikninger uten reaksjonsledd. For alle fire skjema er det beskrevet krav for positivitet, og krav for stabilitet. Et skjema med endelige elementer er utforsket i særlig detalj. Alle skjema har tilhørende kode som ligger ved i appendiks B.

Kapittel 5 Sammenlikner feil og konvergens i de fire skjemaene beskrevet i kapittel 4.

1.2 Redgjørelse av problem

Det absolutte nullpunkt, bestemt som 0 Kelvin representerer en nedre grense for temperaturen til et objekt. Varmelikningen respekterer den nedre temperaturgrensen ved at positive initialdata og randbetingelser gir positive løsninger. Problemet som behandles i denne avhandlingen beskriver både diffusjon, og konveksjon for et rektangulært snitt. Konveksjonsleddet modellerer bevegelse i et medium, og gjør det mulig å studere varmeutvikling i både væsker og faste stoffer. En ønskelig egenskap for numeriske løsninger av varmelikningen er bevaring av strukturene i den analytiske løsningen. Spesielt er det av interesse at numeriske skjema bevarer positivitet.

1.3 En kort historisk redgjørelse

Historiekapittelets mål er å gi et tidsperspektiv for utviklingen av varmelikningen, som står sentralt for denne oppgaven. Det finnes ingen rette linjestykker fra start til slutt i utviklingen av et fagfelt, likevel gjør jeg et forsøk. En kort liste over sentrale matematikere som ikke er nevnt videre inkluderer, men er

ikke begrenset til: Pascal, Descartes, familien Bernoulli, D'Alembert, Laplace, og Dirichlet. Jeg håper ovenfornevnte ville hatt forståelse for ofrene som er gjort i forsøket på å holde introduksjonen kort. Sentralt i differensiallikninger står differensialer, det virker derfor som et naturlig startpunkt å snakke om utviklingen av kalkulus :

Newton

Det tidligste kjente eksempelet av differensialregning kommer fra et manuskript utgitt av sir Isaac Newton i 1665 [1, s. 321]. De tidligste utgitte dokumentene som inneholder Newtons $Method of\ fluxions$ er gitt ut i 1693, ni år etter Leibniz ga ut sin utgave av kalkulus [1, s. 347]. I ettertid blir Newton ansett som kalkulus' far. Til gjengjeld blir Leibnitz sin notasjon $\frac{dx}{dt}$, sammen med notasjonen til Lagrange $x'(t)$ brukt i mye større grad enn Newtons notasjon $\dot{x}$. Newtons ideer er i sin essens like det vi i dag kaller fundamentalteoremet til kalkulus. Newton finner arealet under en kurve ved å dele inn kurven i små biter og regner ut arealet av området produsert av førsteaksen, f(x), og f(x+ Δx). Deretter argumenterer Newton at forskjellen mellom de to funksjonsverdiene f(x), og f(x+ Δx) blir så liten at vi kan se bort fra den, og områdene kan behandles som rektangler. I forsøket på å gjøre utregninger av integralet rigorøst utforsket Newton forholdet mellom f(x) og f(x+ Δx), som han kaller en $fluxion$. Newton klarte så å regne ut arealet under en graf ved hjelp av fluxioner [2, s. 195]. Newton skrev også $Scala\ graduum\ Caloris.\ Calorum\ Descriptiones\ \&\ Signa$, utgitt i en samling av Newtons verker i 1744, som inneholder en tidlig versjon av Newtons avkjølingslov [22]:

$$\dot{T}_B = hA(T_B - T_E) \tag{1.1}$$

hvor T_B er temperaturen i til et objekt, h er en varmeledningskoeffesient, A er kontaktområdet mellom objektet og omgivelsene, og T_E er temperaturen til omgivelsene.

Euler og Lagrange

Leonhard Eulers verker er så mange at forsøk på å samle og oversette arbeidet hans har feilet [7]. Et av forsøkene på å samle verkene til Euler er $The\ Euler : Archive$ [24] som inneholder 855 verker, hvor 207 er oversatt. I 1755 ble $Institutiones\ Calculi\ Differentialis$ utgitt [7], som systematiserte, utbedret, og fullførte arbeidet påstartet av Newton og Leibniz. Påvirkningen av Eulers verker kan illustreres ved å spore globalt aksepterte og kjente notasjoner $\pi = 3.14...$, $i = \sqrt{-1}$, $e = 2.718...$, som konstanter, og $sin(\alpha)$, $cos(\alpha)$, $tan(\alpha)$ for trigonometriske funksjoner tilbake til Euler. Eulers systematisering og utbedring av nærmere samtlige områder innen matematikk på hans levetid. Parallelt med Joseph-Louis Lagrange utviklet Euler variasjonskalkulus[1] for å løse optimeringsproblemer, funnet i andre volum av $Miscellania\ Taurinensia$ [1]. Samlingen $Miscellania\ Taurinensia$ er gitt ut mellom 1758 og 1773. Andre relevante bidrag av Lagrange er beskrivelser av forplantning av bølger, og en generell beskrivelse av bevegelse, kalt lagrange-mekanikk[1].

Fourier og Biot

I 1807 utga Joseph Fourier *Sur la propagation de la chaleur* som foreslår likningen:

$$CD\frac{\partial T}{\partial t} = K\nabla^2 T \tag{1.2}$$

hvor D er densitet, C spesifikk varmekapasitet, og K er konduktivitet[12]. I *Sur la propagation de la chaleur* referer Fourier til Jean Baptiste Biot som inspirasjon for arbeidet. Biot hadde før utgivelsen av Fouriers verk om varme prøvd å beskrive varmekonduktivitet ved å utføre eksperimenter[14]. Fourier bekreftet eksperimentene til Biot, og utførte den analytiske beskrivelsen av varmekonduktivitet. På grunn av Fouriers generøse referanse kalls varmelikningen noen ganger for Fourier-Biot-likningen. Fourier ble kritisert av Siméon Denis Poisson for å anta formen til løsningen av varmelikningen, og dermed sjekke at dene formen gir korrekt løsning. Poisson fortsetter med å oppnå formen til løsningen av varmelikning uten å først anta hvordan løsningen vil se ut[12].

DEL I

The First Part

KAPITTEL 2

En kort historisk introduksjon og redgjørelse for problem

2.1 Byggeblokker

Hensikten til kapittel 2 er å oppnå forståelse for hva det innebærer at et problem er parabolsk. I læreboken *Partial Differential Equations* av Evans, L.C[8] er definisjon 2.4.1 gitt som en selvfølgelighet. Andre bøker om temaet som *Partial Differential Equations of Parabolic Type* av Friedman, A.[9] gir noe forklaring. Litt for fullstendighetens skyld, men mest for nyskjerrighet starter vi fra bunn av med parabler og ellipser i to romdimensjoner. Kapittel 2 vil munne ut i en definisjon av en parabolsk differensiallikning.

Parabler

Definisjon 2.1.1. En parabel i to romdimensjoner er kurven som ligger like langt unna et punkt og en rett linje.

En parabel i to dimensjoner kan genereres, som definisjon 2.2.1 sier, ved å velge et punkt og en linje for dermed å finne alle punkter som ligger like lang fra dem. Defines et fokuspunkt $F_p = (a, b)$, et punkt på parabelen $P_p = (x_0, y_0)$, og en linje L gitt ved $A_0x + B_0y + C_0 = 0$ vil distansen mellom punktet på parabelen og linjen gis ved[5]:

$$d(L, F_p) = \frac{|A_0x_0 + B_0y_0 + C_0|}{\sqrt{A_0^2 + B_0^2}}. \tag{2.1}$$

Distansen mellom punktet $P_p = (x_0, y_0)$ på parabelen og fokuspunktet $F_p = (a, b)$ finnes ved $|F_p - P_p|$ og vi har:

$$d(P_p, F_p) = |F_p - P_p| = 0.$$

som gir:

$$\frac{\sqrt{A_0^2x_0^2 + B_0^2y_0^2 + 2A_0B_0x_0y_0 + 2A_0C_0x_0 + 2B_0C_0y_0 + C_0^2}}{\sqrt{A_0^2 + B_0^2}} =$$

$$\sqrt{a^2 - 2ax_0 + x_0^2 + b^2 - 2by_0 + y_0^2}.$$

Multipliseres begge sider med seg selv, og ledd med lik grad samles finner vi:

$$-B_0^2x_0^2-A_0^2y_0^2+2A_0B_0x_0y_0+2(A_0C_0+a(A_0^2+B_0^2))x_0+2(B_0C_0+b(A_0^2+B_0^2))y_0+F=0.$$

hvor konstantene er samlet i en felles F. Vi ser at parametrene foran førstegradsleddene kan bestemmes vilkårlig, fordi de har ledd a, og b som ikke avhenger av parametrene til annegradsleddene. Skriver om parametre og får:

$$B_0^2x_0^2 + A_0^2y_0^2 - 2A_0B_0x_0y_0 + Dx_0 + Ey_0 + F = 0.$$

hvor annengradsleddene påvirker hverandre ved:

$$(-2A_0B_0)^2 = 4A_0^2B_0^2$$

Fortsettes opprydningen kan vi beskrive alle 2D parabler ved :

$$Ax^2 + Cy^2 \pm \sqrt{4 * A * C}xy + Dx + Ey + F = 0.$$

hvor $A = A_0^2$ og $C = B_0^2$. Dette beviser teorem 2.2.2.

Teorem 2.1.2. *En parabel er en likning på formen:*

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0,$$

som tilfredsstiller kravet

$$B^2 - 4AC = 0.$$

Ellipser

Definisjon 2.1.3. En ellipse i to romdimensjoner er en samling av punkter hvor forholdet mellom avstanden til et punkt og en rett linje er konstant.

Ellipser vil være nyttig senere i kapittelet, så vi utleder krav for disse. Vi kan låne noe av arbeidet som er gjort i forrige seksjon for avstanden mellom et punkt og en linje. Ved hjelp av definisjon 2.2.3 har vi:

$$\frac{\sqrt{A_0^2x_0^2 + B_0^2y_0^2 + 2A_0B_0x_0y_0 + 2A_0C_0x_0 + 2B_0C_0y_0 + C_0^2}}{\sqrt{(A_0^2 + B_0^2)}\sqrt{a^2 - 2ax_0 + x_0^2 + b^2 - 2by_0 + y_0^2}} = K.$$

Etter opprydning vil vi ende opp med:

$$(k_1 + B_0^2)x_0^2 + (k_2 + A_0^2)y_0^2 - 2A_0B_0x_0y_0 + Dx_0 + Ey_0 + F = 0.$$

hvor k_1 og k_2 er positive konstanter.
Sammenliknet med parabler ser vi at parametrene foran x^2 og y^2 har vokst, mens parameteren foran xy-leddet er uforandret, som betyr:

Teorem 2.1.4. *En ellipse er en likning på formen:*

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0,$$

som tilfredsstiller kravet

$$B^2 - 4AC < 0.$$

2.2 Overgang til flere dimensjoner

Som vi ser fra Teorem 2.2.2 er det andregradsleddene som bestemmer om vi har en parabel. Derfor vil vi i dette delkapittelet fokusere på disse, og se bort fra leddene med lavere grad. Ved å først utvide til tre dimensjoner (x, y, z) har vi:

$$f(x, y, z) = Ax^2 + Bxy + Cy^2 + Dxz + Eyz + Fz^2. \tag{2.2}$$

For at (2.2) skal gi oss en parabel må vi oppfylle parabel-kravet fra teorem 2.2.2. Det vil si:

$$B^2 = 4AC.$$
$$D^2 = 4AF.$$
$$E^2 = 4CF.$$

Som konsekvens kan vi skrive (2.2) som:

$$f(x, y, z) = (\pm\sqrt{A}x \pm \sqrt{C}y \pm \sqrt{F}z)^2. \tag{2.3}$$

Parabelkravet settes opp på tilsvarende måte uavhengig av antall dimensjoner. Vi observerer at (2.3) er et kvadrat, og vil alltid ha verdi ≥ 0. Om vi skriver (2.3) på matriseform vil parametrene danne en symmetrisk og positiv semi-definit matrise[11].

$$(a_1x_1 + a_2x_2 \cdots + a_nx_n)^2 = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix} \mathbf{A} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}. \tag{2.4}$$

hvor:

$$\mathbf{A} = \begin{bmatrix} a_1^2 & a_1a_2 & \dots & a_1a_n \\ a_2a_1 & a_2^2 & \dots & a_2a_n \\ & & \vdots & \\ a_na_1 & a_na_2 & \dots & a_n^2 \end{bmatrix}.$$

På grunn av parabel-kravet har $\mathbf{A}$ minst en egenverdi $\lambda = 0$ (dette sjekkes enkelt ved å se at det(A)=0). Gjøres samme prosessen med ellipser som for parabler vil vi istedet finne at alle egenverdiene til den tilhørende matrisen $\mathbf{A_0^*}$ er positive. Legger vi til enda en dimensjon - tiden, og bestemmer den tilhørende parameteren $a_{n+1} = 0$ har vi:

$$\mathbf{A_0^*} = \begin{bmatrix} a_1^2 + k_1 & a_1a_2 & \dots & a_1a_n & 0 \\ a_2a_1 & a_2^2 + k_2 & \dots & a_2a_n & 0 \\ & & \vdots & & \\ a_na_1 & a_na_2 & \dots & a_n^2 + k_n & 0 \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix}.$$

$k_1, k_2, ..., k_n$ er som tidligere positive konstanter. Matrisen er fortsatt symmetrisk og vi har tvunget inn en egenverdi $\lambda = 0$. Den nye matrisen oppfyller dermed samme type krav som vi hadde for parabler i (2.4)

2.3 De siste stegene

Vi bytter ut polynomene fra (2.4) med differensialer, og ser at nederste rad og borteste kolonne i $\mathbf{A_0^*}$ ikke tilføyer stort, utenom å gi oss en egenverdi lik 0. Vi gir derfor den tidsderiverte spesiell behandling og separer den fra differensialene i rom. Med utgangspunkt i formen til (2.4) sitter vi nå med en operator:

$$\frac{\partial}{\partial t} + \begin{bmatrix} \frac{\partial}{\partial x_1} & \frac{\partial}{\partial x_2} & \dots & \frac{\partial}{\partial x_n} \end{bmatrix} \mathbf{A}^* \begin{bmatrix} \frac{\partial}{\partial x_1} \\ \frac{\partial}{\partial x_2} \\ \vdots \\ \frac{\partial}{\partial x_n} \end{bmatrix}. \tag{2.5}$$

Matrisen $\mathbf{A}^*$, nå uten tidsderiverte er gitt ved:

$$\mathbf{A}^* = \begin{bmatrix} a_1^2 + k_1 & a_1 a_2 & \dots & a_1 a_n \\ a_2 a_1 & a_2^2 + k_2 & \dots & a_2 a_n \\ & & \vdots & \\ a_n a_1 & a_n a_2 & \dots & a_n^2 + k_n \end{bmatrix}.$$

er positiv definit (elliptisk). Dette er synonymt med påstanden:

$$\sum_{i,j=1}^{n} a_{ij} \zeta_i \zeta_j \geq \theta |\zeta|^2. \tag{2.6}$$

for alle $\zeta \in \mathbb{R}^n$.

Hvis vi krever at (2.6) holder for alle punkter i rom og tid kan vi bytte ut a_{ij} med en funksjon $a(x,t)_{ij}$. Leddene av lavere grad, som er sett bort fra til nå, kan legged til og vi har funnet en definisjon for en parabolsk operator:

Definisjon 2.3.1. Operatoren $\frac{\partial}{\partial t} + L$ er parabolsk hvis det eksisterer en konstant $\theta > 0$ slik at:

$$\sum_{i,j=1}^{n} a(x,t)_{ij} \times \zeta_i \zeta_j \geq \theta |\zeta|^2.$$

for alle $\zeta \in \mathbb{R}^n$.
hvor:

$$L = -\sum_{i,j=1}^{n} a(x,t)_{ij} \frac{\partial^2}{\partial x_i \partial x_j} + \sum_{i=1}^{n} b_i(x,t) \frac{\partial}{\partial x_i} + c(x,t)u.$$

Merk at $\mathbf{A}$ fra (2.4) ikke har noen grense for antall egenverdier med verdi 0, og $\mathbf{A_0^*}$ har nøyaktig en egenverdi lik 0. Ved å velge en $a_n = 0$ i $\mathbf{A}$ vil både kolonne og rad nummer n bestå av 0. Vi kan tenke oss at $a_n = 0$ ikke tilføyer noe i (2.4) og lar være å ha den med. Med denne prosessen kan vi redusere $\mathbf{A}$ slik at denne matrisen også har nøyaktig èn egenverdi lik 0. Ved definisjon 2.4.1 kan vi skriver vi en parabolsk differensiallikning som:

$$u_t + Lu = f. \tag{2.7}$$

og kan settes opp:

$$\frac{\partial}{\partial t} - \begin{bmatrix} \frac{\partial}{\partial x_1} u & \frac{\partial}{\partial x_2} u & \dots & \frac{\partial}{\partial x_n} u \end{bmatrix} \mathbf{A}^* \begin{bmatrix} \frac{\partial}{\partial x_1} u \\ \frac{\partial}{\partial x_2} u \\ \vdots \\ \frac{\partial}{\partial x_n} u \end{bmatrix} + \begin{bmatrix} b_1 \frac{\partial}{\partial x_1} u \\ b_2 \frac{\partial}{\partial x_2} u \\ \vdots \\ b_n \frac{\partial}{\partial x_n} u \end{bmatrix} + cu = f \qquad (2.8)$$

Merk spesielt at formen til A^* gir positive parametre tilhørende $\frac{\partial}{\partial x_i \partial x_i}$. Positive parametre tilhørende de andregradsderiverte er i overensstemmelse med varmelikningen (1.2), hvor den tilsvarende parameteren beskriver konduktivitet.

KAPITTEL 3

Maksimumprinsipper for parabolske likninger

I dette kapittelet vil vi se på maksimumprinsipper for parabolske likninger definert i definisjon 2.4.1. Maksimumprinsippene vil fremstilles i svak form. Målet med kapittelet er å få en ide om hvilke begrensninger av (2.7) som bevarer positivitet.
Vi lar $\mathbf{U}$ betegne et åpent, avgrenset område, $\mathbf{U_T} = \mathbf{U} \times (0, T]$.Videre har vi $\Gamma_T = \overline{\mathbf{U_T}} - \mathbf{U_T}$. Det presiseres at vi ser etter positivitetsbevarende systemer, og vi har

$$u(0, x) \geq 0$$

$$u(t, x) \geq 0 \; for \; x \in \Gamma_T$$

Hvis u oppnår en ekstremalverdi i $\mathbf{U_T}$ vil førstegradsleddene ved førstederivert-testen ha verdi 0 i dette punktet. Det vil si at den parabolske likningen 2.7 kan skrives:

$$\frac{\partial}{\partial t}u - \sum_{i,j=1}^{n} a(x,t)_{ij} \frac{\partial^2}{\partial x_i \partial x_j} u + c(x,t)u = f. \tag{3.1}$$

i ekstremalpunktene til u, hvor $\frac{\partial}{\partial t}u = 0$ for alle $t \neq T$.

Matrisefaktorisering

$\mathbf{A}^*$ er positiv definit og symmetrisk, derfor har vi [13, s. 171]:

$$\mathbf{O} \cdot \mathbf{D} \cdot \mathbf{O^T} = \mathbf{A}^*.$$

hvor $\mathbf{O}$ er ortogonal, $\mathbf{D}$ inneholder egenverdiene til $\mathbf{A}^*$ langs diagonalen, og er 0 ellers. For et valgt punkt x_0 har vi:

$$\begin{aligned} \mathbf{x^T} \cdot \mathbf{A}^* \cdot \mathbf{x} = \mathbf{x^T O} \cdot \mathbf{D} \cdot \mathbf{O^T} \cdot \mathbf{x} = (\mathbf{O^T} \cdot \mathbf{x})^T \cdot \mathbf{D} \cdot \mathbf{O^T} \cdot \mathbf{x} \\ = \sum_{k=1}^{n} \lambda_k |\mathbf{O^T x}_k|^2 = \sum_{k=1}^{n} \lambda_k \mathbf{x}_k^2. \end{aligned} \tag{3.2}$$

Med (3.2) kan annengradsleddene i (3.1) skrives:

$$\sum_{i,j=1}^{n} a_{i,j} \frac{\partial^2}{\partial x_i \partial x_j} u = \sum_{k=1}^{n} \lambda_k \frac{\partial u}{\partial x_k \partial x_k}. \tag{3.3}$$

hvor egenverdiene λ_k er positive.
Utnyttes formen (3.3) i analysen av (3.1) blir andrederivert-testen et kraftig verktøy for å identifisere maks- og minimumsverdier.

3.1 Reaksjonsledd lik null

f mindre enn, eller lik 0

Valg av *reaksjonsledd* $c = 0$, og en negativ funksjon f, sammen med omskrivingen (3.3) transformeres (3.1) til:

$$\frac{\partial}{\partial t} u - \sum_{k=1}^{n} \lambda_k \frac{\partial^2}{\partial x_k \partial x_k} u = f \leq 0. \tag{3.4}$$

Vi antar at u har et maksimum i $\mathbf{U}_T$ for $t \neq T$. Andrederivert-testen forteller at et maksimum i $\mathbf{U_T}$ har $\frac{\partial^2}{\partial x_k \partial x_k} u \leq 0$, og førstederivert-testen gir $\frac{\partial}{\partial t} u = 0$. Ulikheten (3.4) og antakelsen om et maksimum gir ulikheten:

$$0 \leq -\sum_{k=1}^{n} \lambda_k \frac{du}{dx_k dx_k} = f \leq 0. \tag{3.5}$$

For å få streng motsigelse innføres en funksjon $u^\epsilon = u + \epsilon t$:

$$u_t^\epsilon + Lu^\epsilon = u_t + Lu + \epsilon.$$

$$0 < -\sum_{k=1}^{n} \lambda_k \frac{\partial^2}{\partial x_k \partial x_k} u + \epsilon = f \leq 0.$$

Dette er en streng motsigelse, og $\epsilon \to 0$ gir $u^\epsilon \to u$. Det gjenstår å teste for sluttiden T. Hvis u har et toppunkt i t=T er $\frac{\partial}{\partial t} u \geq 0$ og motsigelsen i (3.5) holder fortsatt. Vi har dermed maksimumprinsippet:

$$\max_{\overline{U_T}} u = \max_{\Gamma_T} u. \tag{3.6}$$

f større enn, eller lik 0

Igjen anvendes andrederivert-testen, denne gangen for er minimum. Hvis u oppnår et minimum i $\mathbf{U_T}$ er $\frac{\partial^2}{\partial x_k \partial x_k} u \geq 0$ og $\frac{\partial}{\partial t} u \leq 0$. Vi innfører en $u^\epsilon = u - \epsilon t$ slik at:

$$u_t^\epsilon + Lu^\epsilon = u_t + Lu - \epsilon.$$

som fører til motsigelsen:

$$0 > -\sum_{k=1}^{n} \lambda_k \frac{\partial^2}{\partial x_k \partial x_k} u + \frac{\partial}{\partial t} u - \epsilon = f \geq 0.$$

$\epsilon \to 0$ gir $u^\epsilon \to u$ og maksimumprinsippet:

$$\min_{\overline{U_T}} u = \min_{\Gamma_T} u. \tag{3.7}$$

Avhengig av fortegnet til funksjonen f har u en øvre, eller nedre grense definert av rand- og initialbetingelsene. Merk at for det homogene problemet vil både (3.6) og (3.7) gjelde.

3.2 Positivt reaksjonsledd

f mindre enn, eller lik 0

Vi antar at u har et maksimum i $\mathbf{U_T}$, og bruker $u^\epsilon = u + \epsilon t$. For et maksimum er $\frac{\partial^2}{\partial x_k \partial x_k} u \geq 0$ og $\frac{\partial}{\partial t} u \geq 0$, og vi har ulikheten:

$$0 < -\sum_{k=1}^{n} \lambda_k \frac{\partial^2}{\partial x_k \partial x_k} + \frac{\partial}{\partial t} u + c(x,t)u + \epsilon = f \leq 0.$$

Dette er en motsigelse, og (3.6) gjelder for valg av $c > 0$, $f \leq 0$

f større enn, eller lik 0

Hvis u er et maksimum får vi:

$$0 \leq -\sum_{k=1}^{n} \lambda_k \frac{\partial^2}{\partial x_k \partial x_k} u + \frac{\partial}{\partial t} u + c(x,t)u = f \geq 0.$$

som ikke gir motsigelse. Hvis u har et minimum i $\mathbf{U_T}$ er $\frac{\partial^2}{\partial x_k \partial x_k} u \geq 0$ og $\frac{\partial}{\partial t} u \leq 0$, mens $c(x,t)u$ gir positive verdier. Hvis vi antar at verdien til reaksjonsleddet er mindre enn, eller lik verdien til første- og annengradsleddet til sammen har vi:

$$\sum_{k=1}^{n} \lambda_k \frac{\partial^2}{\partial x_k \partial x_k} u - \frac{\partial}{\partial t} u \geq c(x,t)u. \tag{3.8}$$

(3.8), med en funksjon $u^\epsilon = u - \epsilon t$ fører til motsigelsen:

$$0 > -\sum_{k=1}^{n} \lambda_k \frac{\partial^2}{\partial x_k \partial x_k} u + \frac{\partial}{\partial t} u + c(x,t)u - \epsilon = f \geq 0.$$

(3.7) gjelder for $f \geq 0$, $c > 0$, men bare hvis (3.8) er oppfylt.

3.3 Negativt reaksjonsledd

f mindre enn, eller lik 0

Likt fortegn for f og c krever tillegskrav for å produsere motsigelser. Antar vi at u har et maksimum i $\mathbf{U_T}$ har vi $\frac{\partial^2}{\partial x_k \partial x_k} u \leq 0$ og $\frac{\partial}{\partial t} u \geq 0$. Vi oppretter kravet:

$$\sum_{k=1}^{n} \lambda_k \frac{\partial^2}{\partial x_k \partial x_k} u - \frac{\partial}{\partial t} u \leq c(x,t)u. \tag{3.9}$$

(3.9) sammen med en funksjon $u^\epsilon = u + \epsilon t$ gir:

$$0 < -\sum_{k=1}^{n} \lambda_k \frac{\partial^2}{\partial x_k \partial x_k} u + \frac{\partial}{\partial t} u + c(x,t)u + \epsilon = f \leq 0.$$

Maksimumprinsippet (3.6) gjelder for $f \leq 0$, $c < 0$, men bare hvis (3.9) holder.

f større enn, eller lik 0

Vi antar at u har et minimum i $\mathbf{U_T}$, som gir $\frac{\partial^2}{\partial x_k \partial x_k} u \geq 0$,og $\frac{\partial}{\partial t} u \leq 0$. Funksjonen $u^\epsilon = u - \epsilon t$ gir motsigelsen:

$$0 > -\sum_{k=1}^{n} \lambda_k \frac{\partial^2}{\partial x_k \partial x_k} u + \frac{\partial}{\partial t} u + c(x,t)u = f \geq 0.$$

Maksimumprinsippet (3.6) gjelder for $f \geq 0$, $c < 0$

For positivitetsbevaring vil kombinasjoner av f og c som oppfyller (3.7) være spesielt interessante, da denne garanterer at løsningen aldri kan være lavere enn rand- og initialbetingelsene.

KAPITTEL 4

Numeriske skjema i to romdimensjoner

I dette kapittelet skal vi se på fire numeriske skjema for varmelikningen med konveksjon. Problemet vi skal se på er:

$$\begin{aligned} u_t - a_1 u_{xx} - a_2 u_{yy} + b_1 u_x + b_2 u_y = f(x,y,t) \quad , \quad (x,y,t) \in \mathbf{U_T} \\ u = g_1(x,y,t) \qquad \text{på} \quad \boldsymbol{\partial}\mathbf{U} \times (0,T] \\ u = g_2(x,y) \qquad \text{på} \qquad \mathbf{U} \times \{t=0\}. \end{aligned} \tag{4.1}$$

Skrivemåten for de partiellderiverte er forkortet ved å la:

$$u_t = \frac{\partial}{\partial t} u.$$

$$u_{xx} = \frac{\partial^2}{\partial x \partial x} u.$$

og tilsvarende for de resterende partiellderiverte. Området $\mathbf{U_T}$ er som beskrevet i kapittel 3, og $\partial\mathbf{U}$ er gitt som

$$\partial\mathbf{U} = \overline{\mathbf{U}} - \mathbf{U}.$$

funksjonen g_1 vil omtales som *randbetingelsene*, og g_2, som *initialbetingelsene*, eller *starttilstanden*. Problemet er åpent for diffusjon og konveksjon i x- og y-retning. Det inomogene leddet gir muligheten for tap eller tilføring av energi fra omgivelsene. Merk at det ikke er med et ledd cu, så vi antar at tap/tilførsel av energi ikke skjer ved kjemiske reaksjoner. a_1, a_2 er funksjoner med verdier > 0 , mens b_1, b_2 er funksjoner som kan ha både positive og negative verdier. I store deler av kapittelet vil a_1, a_2, b_1, b_2 være valgt som konstanter. Diskusjon om betingelser for glatthet til de valgte funksjonene er utelatt. Jeg krever at funksjonene er valgt « glatte nok » for alle skjemaene. Det vil si minst: $g_1(x,y,t), g_2(x,y), f(x,y,t) \in C^1$, og løsningen $u \in C^2$ i $\mathbf{U_T}$.

For alle numeriske metoder trenger vi sett av endelig mange punkter/noder. Alle skjemaene i dette kapittelet har et kvadratisk grid:

$$\Delta x = \Delta y = h \quad , \quad h > 0.$$

Området som undersøkes er et rektangulært snitt med bredde b og høyde s, dette gir $N_1 N_2$ punkter i rom hvor $N_1 = \frac{b}{h} + 1$ og $N_2 = \frac{s}{h} + 1$. Tiden deles inn i M punkter hvor $M = \frac{T}{\Delta t} + 1$. Skrivemåten for funksjonsverdien i en node i, j, m er:

$$u_{i,j}^m = u(x_i, y_j, t_m). \tag{4.2}$$

$$x_i = x_1 + \frac{i-1}{N_1 - 1} b \quad , \quad y_j = y_1 + \frac{j-1}{N_2 - 1} s \quad , \quad t_m = \frac{m}{M} \Delta t$$

Indeksene er heltall $i \in [1, N_1], j \in [1, N_2], m \in [0, M]$. Vi lar (x_1, y_1) være nederste venstre hjørne i rektangelet.

4.1 Forlengs Euler

Eulers metode, oppkalt etter Leonhard Euler som introduserte metoden i *Institutiones calculi integrals volumen Primum*[15]. Eulers metode er en populær første metode for numersk løsning av differensiallikninger, grunnet sin simplisitet. En eksplisitt versjon av Eulers metode er valgt som en myk introduksjon, og en standard metode andre skjema kan måles mot.

Oppsett

Forlengs Euler modellerer (4.1) ved hjelp av endelige differanser. Parametrene a_1, a_2, b_1, b_2 vil bli behandlet som konstanter. Brukes definisjonen til den deriverte kan vi sette opp:

$$\frac{\partial}{\partial t} u(x_i, y_j, t_m) = \frac{u(x, y, t + \Delta t) - u(x, y, t)}{\Delta t} + O(\Delta t). \tag{4.3}$$

$$\frac{\partial}{\partial x} u(x_i, y_j, t_m) = \frac{u(x + \Delta x, y, t) - u(x, y, t)}{\Delta t} + O(\Delta x). \tag{4.4}$$

$$\frac{\partial}{\partial y} u(x_i, y_j, t_m) = \frac{u(x, y + \Delta y, t) - u(x, y, t)}{\Delta y} + O(\Delta y). \tag{4.5}$$

$$\frac{\partial^2}{dx^2} u(x_i, y_j, t_m) = \frac{u(x - \Delta x, y, t) - 2u(x, y, t) + u(x + \Delta x, y, t)}{\Delta x^2} + O(\Delta x^2). \tag{4.6}$$

$$\frac{\partial^2}{dy^2} u(x_i, y_j, t_m) = \frac{u(x, y - \Delta y, t) - 2u(x, y, t) + u(x, y + \Delta y, t)}{\Delta y^2} + O(\Delta y^2). \tag{4.7}$$

Overskuddsfunksjonene kommer av usikkerheten forbundet ved å ikke la endringen go mot 0. Brukes (4.2) kan vi skrive (4.3)-(4.7) på mer kompakt form:

$$\frac{\partial}{\partial t} u(x_i, y_j, t_m) \approx \frac{u_{i,j}^{m+1} - u_{i,j}^m}{\Delta t}. \tag{4.8}$$

$$\frac{\partial}{\partial x} u(x_i, y_j, t_m) \approx \frac{u_{i+1,j}^m - u_{i,j}^m}{h}. \tag{4.9}$$

$$\frac{\partial}{\partial y}u(x_i, y_j, t_m) \approx \frac{u^m_{i,j+1} - u^m_{i,j}}{h}. \tag{4.10}$$

$$\frac{\partial^2}{\partial x^2}u(x_i, y_j, t_m) \approx \frac{u^m_{i-1,j} - 2u^m_{i,j} + u^m_{i+1,j}}{h^2}. \tag{4.11}$$

$$\frac{\partial^2}{\partial y^2}u(x_i, y_j, t_m) \approx \frac{u^m_{i,j-1} - 2u^m_{i,j} + u^m_{i,j+1}}{h^2}. \tag{4.12}$$

I et eksplisitt skjema regner vi direkte på verdiene fremover i tid. Det vil si at vi med (4.8)-(4.12) setter opp et numerisk skjema for (4.1) som:

$$\begin{aligned}\frac{u^{m+1}_{i,j} - u^m_{i,j}}{\Delta t} = &\frac{a_1}{h^2}(u^m_{i-1,j} - 2u^m_{i,j} + u^m_{i+1,j}) + \frac{a_2}{h^2}(u^m_{i,j-1} - 2u^m_{i,j} + u^m_{i,j+1}) \\ &- \frac{b_1}{h}(u^m_{i+1,j} - u^m_{i,j}) - \frac{b_2}{h}(u^m_{i,j+1} - u^m_{i,j}) + f(x_i, y_j, t_m)).\end{aligned} \tag{4.13}$$

Et numerisk skjema av denne typen kaller vi *forlengs Euler*. Opprydning av (4.13) til kjente verdier på høyreside, og de ukjente på venstre side gir:

$$\begin{aligned}u^{m+1}_{i,j} = &u^m_{i-1,j}(\frac{\Delta t}{h^2}a_1) + u^m_{i,j-1}(\frac{\Delta t}{h^2}a_2) + u^m_{i,j}(1 + \frac{\Delta t}{h}(-\frac{2a_1}{h} - \frac{2a_2}{h} + b_1 + b_2)) \\ &+ u^m_{i+1,j}\frac{\Delta t}{h}(\frac{a_1}{h} - b_1) + u^m_{i,j+1}\frac{\Delta t}{h}(\frac{a_2}{h} - b_2) + \Delta t f(x_i, y_j, t_m).\end{aligned} \tag{4.14}$$

Positivitet og stabilitet

For å sikre at (4.14) gir en god approksimasjon av (4.1) settes krav til h og Δt. En analog til maksimumprinsipper utviklet i kapittel 3 er positivitetsbevaring. Hvis vi kan vise at løsningen tilfredsstiller:

$$u^m_{i,j} \geq 0$$

for homogene randbetingelser har vi også vist at et vilkårlig problem (4.1) tilfredsstiller en diskret versjon av (3.7):

$$u^m_{i,j} \geq min\{u^0_{i,j}, u^m_{0,j}, u^m_{N_1 h,j}, u^m_{i,0}, u^m_{i,N_2 h}.\} \tag{4.15}$$

Grunnen til at positivitet fører til et maksimumprinsipp er fordi alle problemer av form (4.1) kan løses ved å innføre en $w = u - v$, som beskrevet i kapittel senere i (4.35). Hvis w, som har homogene randabetingelser bevarer positivitet vil det originale problemet u som konsekvens oppfylle den diskrete versjonen av (3.7).
Positivitet sikres ved å se på høyresiden av (4.14) ledd for ledd. Uttrykkene knyttet til $u^m_{i+1,j}$ og $u^m_{i,j+1}$ er positive hvis:

$$h \leq min\{\frac{a_1}{|b_1|}, \frac{a_2}{|b_2|}\}. \tag{4.16}$$

For leddet $u^m_{i,j}$ har vi:

$$1 + \frac{\Delta t}{h}(-\frac{2a_1 + 2a_2}{h} + b_1 + b_2) \geq 0.$$

som tilfredsstilles for:

$$\Delta t \leq \frac{h^2}{2(a_1+a_2)+h(|b_1+b_2|)}. \tag{4.17}$$

Tilfredsstilles (4.16) og (4.17) vil alle ledd på høyresiden i (4.14) være positive, og det gjenstår å betrakte f. Fra kapittel 3 vet vi at $f \geq 0$ skal bevare positivitet, og ved å se $f \geq 0 \rightarrow f\Delta t \geq 0$ stemmer dette også for (4.14). Krav som bevarer positivitet for negative verdier av f er også mulige med noen begrensninger. Ved å velge $\mathbf{m} = min(u^m_{x_{i\pm1},\mathbf{y}_{j\pm1}})$ og sette dette inn i (4.13) har vi:

$$u^{m+1}_{i,j} \geq \mathbf{m}(1-\Delta t(\frac{2a_1+2a_2-a_1-a_2-a_1-a_2}{h^2}+\frac{b_1+b_2-b_1-b_2}{h}))+\Delta t f(x_i,y_j,t_m).$$

som forenkles til:

$$\mathbf{m}+\Delta t f(x_i,y_j,t_m) \geq 0.$$

For negativ f blir positivitet bevart ved:

$$\Delta t \leq \frac{\mathbf{m}}{|min\{f\}|}. \tag{4.18}$$

(4.18) er et krav som bevarer positivitet i begrenset tid. Svakheten ved (4.18) er at $\mathbf{m}$ kan bli, eller allerede er veldig liten, som fører til tidssteg som konvergerer mot 0. (4.1) er på grunn av forholdet mellom m og f ikke positivitetsbevarende for alle $g_2 \geq 0$. Videre vil vi forholde oss til $f \geq 0$

Kravene gitt i (4.16) og (4.17) gir positivitet i (4.14). En velkommen bonus ved de samme kravene er at skjemaet er stabilt. Brukes lineariteten i (4.1) som argument kan vi sette opp et uttrykk for feilen i det diskrete skjemaet ved hjelp av formen til analytiske løsninger. Kreves forhold mellom feilen i to etterfølgende tidssteg ≤ 1 vil feilledd dempes med tiden, og vi sier at skjemaet er von neumann stabilt ([16, Kapittel 4.3]):

$$\begin{aligned} E(\Delta t(m+1)e^{z(ki+lj)} = E(\Delta tm)(&e^{z(k(i-1)+lj)}(\frac{\Delta t}{h^2}a_1)+e^{z(ki+l(j-1))}(\frac{\Delta t}{h^2}a_2) \\ &+e^{z(ki+lj)}(1+\frac{\Delta t}{h}(-\frac{2a_1}{h}-\frac{2a_2}{h}+b_1+b_2)) \\ &+e^{z(k(i+1)+lj)}\frac{\Delta t}{h}(\frac{a_1}{h}-b_1)+e^{z(ki+l(j+1))}\frac{\Delta t}{h}(\frac{a_2}{h}-b_2)). \end{aligned} \tag{4.19}$$

I (4.19) er $z = \sqrt{-1}\pi hr$, med r en positiv konstant. For stabilitetsanalyse kan vi velge $r = 1$ uten tap av generalitet. $k, l > 0$ er heltall og ved å endre på disse får vi flere løsninger som tilfredsstiller (4.1) uten rand- og initialbetingelser. Ved en superposisjon av de generelle løsningene kan vi bygge opp den spesifikke løsningen som tilfredsstiller g_1 og g_2 (se gjerne [11] for behandling av 1D diffusjon-konveksjon). Deles alle ledd på $E(\Delta tm)e^{z(ki+lj)}$ har vi:

$$\begin{aligned} \frac{E(\Delta t(m+1)}{E(\Delta t(m)} = e^{-zk}(\frac{\Delta t}{h^2}a_1)+e^{-zl}(\frac{\Delta t}{h^2}a_2)+(1+\frac{\Delta t}{h}(-\frac{2a_1}{h}-\frac{2a_2}{h}+b_1+b_2)) \\ +e^{zk}\frac{\Delta t}{h}(\frac{a_1}{h}-b_1)+e^{zl}\frac{\Delta t}{h}(\frac{a_2}{h}-b_2). \end{aligned} \tag{4.20}$$

Ved å anvende kjente trigonometriske identiteter kan vi rydde opp til:

$$\frac{E(\Delta t(m+1)}{E(\Delta t(m)} = 1 - \frac{a_1 \Delta t}{h^2} 4sin^2(k\pi\frac{h}{2}) - \frac{a_2 \Delta t}{h^2} 4sin^2(l\pi\frac{h}{2}) + \frac{\Delta t}{h}((b_1(1-e^{zk}) + b_2(1-e^{zl})).$$

Vi vet at $0 \leq sin^2(zk\frac{h}{2}) \leq 1$, og $0 \leq |1 - e^{zkh}| \leq 2$, og bruker dette til å se om forholdet mellom feilen i to etterfølgene tidssteg er mindre enn 1.I det første ekstremtilfellet velges $sin^2(k\pi\frac{h}{2}) = sin^2(l\pi\frac{h}{2}) = 0$ og vi får:

$$|\frac{E(\Delta t(m+1)}{E(\Delta t(m)}| = 1 \leq 1.$$

I det andre ekstremtilfellet har vi $sin^2(k\pi\frac{h}{2}) = sin^2(l\pi\frac{h}{2}) = 1$ som gir:

$$|\frac{E(\Delta t(m+1)}{E(\Delta t(m)}| = |1 - \frac{\Delta t}{h^2}(4a_1 + 4a_2 - 2hb_1 - 2hb_2)| \leq 1.$$

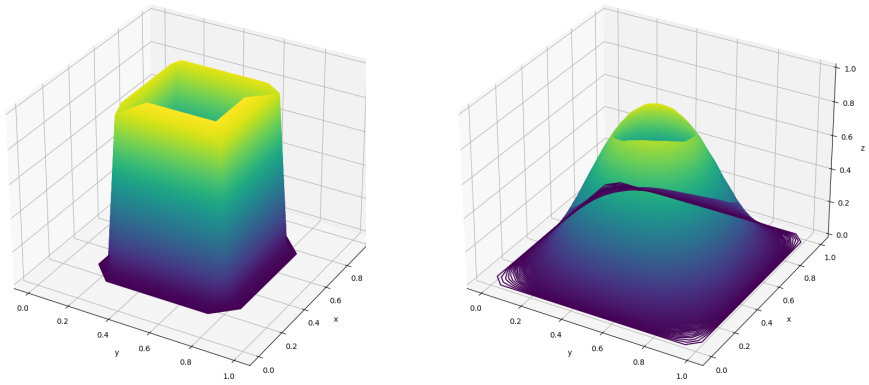
Brukes kravet (4.16) har vi:

$$(4a_1 + 4a_2 - 2hb_1 - 2hb_2) \geq 0.$$

og von neumann stabilitet er oppnådd for:

$$\frac{\Delta t}{h^2}(4a_1 + 4a_2 - 2hb_1 - 2hb_2) \leq 2.$$

Dette er identisk med (4.17) og beviset er fullført. Under er det vist et kjøreeksempel for implementeringen av (4.14)B.1. Eksempelet bruker samme initialbetingelser som vi vil se senere i kapittel 4.4:



(a) Kuboide etter ett tidssteg (b) Kuboide etter litt tid

Figur 4.1: Stresstest av skjema med endelige-differanser h=1/20 og Δt=1/10000

Fordelen med et skjema som forlengs Euler er lav kompleksitet. Ulempen er at stabilitetskravene er strengere enn ønskelig. Konvergens mot en analytisk løsning er også tregere enn andre skjemaer diskutert i denne avhandlingen (se kapittel 5.3). Det koster derfor mye prosessorkraft å produsere nøyaktige løsninger med forlengs Euler. Spesielt vil det være strenge krav til tidsstegene. Det første forsøket på et mer nyttig skjema er et Crank-Nicolson skjema:

4.2 Crank-Nicolson

Crank-Nicolson(CN) er en optimering av Eulers metode, som bruker en kombinasjon av forlengs-, og baklengs Euler. Metoden ble utviklet av John Crank og Phyllis Nicolson på midten av 1900-tallet [3].

Oppsett

I forlengs Euler bruker vi verdier fra nåværende tid for å regne ut partiellderiverte. I et Crank-Nicolson type skjema finner vi approksimasjoner for de partiellderiverte ved å se på gjennomsnittet av nåværende tidssteg og det neste. Vi sier da at skjemaet er sentrert i tiden $\Delta t(m+\frac{1}{2})$ for tidssteg m, og stedet $((i-1)h,(j-1)h)$ for i,j. De andrederiverte er allerede sentrert i (x_i,y_j), og de førstederiverte til x og y sentreres ved:

$$\frac{\partial}{\partial x}u(x_i,y_j,t_m) \approx \frac{1}{2}\big(\frac{u^m_{i+1,j}-u^m_{i,j}}{h}+\frac{u^m_{i,j}-u^m_{i-1,j}}{h}\big). \tag{4.21}$$

(4.21) forkortes i skjemaet, men vi utnytte den fulle formen under stabilitetsanalysen. Satt sammen har vi:

$$\frac{\partial}{\partial t}u(x_i,y_j,t_{m+\frac{1}{2}}) \approx \frac{u^{m+1}_{i,j}-u^m_{i,j}}{\Delta t}. \tag{4.22}$$

$$\frac{\partial}{\partial x}u(x_i,y_j,t_{m+\frac{1}{2}}) \approx \frac{1}{2}\big(\frac{u^m_{i+1,j}-u^m_{i-1,j}}{2h}+\frac{u^{m+1}_{i+1,j}-u^{m+1}_{i-1,j}}{2h}\big). \tag{4.23}$$

$$\frac{\partial}{\partial y}u(x_i,y_j,t_{m+\frac{1}{2}}) \approx \frac{1}{2}\big(\frac{u^m_{i,j+1}-u^m_{i,j-1}}{2h}+\frac{u^{m+1}_{i,j+1}-u^{m+1}_{i,j-1}}{2h}\big). \tag{4.24}$$

$$\frac{\partial^2}{\partial x^2}u(x_i,y_j,t_{m+\frac{1}{2}}) \approx \frac{1}{2}\big(\frac{u^m_{i-1,j}-2u^m_{i,j}+u^m_{i+1,j}}{h^2}+\frac{u^{m+1}_{i-1,j}-2u^{m+1}_{i,j}+u^{m+1}_{i+1,j}}{h^2}\big). \tag{4.25}$$

$$\frac{\partial^2}{\partial y^2}u(x_i,y_j,t_{m+\frac{1}{2}}) \approx \frac{1}{2}\big(\frac{u^m_{i,j-1}-2u^m_{i,j}+u^m_{i,j+1}}{h^2}+\frac{u^{m+1}_{i,j-1}-2u^{m+1}_{i,j}+u^{m+1}_{i,j+1}}{h^2}\big). \tag{4.26}$$

Fordelen ved å sette opp skjemaet med (4.22)-(4.26) er at skjemaet vil være grad 2 nøyaktig. Et formelt bevis for nøyaktigheten blir ikke ført, men jeg vil gi en fungerende praktisk forklaring: I forlengs Euler brukes partiellderiverte for x

og y i tiden m til å finne endringen fra m til $m+1$. Dette er bare eksakt for lineær tidsutvikling. Ved å bruke verdier i $m+\frac{1}{2}$ for å finne endringen fra m til $m+1$ har vi eksakte verdier for utvikling opp til et annengradspolynom. Ved å la polynomgraden som kan tilnærmes eksakt betegne graden av nøyaktighet har vi derfor første grad nøyaktighet for forover Euler, og grad to nøyaktighet for Crank-Nicolson. Dette betyr ikke at skjemaet må være mer nøyaktig for alle valg av h og Δt, men at feilen vil konvergere fortere ved å reduserer h og Δt. For oversiktlig implementering av C-N for en standard varmelikning og utledning av nøyaktighet, se [4].
Skjemaet for (4.1) satt opp med (4.22)-(4.26) ser slik ut:

$$\begin{aligned} u_{i,j}^{m+1}(1+\frac{\Delta ta_1}{h^2}+\frac{\Delta ta_2}{h^2}) - u_{i-1,j}^{m+1}(\frac{\Delta ta_1}{2h^2}+\frac{\Delta tb_1}{4h}) - u_{i+1,j}^{m+1}(\frac{\Delta ta_1}{2h^2}-\frac{\Delta tb_1}{4h}) \\ - u_{i,j-1}^{m+1}(\frac{\Delta ta_2}{2h^2}+\frac{\Delta tb_2}{4h}) - u_{i,j-1}^{m+1}(\frac{\Delta ta_2}{2h^2}-\frac{\Delta tb_2}{4h}) = \\ u_{i,j}^{m}(1-\frac{\Delta ta_1}{h^2}-\frac{\Delta ta_2}{h^2}) + u_{i-1,j}^{m}(\frac{\Delta ta_1}{2h^2}+\frac{\Delta tb_1}{4h}) + u_{i+1,j}^{m}(\frac{\Delta ta_1}{2h^2}-\frac{\Delta tb_1}{4h}) \\ + u_{i,j-1}^{m}(\frac{\Delta ta_2}{2h^2}+\frac{\Delta tb_2}{4h}) + u_{i,j+1}^{m}(\frac{\Delta ta_2}{2h^2}-\frac{\Delta tb_2}{4h}) + \Delta t(f(x_i,y_j,t_{m+\frac{1}{2}})). \end{aligned} \tag{4.27}$$

(4.27) inneholder flere ukjente enn vi kan løse for eksplisitt, og vi må bruke en implisitt metode. For å løse systemet må vi utnytte verdier på $\partial\mathbf{U}$ hvor vi kjenner verdiene. Om vi starter i nedre venstre hjørne av et rektangelært står vi igjen med tre ukjente. Vi har to akser og kan ved å sette inn fra forrige tidssteg langs både x- og y-aksen over området sette opp et løselig likningssystem. Før vi går videre med løsningen kan vi observere at høyresiden av (4.27) vil være positiv ved å oppfylle

$$h \leq min\{\frac{2a_1}{|b_1|},\frac{2a_2}{|b_2|}\}. \tag{4.28}$$

og

$$\Delta t \leq \frac{h^2}{a_1+a_2}. \tag{4.29}$$

Kravene (4.28) og (4.29) fører til at alle ledd untatt det første på venstreside i (4.27) er negative. Kreves summen av parametre på venstresiden større enn 0:

$$1+\frac{\Delta ta_1}{h^2}+\frac{\Delta ta_2}{h^2}-\frac{\Delta ta_1}{2h^2}-\frac{\Delta tb_1}{4h}-\frac{\Delta ta_1}{2h^2}+\frac{\Delta tb_1}{4h}-\frac{\Delta ta_2}{2h^2}-\frac{\Delta tb_2}{4h}-\frac{\Delta ta_2}{2h^2}+\frac{\Delta tb_2}{4h} \geq 0.$$

som gir:

$$1 \geq 0$$

har vi allerede bevist at systemet bevarer positivitet. Dette fordi matriseformen til (4.27) kan settes opp som:

$$(\mathbf{I}+\mathbf{S})\cdot\mathbf{u^{m+1}} = (\mathbf{I}-\mathbf{S})\cdot\mathbf{u^m} + \Delta t\mathbf{F} = \mathbf{b}.$$

hvor $\mathbf{b}$ er en vektor med positive verdier, og $\mathbf{I}+\mathbf{S}$ er en M-matrise (se plemmons [20, s. 175–185]). CN- skjemaet oppfyller ved positivitet maksimumprinsippet (4.15), på lik linje med forlengs Euler. Definerer videre:

$$\frac{\Delta t(2a_1+hb_1)}{4h^2} = \sigma_1.$$

$$\frac{\Delta t(2a_1 - hb_1)}{4h^2} = \sigma_2.$$
$$\frac{\Delta t(2a_2 + hb_2)}{4h^2} = \omega_1.$$
$$\frac{\Delta t(2a_2 - hb_2)}{4h^2} = \omega_2.$$
$$\frac{\Delta t(a_1 + a_1)}{h^2} = \alpha.$$

og $\nu_n \geq 0$ som relaterer to etterfølgende kjente ledd i tid på $\partial\mathbf{U}$, som gir:

$$\mathbf{S} = \begin{bmatrix} \nu_1 & 0 & \cdots & & & & & & 0 \\ 0 & \nu_2 & \cdots & & & & & & 0 \\ \vdots & & & & & & & & \\ 0 & -\omega_1 & \cdots & -\sigma_1 & \alpha & -\sigma_2 & \cdots & -\omega_2 & \cdots \\ \cdots & -\omega_1 & \cdots & -\sigma_1 & \alpha & -\sigma_2 & \cdots & -\omega_2 & \cdots \\ \vdots & & & & & & & & \end{bmatrix}.$$

Matrisen $\mathbf{S}$ har størrelse $N_1N_2 \times N_1N_2$ og er symmetrisk. Siden randbetingelser allerede er gitt kan vi velge parametrene $\nu_n = 0$ og manuelt sette inn verdiene for neste tidssteg på randen. Radene med ν oppstår ved de $N_1 + 1$ første og siste radene i $\mathbf{S}$ i tilligg til to rader med mellomrom på $N_1 - 2$. Det er brukt $\cdots$ hvor det er mer enn et element med verdi 0. Mellom $-\omega_1$ til σ_1, og σ_2 til $-\omega_2$ er det et mellomrom på $N_1 - 2$ elementer.
Programmet som løser (4.27) er lagt ved som vedlegg B.2. Det gjenstår å bekrefte stabiliteten til (4.27).

Stabilitet

På samme måte som for forlengs Euler viser vi von Neumann stabilitet ved å kreve demping av feilledd over tid:

$$\begin{aligned} E(\Delta t(m+1))\big((1+\alpha+\frac{h(b_1+b_2)}{4h^2}-\frac{h(b_1+b_2)}{4h^2})e^{z(ki+lj)}-\sigma_1 e^{z(k(i-1)+lj)}-\sigma_2 e^{z(k(i+1)+lj)} \\ - \omega_1 e^{z(ki+l(j-1))} - \omega_2 e^{z(ki+l(j+1))}\big) \\ = E(\Delta tm)\big((1-\alpha+\frac{h(b_1+b_2)}{4h^2}-\frac{h(b_1+b_2)}{4h^2}))e^{z(ki+lj)} + \sigma_1 e^{z(k(i-1)+lj)} \\ + \sigma_2 e^{z(k(i+1)+lj)} + \omega_1 e^{z(ki+l(j-1))} + \omega_2 e^{z(ki+l(j+1))}\big). \end{aligned} \quad (4.30)$$

Stabilitetsanalysen bruker formen (4.21) for de førstederiverte for x og y, som fører til et uttrykk som er lettere å analysere. Deler begge sider på $E(\Delta tm)e^{z(ki+lj)}$, sorterer etter ledd med like parametre, anvender trigonometriske identiteter som i forrige delkapittel og får:

$$\frac{E(\Delta t(m+1))}{E(\Delta tm)} = \frac{1 - \frac{2\Delta t}{h^2}\big(a_1 sin^2(k\pi\frac{h}{4}) + a_2 sin^2(l\pi\frac{h}{2}) + \frac{hb_1}{4} sin^2(k\pi\frac{h}{2}) - \frac{hb_2}{2} sin^2(l\pi\frac{h}{2})\big)}{1 + \frac{2\Delta t}{h^2}\big(a_1 sin^2(k\pi\frac{h}{2}) + a_2 sin^2(l\pi\frac{h}{2}) - \frac{hb_1}{4} sin^2(k\pi\frac{h}{2}) + \frac{hb_2}{4} sin^2(l\pi\frac{h}{2})\big)}. \quad (4.31)$$

Kravet (4.28) gir positive uttrykk i parantesene til (4.31), og forholdet $\frac{E(\Delta t(m+1))}{E(\Delta tm)} \leq 1$ er oppfylt. Kravet for Δt gitt i (4.29), som allerede er snillere enn (4.18) kan derfor bli sett på som veiledende. Crank-Nicolson skjemaet (4.27) krever flere utregninger for hvert tidssteg enn forlengs Euler, men vi oppnår stabile løsninger tidligere, som kan gjøre crank-Nicolson ønskelig i situasjoner hvor vi modellerer større tidsperioder. I kapittel 5.3 vil vi se at CN konvergerer raskt mot den analytiske løsningen-igjen i forhold til forlengs Euler.

4.3 Endelige elementer, delvis diskretisert

Metoden med endelige elementer fungerer slik navnet tilsier-ved å dele et stort problem i mange mindre elementer. Elementene blir bestemt med noder som binder dem sammen. Inne i elementene blir det opprettet funksjoner som vil interpolere den faktiske løsningen. I vårt tilfelle vil dette være bilineære funksjoner. For å finne en interpolasjon vil vi gjerne unngå å jobbe med andrederiverte, som krever omskriving av problemet til svak form. Delmålene er dermed:

1. Skriv om problem til svak form.
2. Diskretiser problemet.
3. Sett opp og løs lineært likningssystem.

På engelsk heter metoden *finite element method*, som ofte forkortes til FEM.

Svak form

Den svake formen skriver om (4.1) til integralform. Dette oppnås ved å multiplisere inn en en testfunksjon φ, og integrere over området. Skjemaet vi skal bygge bruker endelige elementer i romdimensjonene og endelige differanser i tid, så området vi integrerer over er det 2-dimensjonale rektangelet i rom.(4.1) på svak form blir:

$$\int_0^s \int_0^b (u_t - a_1 u_{xx} - a_2 u_{yy} + b_1 u_x + b_2 u_y)\varphi \; dxdy = \int_0^s \int_0^b f(x,y,t)\varphi \; dxdy.$$

hvor de andrederiverte kan skrives om ved delvis integrasjon:

$$-\int_0^s \int_0^b a_1 u_{xx}\varphi dxdy = -\int_0^s [u_x a_1 \varphi]_{x=0}^{x=b} + \int_0^s \int_0^b u_x(\varphi_x a_1 + \varphi(a_1)_x)dxdy. \tag{4.32}$$

Vi innfører en ny notasjon for integralet av produktet til to funksjoner:

$$\int a(x)b(x)dx = (a,b). \tag{4.33}$$

Med (4.33) kan vi skrive den svake formen til (4.1):

$$(u_t, \varphi) + (a_1 u_x, \varphi_x) + ((a_1)_x u_x, \varphi) + (a_2 u_y, \varphi_y) + ((a_2)_y u_y, \varphi)$$

$$+ (b_1 u_x, \varphi) + (b_2 u_y, \varphi) = \int_0^s [a_1 u_x \varphi]_{x=0}^{x=b} + \int_0^b [a_2 u_y \varphi]_{y=0}^{y=s} + (f, \varphi). \quad (4.34)$$

Vi må ta et valg om behandling av randverdiene. Det er mulig å løse (4.34) for

$$w_{i,j}^m = ui, j^m - vi, j \quad (4.35)$$

hvor w har homogene randbetingelsene. Ved å velge v som løsningen til en stasjonær variant av (4.1) med like randbetingelser vil denne være konstant i en tidsutvikling. Løsningen på det originale problemet finner vi da ved $ui, j^{m+1} = w_{i,j}^{m+1} + vi, j$. Omformingen til w fører til at de første to leddene på høyreside i (4.34) være lik 0. Løser vi (4.34) med inhomogene randbetingelser må ta spesielt hensyn til integralene på randen. Videre i kapittelet vil vi bruke homogene randbetingelser for simplisitet. Matrisene vi bygger senere i kapittelet vil inkludere randnodene, og kan anvendes på et uhomogent problem, med forbehold om at vi da må ta hensyn til integralene på randen.

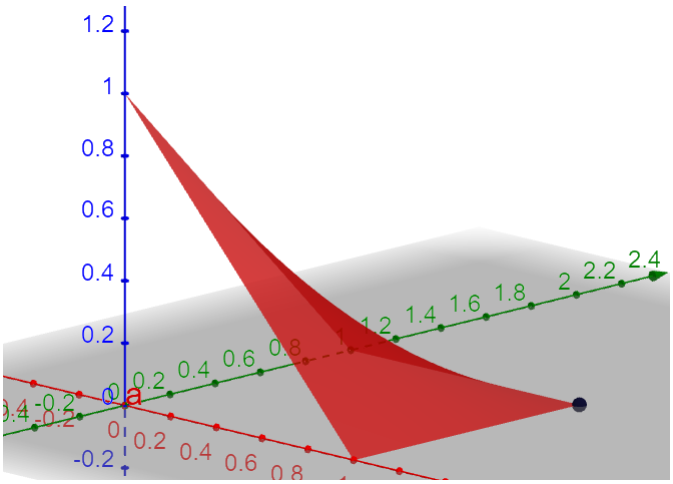
Diskretisering av området

For tydelighet presiseres det at problemet vi vil løse er gitt ved:

$$\begin{gathered} u_t - a_1 u_{xx} - a_2 u_{yy} + b_1 u_x + b_2 u_y = f(x, y, t) \quad , \quad (x, y, t) \in \mathbf{U_T} \\ u = 0 \qquad p\text{å} \quad \boldsymbol{\partial}\mathbf{U} \times (0, T] \\ u = g_2(x, y) \qquad p\text{å} \qquad \mathbf{U} \times \{t = 0\}. \end{gathered} \quad (4.36)$$

Vi oppretter noder i punktene definert i starten av kapittelet. Fire og fire noder vil danne kvadratiske elementer med bredde og høyde lik h. I de kvadratiske elementene vil funksjonen u tilnærmes ved en bilineær funksjon. Ideen er å opprette en funksjon som gir funksjonsverdien $u_{x_i, y_j, t_m} = \beta_{i,j}^m$ i noden $ih, jh, \Delta tm$, og skaper en bilineær interpolasjon av den faktiske løsningen. For hver node opprettes en basisfunksjon $\phi_{i,j}$. Den diskrete formen til u blir dermed:

$$u_{i,j}^m = \beta_{i,j,m} \phi_{i,j}. \quad (4.37)$$



Figur 4.2: Basisfunksjon sett fra en node i et element

Figur 4.2 er instruktiv i hvordan basisfunksjonene vil se ut. I (4.38) er det gitt funksjonen som beskriver det samme:

$$\phi_{i,j} = \begin{cases} (1+\frac{x-x_i}{h})(1+\frac{y-y_i}{h}) & x \in [x_{i-1}, x_i], y \in [y_{i-1}, y_i] \\ (1-\frac{x-x_i}{h})(1+\frac{y-y_i}{h}) & x \in [x_i, x_{i+1}], y \in [y_{i-1}, y_i] \\ (1+\frac{x-x_i}{h})(1-\frac{y-y_i}{h}) & x \in [x_{i-1}, x_i], y \in [y_i, y_{i+1}] \\ (1-\frac{x-x_i}{h})(1-\frac{y-y_i}{h}) & x \in [x_i, x_{i+1}], y \in [y_i, y_{i+1}] \\ 0 \quad \text{ellers} \end{cases} . \tag{4.38}$$

For randnodene vil ϕ være noe annerledes, fordi noen av områdene beskrevet i (4.38) ikke er med i $\mathbf{U_T}$. Vi bestemmer $\{\phi_{i,j} = 0\,,\ x, y \notin \overline{\mathbf{U_T}}\}$. I håp om å senere kunne utforske inhomogene randbetingelser finner vi basisfunksjonene også for randnoder. Homogene randbetingelser gjør det mulig å bestemme $\phi = 0$ på randen, da disse vil bli multiplisert med 0 (se 4.36).

Før likningssettet settes opp bestemmes testfunksjonen φ. Vi velger å bruke basisfunksjonen som testfunksjon. Dette valget ble først gjort av den russiske matematikeren Boris Galerkin. FEM med bruk av ϕ som testfunksjon kalles Galerkins metode, oppkalt etter Boris Galerkin[10].

4.4 Oppsett av FEM-skjema

Kompleksiteten i (4.34) reduseres i første omgang ved å bestemme $a_1 = a_2 = b_1 = b_2 = 1$. Fundamentet som bygges i denne seksjonen vil kunne brukes som grunnlag for mer generelle oppsett. (4.36) kan nå settes opp:

$$\sum_{i,j=1}^{i=N_1,j=N_2} \beta_{i,j}^m \Big(((\phi_{i,j})_x, (\phi_{l,k})_x) + ((\phi_{i,j})_y, (\phi_{l,k})_y) + ((\phi_{i,j})_x, (\phi_{l,k})) + ((\phi_{i,j})_y, (\phi_{l,k})) \Big)$$
$$= \sum_{i,j=1}^{i=N_1,j=N_2} (f, \phi_{l,k}) - (u_t, \phi_{l,k}). \tag{4.39}$$

hvor det er valgt en likning med utgangspunkt i hver node. Summen av likningene satt opp for hver node danner et likningssystem som kan løses på lik måte som for Crank-Nicolson-skjemaet. Den tidsderiverte blir modellert som en endelig differanse, og kan skrives:

$$\int_0^s \int_0^b u_t \phi_{l,k} dxdy = \sum_{i,j=1}^{i=N_1,j=N_2} \frac{\beta_{i,j}^{m+1} - \beta_{i,j}^{m}}{\Delta t} (\phi_{i,j}, \phi_{l,k}).$$

Vi har definert 5 ulike integraler som må finnes i hver node, i tillegg til integralet for produktet av f og ϕ. I implementeringen av skjemaet brukes det en numerisk løsning for integralet av $f\phi$. For å få en følelse for hvordan dette leddet vil se ut er det under vist et eksempel hvor $f = r$, hvor r er en konstant:

$$\int_0^s \int_0^b f\phi_{l,k} dxdy = r \int_{y_{i-1}}^{y_{i+1}} \int_{x_{i-1}}^{x_{i+1}} \phi_{i,j} dxdy = Cr \int_{y_1}^{y_2} \int_{x_1}^{x_2} \phi_{1,1} dxdy.$$

C er en konstant som avhenger av hvilken node vi er i. Hvis noden er et hjørne, er $C = 1$. For de gjenværende randnodene har vi $C = 2$, og for noder i $\mathbf{U}$ er $C = 4$. Ved bruk av (4.38) regner finner vi:

$$Cr \int_{y_1}^{y_2} \int_{x_1}^{x_2} \phi_{1,1} dxdy = Cr \frac{h^2}{4}. \tag{4.40}$$

Integralet til produktet av basisfunksjonene i hver node vil være like, bortsett fra ved randen, og det er mulig å regne ut disse analytisk. Vi vil senere se at bilineære funksjoner kan integreres nøyaktig ved hjelp av enkel gaussisk kvadratur. Hvis tidsbruk er av hensikt anbefales gaussisk kvadratur over analytisk integrasjon.

$$\int_0^s \int_0^b (\phi_{i,j})(\phi_{l,k}) dxdy = \begin{cases} = \frac{h^2}{36} & l = i-1 \text{ eller } i+1 \text{ og } k = j-1 \text{ eller } j+1 \\ = 4\frac{h^2}{36} & l = i \text{ og } k = j-1 \text{ eller } j+1 \\ = 4\frac{h^2}{36} & l = i-1 \text{ eller } i+1 \text{ og } k = j \\ = 16\frac{h^2}{36} & l = i \text{ og } k = j \\ 0 \quad \text{ellers} & \end{cases}. \tag{4.41}$$

$$\int_0^s \int_0^b (\phi_{i,j})_x (\phi_{l,k})_x dxdy = \begin{cases} = -\frac{1}{6} & l = i-1 \text{ eller } i+1 \text{ og } k = j-1 \text{ eller } j+1 \\ = 2\frac{1}{6} & l = i \text{ og } k = j-1 \text{ eller } j+1 \\ = -4\frac{1}{6} & l = i-1 \text{ eller } i+1 \text{ og } k = j \\ = 8\frac{1}{6} & l = i \text{ og } k = j \\ 0 \quad \text{ellers} & \end{cases}. \tag{4.42}$$

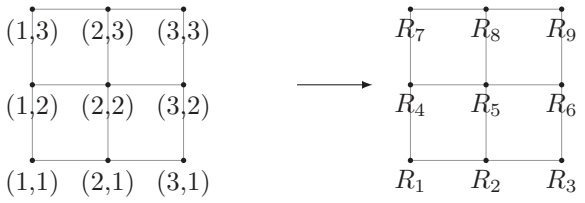
$$\int_0^s \int_0^b (\phi_{i,j})_y (\phi_{l,k})_y dxdy = \begin{cases} = -\frac{1}{6} & l = i-1 \text{ eller } i+1 \text{ og } k = j-1 \text{ eller } j+1 \\ = -4\frac{1}{6} & l = i \text{ og } k = j-1 \text{ eller } j+1 \\ = 2\frac{1}{6} & l = i-1 \text{ eller } i+1 \text{ og } k = j \\ = 8\frac{1}{6} & l = i \text{ og } k = j \\ 0 \quad \text{ellers} & \end{cases} . \tag{4.43}$$

$$\int_0^s \int_0^b (\phi_{i,j})_x (\phi_{l,k}) dxdy = \begin{cases} = \frac{h}{12} & l = i-1 \text{ og } k = j-1 \text{ eller } j+1 \\ = -\frac{h}{12} & l = i+1 \text{ og } k = j-1 \text{ eller } j+1 \\ = 4\frac{h}{12} & l = i-1 \text{ og } k = j \\ = -4\frac{h}{12} & l = i+1 \text{ og } k = j \\ 0 \quad \text{ellers} & \end{cases} . \tag{4.44}$$

$$\int_0^s \int_0^b (\phi_{i,j})_y (\phi_{l,k}) dxdy = \begin{cases} = \frac{h}{12} & l = i-1 \text{ eller } i+1 \text{ og } k = j-1 \\ = -\frac{h}{12} & l = i-1 \text{ eller } i+1 \text{ og } k = j+1 \\ = 4\frac{h}{12} & l = i \text{ og } k = j-1 \\ = -4\frac{h}{12} & l = i \text{ og } k = j+1 \\ 0 \quad \text{ellers} & \end{cases} . \tag{4.45}$$

Integralene beskrevet i (4.41)-(4.45) ha verdi 0 hvis området er utenfor $\boldsymbol{\partial}\mathbf{U}$. Tilpassede utregninger for randen ligger i vedlegg A.1. Når alle verdier er satt opp for hver node, må vi bestemme et tellesystem for nodene for å plassere dem i en matrise.

Fra (4.39) ser vi at systemet blir satt opp med hensyn til en node av gangen. Denne noden har overlappende basisfunksjon bare med nodene rundt. Fordi en node bare interagerer med nabonoder kan vi sette opp en matrise av størrelse $N_1N_2 \times N_1N_2$, hvor maks 9 elementer i hver kolonne/rad $\neq 0$. Vi lar hver kolonne i matrisene produsert med (4.41)-(4.45) ha $\phi_{i,j}$ fast, hvor $\phi_{l,k}$ varierer, og motsatt for rader. Tellesystem for å komme fra noder (x_i, y_j) til rader i en matrise fungerer som illustrert i figur 4.3:



Figur 4.3: Figuren viser hvordan nodene i diskretiseringen av U er organisert. K står for kolonne

Matrisene som oppstår ved $(\phi,\phi),(\phi_x,\phi_x)$, (ϕ_y,ϕ_y), (ϕ_x,ϕ), (ϕ_y,ϕ) ser slik ut:

$$\mathbf{M} = \begin{bmatrix} (\phi_{1,1},\phi_{1,1}) & (\phi_{2,1},\phi_{1,1}) & \cdots & (\phi_{N_1,N_2},\phi_{1,1}) \\ (\phi_{1,1},\phi_{2,1}) & (\phi_{2,1},\phi_{2,1}) & \cdots & (\phi_{N_1,N_2},\phi_{2,1}) \\ \vdots & & & \\ (\phi_{1,1},\phi_{N_1,N_2}) & (\phi_{2,1},\phi_{N_1,N_2}) & \cdots & (\phi_{N_1,N_2},\phi_{N_1,N_2}) \end{bmatrix}.$$

$$\mathbf{K_1} = \begin{bmatrix} ((\phi_{1,1})_x,(\phi_{1,1})_x) & ((\phi_{2,1})_x,(\phi_{1,1})_x) & \cdots & ((\phi_{N_1,N_2})_x,(\phi_{1,1})_x) \\ ((\phi_{1,1})_x,(\phi_{2,1})_x) & ((\phi_{2,1})_x,(\phi_{2,1})_x) & \cdots & ((\phi_{N_1,N_2})_x,(\phi_{2,1})_x)) \\ \vdots & & & \\ ((\phi_{1,1})_x,(\phi_{N_1,N_2})_x) & ((\phi_{2,1})_x,(\phi_{N_1,N_2})_x) & \cdots & ((\phi_{N_1,N_2})_x,(\phi_{N_1,N_2})_x) \end{bmatrix}.$$

$$\mathbf{K_2} = \begin{bmatrix} ((\phi_{1,1})_y,(\phi_{1,1})_y) & ((\phi_{2,1})_y,(\phi_{1,1})_y) & \cdots & ((\phi_{N_1,N_2})_y,(\phi_{1,1})_y) \\ ((\phi_{1,1})_y,(\phi_{2,1})_y) & ((\phi_{2,1})_y,(\phi_{2,1})_y) & \cdots & ((\phi_{N_1,N_2})_y,(\phi_{2,1})_y)) \\ \vdots & & & \\ ((\phi_{1,1})_y,(\phi_{N_1,N_2})_y) & ((\phi_{2,1})_y,(\phi_{N_1,N_2})_y) & \cdots & ((\phi_{N_1,N_2})_y,(\phi_{N_1,N_2})_y) \end{bmatrix}.$$

$$\mathbf{C_1} = \begin{bmatrix} ((\phi_{1,1})_x,\phi_{1,1}) & ((\phi_{2,1})_x,\phi_{1,1}) & \cdots & ((\phi_{N_1,N_2})_x,\phi_{1,1}) \\ ((\phi_{1,1})_x,\phi_{2,1}) & ((\phi_{2,1})_x,\phi_{2,1}) & \cdots & ((\phi_{N_1,N_2})_x,\phi_{2,1}) \\ \vdots & & & \\ ((\phi_{1,1})_x,\phi_{N_1,N_2}) & ((\phi_{2,1})_x,\phi_{N_1,N_2}) & \cdots & ((\phi_{N_1,N_2})_x,\phi_{N_1,N_2}) \end{bmatrix}.$$

$$\mathbf{C_2} = \begin{bmatrix} ((\phi_{1,1})_y,\phi_{1,1}) & ((\phi_{2,1})_y,\phi_{1,1}) & \cdots & ((\phi_{N_1,N_2})_y,\phi_{1,1}) \\ ((\phi_{1,1})_y,\phi_{2,1}) & ((\phi_{2,1})_y,\phi_{2,1}) & \cdots & ((\phi_{N_1,N_2})_y,\phi_{2,1})) \\ \vdots & & & \\ ((\phi_{1,1})_y,\phi_{N_1,N_2}) & ((\phi_{2,1})_y,\phi_{N_1,N_2}) & \cdots & ((\phi_{N_1,N_2})_y,\phi_{N_1,N_2}) \end{bmatrix}.$$

$\mathbf{M}, \mathbf{K_1}$ **og** $\mathbf{K_2}$ har symmetriske rader og kolonner, og er positive definit fordi de beskriver et kvadrat. Det er vanlig å kalle $\mathbf{M}$ for *massematrisen* og $\mathbf{K_1} + \mathbf{K_2} + \mathbf{C_1} + \mathbf{C_2} = \mathbf{K}$ for stivhetsmatrisen. Vi skriver *kraftvektoren* $\mathbf{F}$ som:

$$\mathbf{F} = \begin{bmatrix} (f,\phi_{1,1}) \\ (f,\phi_{2,1}) \\ \vdots \\ (f,\phi_{N_1,N_2}) \end{bmatrix}.$$

Med matrisene over kan vi sette opp (4.39) på en kompakt og ryddig form:

$$\mathbf{M}\cdot\frac{\beta^{m+1}-\beta^m}{\Delta t} + \mathbf{K}\cdot\beta^m = \mathbf{F}.$$

som gir:

$$\beta^{m+1} = \beta^m - \mathbf{M}^{-1}\cdot\Delta t\cdot\mathbf{K}\cdot\beta^m + \mathbf{M}^{-1}\cdot\Delta t\cdot\mathbf{F}. \tag{4.46}$$

4.5 Positivitetsbevaring i FEM-skjema

(4.46) skrives om til:

$$\mathbf{M} \cdot \beta^{m+1} = (\mathbf{M} - \Delta t \cdot \mathbf{K})) \cdot \beta^m + \Delta t \cdot \mathbf{F}. \tag{4.47}$$

Det er naturlig å kreve at høyresiden av (4.47) bare kan gi positive verdier hvis positivitet skal bevares. a_1, a_2, b_1, og b_2 velges nå som vilkårlige konstanter, med begrensning $a_1, a_2 > 0$.

Stasjonært problem

Undersøkelsen av positivitet starter med å undersøke det stasjonære problemet:

$$\begin{aligned} -a_1 u_{xx} - a_2 u_{yy} + b_1 u_x + b_2 u_y = f(x,y,t) \quad , \quad (x,y,t) \in \mathbf{U_T} \\ u = 0 \qquad \text{på} \quad \boldsymbol{\partial}\mathbf{U} \times [0,T]. \end{aligned} \tag{4.48}$$

(4.48) er tett knyttet til (4.36) og ved å bruke skjemaet som allerede er utviklet kjenner vi igjen en løsningen som:

$$\mathbf{K} \cdot \beta = \mathbf{F}. \tag{4.49}$$

Fra kapittel 3 vet vi at det mest interressante tilfellet for positivitetsbevaring er $f \geq 0$. Om vi kan vise at $\mathbf{K}$ er en M-matrise kan vi konkludere med at alle elementene i β er positive, og skjemaet er positivitetsbevarende. For en M-matrise må summen av radene i matrisen ha verdi ≥ 0, med alle diagonalelementene > 0, og alle elementer utenfor diagonalen ≤ 0[20]. Ved å sette sammen verdier fra (4.42)-(4.45), og tilsvarende fra A.2 har vi:
For hjørnenoder:

$$4(a_1 + a_2) - h(|2(b_1 + b_2)|) \geq 0.$$
$$-4a_1 + 2a_2 + h(|2b_1 - b_2|) \leq 0.$$
$$2a_1 - 4a_2 + h(|-b_1 + 2b_2|) \leq 0.$$

Noder i nedre/øvre del av ∂U:

$$8(a_1 + a_2) - h|4b_2| \geq 0.$$
$$-4a_1 + 2a_2 + h(|2b_1 + b_2|) \leq 0.$$
$$4a_1 - 8a_2 + h|4b_2| \leq 0.$$
$$-2(a_1 + a_2) + h(|b_1 - b_2|) \leq 0.$$

høyre/venstre del av ∂U:
Tilsvarende som for nedre/øvre, men byttet a_1 for a_2, og b_1 for b_2.
Noder inne i U:

$$16(a_1 + a_2) \geq 0.$$
$$-2(a_1 + a_2) + h(|h_1 + h_2|) \leq 0.$$

$$4a_1 - 8a_2 + h|4b_2| \leq 0.$$
$$-2(a_1 + a_2) + h(|h_1 - h_2|) \leq 0.$$
$$-8a_1 + 4a_2 + h|4b_1| \leq 0.$$

Merk at ulikhetene er multiplisert med 12 for å unngå uttrykk med brøker. For at ulikhetene skal holde må a_1 og a_2 begrenset ved:

$$\frac{a_1}{2} < a_2 < 2a_1. \tag{4.50}$$

Vi vil se at (4.50) ikke er nødvendig for stabilitet i (4.47), men det er nødvendig for positivitet i det stasjonære problemet. Ulikhetene som er vanskeligst å oppfylle ligge langs randen hvor

$$-4a_1 + 2a_2 + h(|2b_1 + b_2|) \leq 0.$$

gir:

$$h \leq \frac{4a_1 - 2a_2}{|2b_1 + b_2|}. \tag{4.51}$$

(4.51) kommer i fire forskellige varianter hvor a_1 og a_2 bytter plass og b_1 og b_2 bytter plass. Derfor får vi kravet:

$$h < \min\left\{ \frac{4a_1 - 2a_2}{\max\{|2b_1 + b_2|, |b_1 + 2b_2|\}}, \frac{-2a_1 + 4a_2}{\max\{|2b_1 + b_2|, |b_1 + 2b_2|\}} \right\}. \tag{4.52}$$

Til sammenlikning kan vi velge gunstige parametre for$a_1 = a_2$ og $b_1 = 0$ hvor kravet sammenfaller med (4.15). Endres forholdet mellom a_1 og a_2), eller $b_1, b_2 \neq 0$ vil (4.52) være strengere enn tilsvarende krav for forlengs Euler.I neste seksjon vil vi se at (4.52) er strengere enn nødvendig. For et skjema som tar høyde for homogene randbetingelse trengs kun krav for nodene inne i området $\mathbf{U_T}$ hvor ulikhetene over er oppfylt for:

$$h < \min\left\{ \frac{4a_1 - 2a_2}{|2b_1|}, \frac{-2a_1 + 4a_2}{|2b_2|} \right\}. \tag{4.53}$$

Fordelen med å oppfylle (4.50), og (4.52), selv om de ikke vil være strengt nødvendige, er at vi kan bruke $\mathbf{K}$ er en M-matrise som et kraftig argument. Ulempen med (4.50), og (4.52) er at førstnevnte begrenser kraftig mengden problemer som kan løses, og sistnevne er unødvendig strengt. I neste seksjon vil vi se krav som gir positiv høyreside i (4.46) uten de samme begrensningene som det stasjonære problemet.

Tidsutvikling med f større enn, eller lik 0

Om $f \geq 0$ vil $\Delta t \cdot \mathbf{F}$ også ha den samme egenskapen, fokuset er derfor på positivitet i $(\mathbf{M} - \Delta t \cdot K) \cdot \beta^m$. Hvis $\mathbf{K}$ oppfyller kravene til en M-matrise vet vi at $\mathbf{K}$ har positive elementer bare langs diagonalen, og dermed at $-\Delta t\mathbf{K}$ bare har negative elementer på diagonalen. For å oppnå en $\mathbf{M} - \Delta t \cdot K$ hvor alle elementer er ≥ 0 må vi derfor se på verdier langs diagonalelementer:

$$\max\left\{ |\frac{\Delta t K_{ii}}{M_{ii}}| \right\} \leq 1.$$

For å unngå krav av typen (4.50), som begrenser forholdet mellom parametrene kan vi i tillegg kreve:

$$\min\left\{|\frac{\Delta t K_{ii}}{M_{ii}}|\right\} \geq \max\left\{|\frac{\Delta t K_{ij}}{M_{ij}}|\right\},\ i \neq j.$$

som sikrer at forholdet langs diagonalen bestemmer positiviteten til høyresiden av (4.47). Ved samme argument om homogene randbetingelser som i forrige seksjon er ulikhetene som må undersøkes redusert til:

$$\max\left\{|\frac{\Delta t K_{ij}}{M_{ij}}|\right\} \leq \max\left\{\frac{|36\Delta t(-2(a_1+a_2)+h(b_1+b_2))|}{48h^2}, \right.$$
$$\left.\frac{|36\Delta t(-8a_1+4a_2+h(4b_1))|}{96h^2}\ ,\ \frac{|36\Delta t(4a_1-8a_2+h(4b_2))|}{96h^2}\right\}. \quad (4.54)$$

og:

$$\left\{|\frac{\Delta t K_{ii}}{M_{ii}}|\right\} = \frac{36\Delta t(16(a_1+a_2)}{192h^2}. \quad (4.55)$$

Vi krever (4.53) og (4.54) og får

$$16(a_1+a_2) \geq 8|a_1-2a_2|+4h|b_2|.$$

$$h \leq \frac{2(a_1+a_2)-|a_1-2a_2|}{|b_2|}.$$

Ulikheten over blir strengest ved valg av $a_2 > a_1$. Settes tilsvarende ulikhet opp for de to resterende uttrykkene i (4.54) har vi:

$$h \leq \min\left\{\frac{3a_1}{|b_2|}\ ,\ \frac{3a_2}{|b_1|}\ ,\ \frac{2(a_1+a_2)}{|b_1+b_2|}\right\}. \quad (4.56)$$

som likner på kravet for C-N skjemaet. Vi kan nå teste om forholdet på diagonalen er mindre enn 1. Fra (4.55) har vi:

$$3\Delta t\frac{a_1+a_2}{h^2} \leq 1.$$

som gir:

$$\Delta t \leq \frac{h^2}{3(a_1+a_2)}. \quad (4.57)$$

Sammenliknet med tilsvarende krav (4.16) og (4.29) for både forlengs Euler og C-N er (4.57) strengere.

Hva med negativ f?

Vi vet fra kapittel 3 at positivitetsbevaring over lengre tid for negative funksjoner ikke er gitt. Spørsmålet er om vi kan gå kortere perioder med negativ f. Starter med å finne et forhold γ:

$$\gamma = max\{\frac{|\mathbf{F}_i|}{\beta_i}\}.$$

Siden $\beta_i = 0$ på randen krever vi:

$$\mathbf{f} = 0, \quad x \in \partial\mathbf{U_T}.$$

eller mer generelt:

$$f(x_i, y_j, t_m) = 0 \quad \text{hvis } u(x_i, y_j, t_m) = 0. \tag{4.58}$$

Begrensningene gitt i (4.58) gir positivitet på høyreside av (4.47) for:

$$\Delta t \leq \frac{h^2}{3(a_1 + a_2) + 9\gamma}. \tag{4.59}$$

Som for forlengs Euler kan negative funksjonsverdier i f aksepteres over begrensede tidsperioder.

Massematrisen og venstresiden

Med kravene (4.55) og (4.57) kan (4.47) skrives:

$$\mathbf{M} \cdot \beta^{m+1} = \mathbf{b}. \tag{4.60}$$

hvor $\mathbf{b}$ er en vektor verdi ≥ 0 for alle elementer. Spørsmålet om positivitet går nå videre til massematrisen $\mathbf{M}$. Vi vet at $\mathbf{M}$ er positiv definit, som gir en positiv definit inversmatrise. Eksistens av en inversmatrise til $\mathbf{M}$ bekrefter at (4.47) er løselig, men fungerer dårlig som argument for positivitetsbevaring. En positiv definit matrise impliserer ikke positiv $\mathbf{x}$ for $\mathbf{M} \cdot \mathbf{x} = \mathbf{y}$, hvor $y_i \geq 0$. Vi kan forsøke å bruke egenskapen at $\mathbf{M}$ bare inneholder positive elementer til å si at $x_i \geq 0$ impliserer $\mathbf{M} \cdot \mathbf{x} = \mathbf{y}$ igjen med $\mathbf{y_i} \geq 0$. Desverre holder ikke denne antakelsen begge veier. Kan vi vise at massematrisen er en M-matrise ville påstanden holdt begge veier , men vi finner enkelt eksempler som viser at massematrisen ikke er en M-matrise. Under er et eksempel for et grid med 3×3 punkter:

$$\mathbf{M} \cdot \mathbf{x} = \mathbf{y}$$

$$\frac{h^2}{36} \begin{bmatrix} 4 & 2 & 0 & 2 & 1 & 0 & 0 & 0 & 0 \\ 2 & 8 & 2 & 1 & 4 & 1 & 0 & 0 & 0 \\ 0 & 2 & 4 & 0 & 1 & 2 & 0 & 0 & 0 \\ 2 & 1 & 0 & 8 & 4 & 0 & 2 & 1 & 0 \\ 1 & 4 & 1 & 4 & 16 & 4 & 1 & 4 & 1 \\ 0 & 1 & 2 & 0 & 4 & 8 & 0 & 1 & 2 \\ 0 & 0 & 0 & 2 & 1 & 0 & 4 & 2 & 0 \\ 0 & 0 & 0 & 1 & 4 & 1 & 2 & 8 & 2 \\ 0 & 0 & 0 & 0 & 1 & 2 & 0 & 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} -1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \frac{h^2}{36} \begin{bmatrix} 1 \\ 14 \\ 9 \\ 14 \\ 34 \\ 18 \\ 9 \\ 18 \\ 9 \end{bmatrix}.$$

Eksempelet over tar ikke hensyn til homogene randbetingelser, og inneholder bare ett indre punkt. Ved utvidelse til et 4×4 grid vil gi likt resultat selv med krav om u=0 på randen.

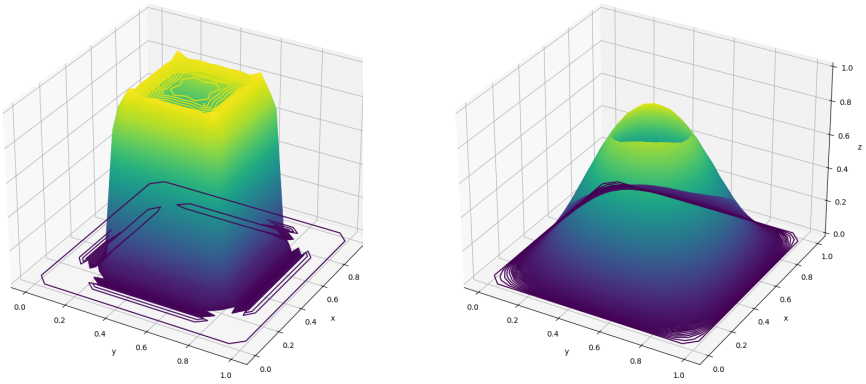
En utforskende metode

Siden vi ikke har funnet konkrete krav for positivitet i (4.47) analytisk, søkes bedre innsikt med empirisk utforsking. Brukes kravene (4.56), og (4,57) som utgangspunkt kan vi utforske en implementering av (4.47). Koden til

skjemaet ligger i vedlegg B.4 og B.3. For utforskingen er det valgt $\mathbf{F} = 0$, $a_1 = a_2 = b_1 = b_2 = 1$, og initialbetingelser:

$$u(x, y, 0) = \begin{cases} 1 & hvis \quad \frac{1}{4} < x, y < \frac{3}{4}. \\ 0 & \text{ellers.} \end{cases} \tag{4.61}$$

Kuboidefunksjonen (4.61) er brukt som en stresstest for skjemaet grunnet den bratte overgangen fra 0 til 1. For $h = \frac{1}{2}$ vil kuboidefunksjonen likne en Dirac-delta funksjon med verdi i bare ett punkt. Den bredere formen til kuboiden i forhold til å bruke Dirac-delta gjør det enklere å se skjemaet visuelt. Brukes (4.57) som krav til Δt er (4.47) ustabilt for kuboidefunksjonen (4.61). Tidsinndeling nøyaktig dobbelt så fin som (4.57) gir stabilitet, men det oppstår små bølger med negative verdier i rommet mellom randen og firkanten $\frac{1}{4} < x, y < \frac{3}{4}$.



(a) Kuboide etter ett tidssteg (b) Kuboide etter litt tid

Figur 4.4: Stresstest av skjema
h=1/20 og Δt=1/10000

Figur 4.4(a) har bølger som oscillerer mellom små positive og små negative verdier i området mellom kuboiden og randen, hvor den laveste bølgebunnen har størrelse $-2 \times 10^{-}2$. Finere inndeling i tid indikerer at bølgene av negative verdier er feilledd, hvor de negative verdiene synker for mindre Δt. Oscillasjonene synker også i verdi over tid. Etter seks tidssteg er bølgene av negative verdier fra figur 4.4(a) borte, og positivitet er bevart etter dette. Med observasjonene gjort ved utforsking ser vi på oscillasjonene som feilledd, og ignorerer negative verdier-så lenge skjemaet er stabilt. Før vi ser på stabilitet introduseres en modifikasjon av (4.47) som bevarer positivitet.

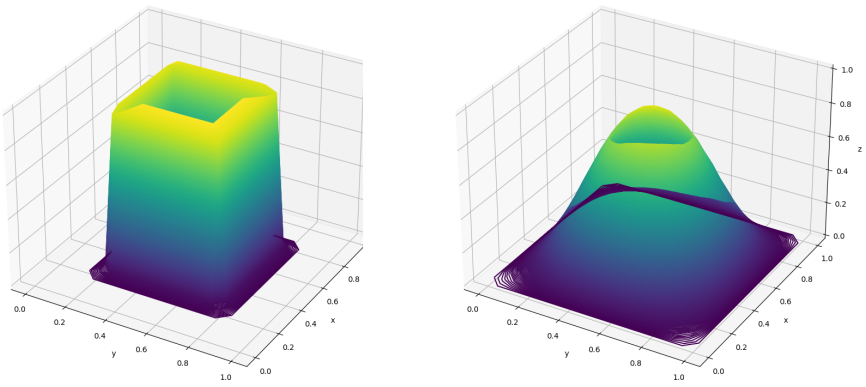
Mass-lumping

Massematrisen fører til at standard FEM ikke oppfyller det ønskede maksimumprinsipper $u_{i,h}^m \geq 0$. Et forsøk på bedre strukturbevaring, og effektivisering

av (4.47) er mass-lumping av massematrisen. Mass-lumping bruker at verdien i nabonoder vil konvergere mot hverandre hverandre for mindre h [11]. Ved å summere alle elementer i rad n av massematrisen produseres en diagonal, mass-lumped versjon av $\mathbf{M}$. $\mathbf{K}$ blir behandlet som tidligere og positivitet er derfor ikke garantert ved (4.56) og (4.57) alene. Kreves (4.50), (4.53), og (4.57) vil $-\mathbf{K}$ inneholde bare positive elementer utenfor diagonalen, og positivitet er garantert. Antas stabilitet i en mass-lumped versjon av (4.47) vil likningen vi løser ha form:

$$h^2 \beta_{i,j}^{m+1} = C_1. \tag{4.62}$$

hvor C_1 er en konstant. Dersom effektivitet verdsettes er mass-lumping et attraktivt alternativ til standard FEM fordi det reduserer antall operasjoner i hvert tidssteg.



(a) Kuboide etter ett tidssteg (b) Kuboide etter litt tid

Figur 4.5: Test av mass-lumped skjema
h=1/20 og Δt=1/10000

Figur 4.5 viser en mass-lumped versjon med like initialbetingelser som i figur 4.4. I kjøreeksempelet er positivitet bevart.

Krav som avhenger av initialdata

(4.50) er ikke alltid oppfylt, derfor presenteres et forslag til krav for h, og Δt som ikke bygger på denne antakelsen. Vi utleder først krav til mass lumped skjema, som enkelt utvides til tilsvarende krav for standard FEM. Nabonoder relateres til hverandre ved å finne den største stigningen i diskretiseringen av u:

$$C_{ij} = max\{|g_2((i \pm 1)h, jh) - g_2(ih, jh)|, |g_2(ih, (j \pm 1)h) - g_2(ih, jh)|\}.$$

En nedre grense til forskjellen mellom to etterflgende tidssteg kan beskrives med ulikheten:

$$h^2 u_{i,j}^{m+1} \geq h^2 u_{i,j}^{m} - \frac{\Delta t}{12}\left(\mathbf{K_{p1}} \cdot \mathbf{b_{p1}^T} + \mathbf{K_{p2}} \cdot \mathbf{b_{p2}^T}\right) \geq 0. \tag{4.63}$$

for en valgt node (x_i, y_j). Vektorene $\mathbf{K_{p1}}, \mathbf{K_{p2}}$ og$\mathbf{b_{p2}}, \mathbf{b_{p2}}$ er gitt ved:

$$\begin{aligned}
\mathbf{K_{p1}} &= [-2a_1 - hb_1, 4a_1, -2a_1 + hb_1, -8a_1 - 4hb_1, 16a_1 \\
&\quad , -8a_1 + 4hb_1, -2a_1 - hb_1, 4a_1 + 4hb_2, -2a_1 + hb_1]. \\
\mathbf{K_{p2}} &= [-2a_2 - hb_2, -8a_2 - 4hb_2, -2a_2 - hb_2, 4a_2, 16a_2 \\
&\quad , 4a_2, -2a_2 + hb_2, -8a_2 + 4hb_2, -2a_2 + hb_2]. \\
\mathbf{b_{p1}} &= u^m_{i,j}[1,1,1,1,1,1,1,1,1] + C_{ij}[-1,1,-1,-1,0,-1,-1,1,-1]. \\
\mathbf{b_{p2}} &= u^m_{i,j}[1,1,1,1,1,1,1,1,1] + C_{ij}[-1,-1,-1,1,0,1,-1,-1,-1]. \quad (4.64)
\end{aligned}$$

a_1 og a_2 vil være de uttellende leddene om (4.53) kreves, (4.62) bestemmer derfor en øvre grense for endringen i et tidssteg:

$$h^2 u^{m+1}_{i,j} \geq h^2 u^m_{i,j} - \frac{16 C_{ij} \Delta t}{12}(a_1 + a_2) \geq 0.$$

$$u^m_{i,j} - u^{m+1}_{i,j} \leq \frac{4 C_{ij} \Delta t}{3h^2}(a_1 + a_2). \quad (4.65)$$

$$\Delta t \leq \frac{3h^2 u^m_{i,j}}{4 C_{ij}(a_1 + a_2)}. \quad (4.66)$$

(4.65) og (4.66) er ikke eksplisitte som tidligere krav, men vi forsøker å anvende dem for kuboidefunksjonen (4.61). Det er naturlig å teste området rundt:

$$\begin{cases} x = \frac{1}{4} \; eller \; \frac{3}{4} \; og \; y \in [\frac{1}{4}, \frac{3}{4}] \\ y = \frac{1}{4} \; eller \; \frac{3}{4} \; og \; x \in [\frac{1}{4}, \frac{3}{4}] \end{cases}.$$

hvor den største differansen $C_{ij} = 1$ er oppnådd.

$$u^m_{i,j} - u^{m+1}_{i,j} \leq \frac{4\Delta t}{3h^2}(a_1 + a_2).$$

$$\Delta t \leq \frac{3h^2 u^m_{i,j}}{4(a_1 + a_2)}.$$

Utenfor kuboiden er initialbetingelsene 0, som gir:

$$\begin{aligned} -u^{m+1}_{i,j} &\leq \frac{4\Delta t}{3h^2}(a_1 + a_2). \\ \Delta t &\leq 0. \end{aligned} \quad (4.67)$$

(4.67) gjør tidsutvikling med bevart positivitet umulig for kuboidefunksjonen. Kontinuerlige initialbetingelser $g_2 > \epsilon$, for en valgt $\epsilon > 0$ og (4.53), gir positivitetsbevaring ved:

$$\Delta t \leq \frac{3h^2 \epsilon}{max\{C_{ij}\} 4(a_1 + a_2)}. \quad (4.68)$$

Analoge krav blir funnet for standard FEM ved å gjeninnføre massematrisen og realatere denne til et ekstremtilfelle:

$$h^2 u^{m+1}_{i,j} + \frac{20 C_{ij} h^2}{36} \geq h^2 u^m_{i,j} - \frac{20 C_{ij} h^2}{36} - \frac{16 C_{ij} \Delta t}{12}(a_1 + a_2) \geq 0. \quad (4.69)$$

som gir for $g_2 \geq \epsilon$:

$$\Delta t \leq \frac{3h^2}{4(a_1 + a_2)} \left(\frac{\epsilon}{max\{C_{ij}\}} - \frac{2}{3} \right). \tag{4.70}$$

(4.68) og (4.70) viser at positivitet kan bevares for både FEM, og mass-lumped FEM med begrensninger av initialbetingelsene. Begrensningen av initialbetingelser er kanskje akseptabel fra et fysisk standpunkt, hvor absolutt null kan nærmes, men aldri oppnås. Matematisk er valg av $g_2 > \epsilon$ utilfredsstillende fordi vi fortsatt vil ha:

$$\lim_{u \to 0} \Delta t = 0.$$

Det vil si, Δt må bestemmes forsvinnende liten for initialbetingelser som har stor variasjon, kombinert med verdier nær 0. I skjemaene med endelige elementer begrenser vi oss til å kreve stabilitet for alle a_1 , a_2 og positivitet i mass lumped FEM bare hvis (4.50) er oppfylt.

4.6 Stabilitet

Kravene (4.56) og (4.57) gir stabilitet i (4.47) stabilt for glatte nok initialbetingelser, for dobbelt så strenge krav er (4.47) stabilt for kuboidefunksjonen. FEM-skjemaet ser derfor ut til å være stabilt ved:

$$\Delta t \leq \frac{h^2}{6(a_1 + a_2)}. \tag{4.71}$$

En inhomogen kraftfunksjon $f \geq 0$ gir også god oppførsel hvis (4.52) og (4.71) er oppfylt. I delkapittel 4.5 fant vi ved utforsking at feilleddene er oscillerende i natur, som motiverer bruk av von Neumann stabilitetsanalyse på lik linje som forlengs Euler og Crank-Nicolson. Vi setter opp et uttrykk for feilledd i en node:

$$E(\Delta t(m+1)) \sum_{p_2=-1}^{1} \sum_{p_1=-1}^{1} q_{ij} e^{z(k(i+p_1)+l(j+p_2))} = E(\Delta tm) \left(\sum_{p_2=-1}^{1} \sum_{p_1=-1}^{1} q_{ij} e^{z(k(i+p_1)+l(j+p_2))} - \sum_{p_2=-1}^{1} \sum_{p_1=-1}^{1} d_{ij} e^{z(k(i+p_1)+l(j+p_2))} \right). \tag{4.72}$$

hvor:

$$q_{ij} = \frac{h^2}{36}[1, 4, 1, 4, 16, 4, 1, 4, 1].$$

$$d_{ij} = \frac{\Delta t}{12}\left(\mathbf{K_{p1}} + \mathbf{K_{p2}}\right).$$

Som i tidligere stabilitetsanalyser deler vi begge sider på $E(\Delta tm)e^{z(ki+lj)}$. Etter en del algebra og telling sitter vi igjen med:

$$\begin{aligned}\frac{E(\Delta t(m+1))}{E(\Delta tm)} &= \frac{Faktor_1 + Faktor_2}{Faktor_2}\\ Faktor_1 &= 6\Delta t(a_1+a_2)(e^{-z(k+l)}+e^{z(k+l)}) + 12\Delta t(-a_1+2a_2)(e^{-zl}+e^{zl})\\ &\quad + 6\Delta t(a_1+a_2)(e^{-z(-k+l)}+e^{z(-k+l)})\\ &\quad + 12\Delta t(2a_1-a_2)(e^{-zk}+e^{zk}) - 48\Delta t(a_1+a_2)\\ Faktor_2 &= h^2(e^{-z(k+l)}+e^{z(k+l)}) + 4h^2(e^{-zl}+e^{zl}) + h^2(e^{-z(-k+l)}\\ &\quad + e^{z(-k+l)}) + 4h^2(e^{-zk}+e^{zk}) + 16h^2.\end{aligned} \tag{4.73}$$

Ved å anvende $e^{-ik}+e^{ik} = 2cos(k)$ har vi:

$$\begin{aligned}Faktor_1 &= 12\Delta t(a_1+a_2)cos(\pi h(k+l)) + 24\Delta t(-a_1+2a_2)cos(\pi hl)\\ &+ 12\Delta t(a_1+a_2)cos(\pi h(-k+l)) + 24\Delta t(2a_1-a_2)cos(\pi hk) - 48\Delta t(a_1+a_2)\\ Faktor_2 &= 2h^2cos(\pi h(k+l)) + 8h^2cos(\pi hl) + 2h^2cos(\pi h(-k+l)) + 8h^2cos(\pi hk) + 16h^2.\end{aligned} \tag{4.74}$$

Vi er ute etter å finne krav som oppfyller:

$$|\frac{E(\Delta t(m+1))}{E(\Delta tm)}| \leq 1.$$

$Faktor_2$ er alltid positiv, dermed har vi kravet:

$$-2 \leq \frac{Faktor_1}{Faktor_2} \leq 0. \tag{4.75}$$

Høyre ulikhet i (4.75) er oppfylt for alle h og Δt. For å vise den venstre ulikheten i (4.75) bruker vi kombinasjonen $cos(\pi hk = -1$ og $cos(\pi hl = -1$ som gir det strengeste kravet:

$$-2 \leq \frac{6\Delta t(a_1(1+2+1-4-4)+a_2(1-4+1+2-4))}{h^2(1-4+1-4+8)}.$$

$$\frac{-12\Delta t(a_1+a_2)}{h^2} \leq 2.$$

$$\Delta t \leq \frac{h^2}{6(a_1+a_2)}.$$

Stabilitetskravet utledet med von Neumann stabilitetsanalyse stemmer overens med (4.71) som ble funnet empirisk. Sammenhengen mellom den empiriske stabiliteten og von Neumann stabilitet indikerer at von Neumann også kan anvendes på FEM-skjemaer for parabolske problemer. Tilsvarende utregninger som over for en mass-lumped versjon av (4.47) gir:

$$\Delta t \leq \min\{\frac{h^2}{4(a_1)}, \frac{h^2}{4(a_2)}\}. \tag{4.76}$$

(4.76) gir ved en test på kuboidefunksjonen (4.61) løsninger med god oppførsel for ML-FEM (mass-lumped finite element method).

4.7 Praktisk utførelse

Forlengs Euler

Forlengs Euler implementeres ved å definere initialbetingelser, randbetingelser, parametre, og en loop som regner ut (4.14) i alle punkter for hvert tidssteg. Foran loopen som regner ut (4.14) er en test for h og Δt som korrigerer disse verdiene hvis de ikke tilfredsstiller (4.16) og (4.17). Koden er gitt i B.1 og kan produsere et bilde av sluttilstanden, eller en animasjon av utviklingen. Animasjonen er programmert til å gå flere tidssteg av gangen etterhvert som tiden går.

Crank-Nicolson

Parametre, inital-og randbetingelser opprettes på lik linje som i forlengs Euler. Løsningen skjer deretter ved å sette opp matrisen $\mathbf{S}$ fra delkapittel 4.2 i glissen form. Høyresiden i (4.27) ($\mathbf{H_s}$) blir regnet ut, og vi løser:

$$(\mathbf{I} + \mathbf{S}) \cdot \mathbf{u^{m+1}} = \mathbf{H_s}.$$

Cholmod pakken, som utnyttes i implementeringen av FEM-skjemaene, kan ikke brukes fordi $\mathbf{S}$ ikke er symmetrisk. Biblioteket *scikit.sparse*[23] blir brukt for løse skjemaet ved LU-faktorisering. Løsning av $(\mathbf{I} + \mathbf{S} \cdot u^{m+1}) = \mathbf{b}$ ved LU faktorisering er analogt med løsning ved Cholesky-faktorisering beskrevet for implementering av FEM. Forskjellen er at matrisen ikke trenger å være symmetrisk for LU-faktorisering, og algoritmen er som konsekvens omtrent dobbelt så treg[17]. Koden i B.2 produserer et bilde av sluttilstanden, men utvides enkelt til å se på utviklingen i tid ved å kopiere over deler av forlengs Euler.

FEM-skjema

Massematrisen $\mathbf{M}$ er inverterbar, fordi den er positiv definit, og skjemaet gitt i (4.46) er løselig. Det er ikke ønskelig å løse (4.46) slik det er fordi løsningen krever invertering av en $N_1N_2 \times N_1N_2$ matrise. Å løse skjemaet på formen (4.47) er også krevende, men vi kan utnytte at $\mathbf{M}$ og $\mathbf{K}$ er glissne matriser. En glissen matrise er en matrise hvor de fleste elementene har verdi 0. For hver rad, og hver kolonne i begge matrisene er det maks 9 elementer med verdi $\neq 0$. $0 \times x = 0 \, \forall \, x$. Elementer med verdi 0 er lite effektivt å ha med i utregningen. Elementer med verdi $\neq 0$ blir med pakken *sksparse* lagres som en kombinasjon av verdien til elementet, og et koordinat. Resultatet av å bruke *sksparse* er en liste med verdier og tilhørende punkter, som kan behandles som den originale matrisen i utregninger (Scikit sparse bibliotek: [18]). I tillegg til å utnytte at $\mathbf{M}, \mathbf{K}$ er glissne, kan vi utnytte at $\mathbf{M}$ er symmetrisk. Cholesky-faktorisering, som beskrevet i (4.77) anvendes fordi M er positiv definit og symmetrisk [21]:

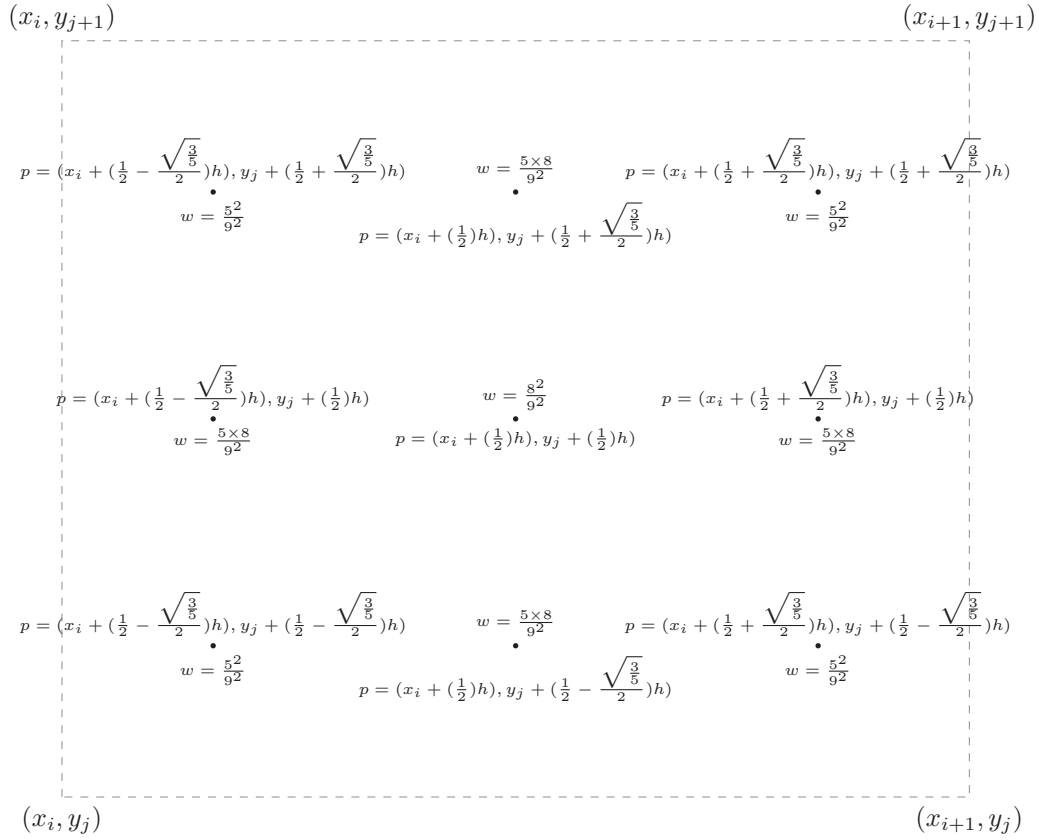
$$\begin{aligned} \mathbf{M}\beta &= \mathbf{b} \\ \mathbf{L} \cdot \mathbf{L^T}\beta &= \mathbf{b}. \end{aligned} \tag{4.77}$$

(4.77) løses effektivt ved:

$$\mathbf{Ly} = b.$$

$$\mathbf{L^T}\beta = \mathbf{y}.$$

Integrasjon av produktet til en vilkårlig funksjon f og basisfunksjonene ϕ er av tidshensyn ikke gjennomførbart analytisk. Gauss-Legendre kvadratur (GL kvadratur) anvendes for numerisk integrasjon. GL kvadratur gir eksakt løsning for integralet av polynomer opp til grad $2n-1$[19], hvor n tilsvarer antall $Gauss-punkter$. For 3×3 Gauss-punkter vil utregningen av integralet settes opp som illustrert i figur 4.6 og likning (4.78).



Figur 4.6: Figuren viser vekting og organisering av punkter for bruk i GL kvadratur

Summen av alle vekttallene w er 4, og arealet til området er h^2. Dermed blir formelen for integralet:

$$\int_{x_i}^{x_{i+1}} \int_{y_j}^{y_{j+1}} f\phi \; dydx \approx \frac{h^2}{4} \sum f(p)w. \tag{4.78}$$

hvor vi summerer over alle punktene i figur 4.6. Koden til FEM-implementeringen er delt inn i to deler. I den første delen velges h og Δt, som blir korrigert

automatisk om de ikke oppfyller (4.56) og (4.71) for standard løsning, eller (4.56) og (4.76) for mass-lumped skjema. Den første delen av FEM-implementeringen oppretter i tillegg alle vektorer og matriser vi trenger for å sette opp (4.47), og en valgfri test for å begrense feilen i starttilstanden. Den valgfrie testen for feil i starttilstanden fungerer ved å finne den største deltaen for nabonoder og sjekker om produktet av deltaen og h^2 er mindre enn en bestemt verdi. Andre del av FEM-implementeringen løser skjemaet (4.47) for parametrene valgt i den første, enten som animasjon, eller som stillebilde for en gitt sluttid. Valget $kompansert = 1$ i koden løser problemet for homogene randbetingelser. Velges heller $kompansert = 0$ er koden per nå en ufullstendig implementasjon av et skjema for inhomogene randbetingelser. Første del av koden ligger ved i B.4, og del 2 i B.3

KAPITTEL 5

Sammenlikning av skjema

5.1 En analytisk løsning

For å vurdere numeriske skjema, er det ønskelig med en analytisk løsning. Den analytiske løsningen må tilfredsstille homogene randbetingelser for en firkant $(0,1) \times (0,1)$:

$$u(x,y,t) = e^{-2t\pi^2} sin(\pi x) sin(\pi y). \tag{5.1}$$

Vi regner ut de partiellderiverte:

$$\begin{aligned} u_t &= -2\pi^2 e^{-2t\pi^2} sin(\pi x) sin(\pi y). \\ u_x &= e^{-2t\pi^2} \pi cos(\pi x) sin(\pi y). \\ u_{xx} &= -\pi^2 e^{-2t\pi^2} sin(\pi x) sin(\pi y). \\ u_y &= e^{-2t\pi^2} \pi sin(\pi x) cos(\pi y). \\ u_{yy} &= -\pi^2 e^{-2t\pi^2} sin(\pi x) sin(\pi y). \end{aligned} \tag{5.2}$$

og bestemmer $a_1 = a_2 = b_1 = b_2 = 1$. Settes informasjon om de partiellderiverte inn i (4.36) er problemet ved:

$$f(x,y,t) = e^{-2t\pi^2} \pi (cos(\pi x) sin(\pi y) + sin(\pi x) cos(\pi y)). \tag{5.3}$$

5.2 Stabilitetskrav oppsummert

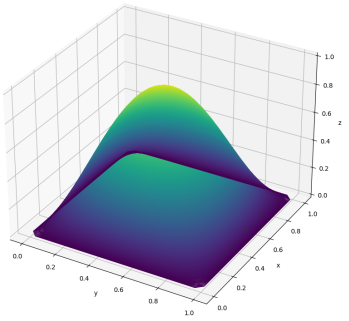
Stabilitetskrav spiller en viktig rolle, spesielt i situasjoner hvor hastighet er prioritert over presisjon.

Skjema	Krav for h	Krav for Δt
Forlengs Euler	$\min\{\frac{a_1}{\lvert b_1\rvert}, \frac{a_2}{\lvert b_2\rvert}\}$	$\frac{h^2}{2(a_1+a_2)+h\lvert b_1+b_2\rvert}$
Crank-Nicolson	$\min\{\frac{2a_1}{\lvert b_1\rvert}, \frac{2a_2}{\lvert b_2\rvert}\}$	ingen krav. ($\frac{h^2}{a_1+a_2}$ for positive parametre)
FEM	$\min\left\{\frac{3a_1}{\lvert b_2\rvert}, \frac{3a_2}{\lvert b_1\rvert}, \frac{2(a_1+a_2)}{\lvert b_1+b_2\rvert}\right\}$	$\frac{h^2}{6(a_1+a_2)}$
FEM med ML	$\min\{\frac{2a_1}{\lvert b_1\rvert}, \frac{2a_2}{\lvert b_2\rvert}\}$	$min\{\frac{h^2}{4a_1}, \frac{h^2}{4a_2}\}$

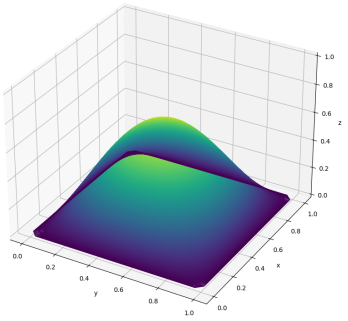
Krav til h er dobbelt så strengt for forlengs Euler sammenliknet med CN og FEM med ML. For FEM gir optimale valg av parametre det snilleste kravet, og uheldige valg gir det strengeste kravet. Forlengs Euler skiller seg ut i krav for diskretisering i tid ved å avhenge av konveksjonsleddene. Skal vi tro minstekravene til stabilitet bør Crank-Nicolson gi resultater med minst feil i forhold til den analytiske løsningen.

5.3 Sammenlikning og feilestimater

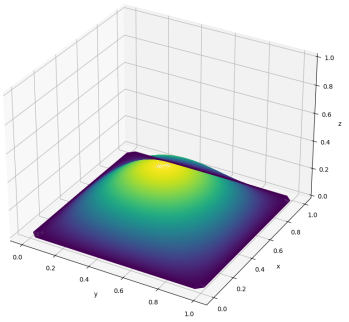
Sammenlikningen av skjemaene er delt i tre deler. I del 1 er h konstant, og Δt variabel. I del 2 er Δt konstant, og h variabel.I del 3 varierer h, og Δt proporsjonalt. Det er valgt tre sluttider, $T = 0.005$, $T = 0.02$, og $T = 0.05$:



(a) T=0.005



(b) T=0.02



(c) T=0.05

Figur 5.1: Analytisk løsning av problemet

For utregning av feilen i skjemaene brukes en l^2-norm av differansen mellom de numeriske løsningene, og den analytiske.En mer rettferdig sammenlikning av feil/node forsøkes oppnådd ved å dele l^2-normen på kvadratet av antall noder i griddet:

$$E_{u_h} = \sqrt{\frac{\sum_{i,j=1}^{N_1,N_2}(u-u_h)^2}{(N_1-2)(N_1-2)}}. \tag{5.4}$$

u_h er den numeriske løsningen i en node, og u den analytiske løsningen i tilsvarende node. Oversikt over størst feil ligger i vedlegg A.3. E_{u_h} blir videre bli betegnet som $absolutt\ feil$.

Konstant inndeling i rom:

$h = \frac{1}{15}$ gir absolutt feil:

T=0.005, absolutt feil $\times 10^{-2}$			
Tidssteg	$\Delta t = \frac{1}{2800}$	$\Delta t = \frac{1}{5000}$	$\Delta t = \frac{1}{10000}$
Forlengs Euler	0.648424	0.267775	0.049407
Crank-Nicolson	0.322759	0.173527	0.078455
FEM	0.070672	0.06683	0.065143
FEM med ML	0.933522	0.891804	0.878784
T=0.02, absolutt feil $\times 10^{-2}$			
Tidssteg	$\Delta t = \frac{1}{2800}$	$\Delta t = \frac{1}{5000}$	$\Delta t = \frac{1}{10000}$
Forlengs Euler	2.004879	0.818314	0.14059
Crank-Nicolson	0.202931	0.092918	0.027686
FEM	0.152689	0.150774	0.149602
FEM med ML	2.071976	2.060653	2.05787
T=0.05, absolutt feil $\times 10^{-2}$			
Tidssteg	$\Delta t = \frac{1}{2800}$	$\Delta t = \frac{1}{5000}$	$\Delta t = \frac{1}{10000}$
Forlengs Euler	2.659892	1.104384	0.176057
Crank-Nicolson	0.071935	0.023934	0.038976
FEM	0.143231	0.141836	0.140865
FEM med ML	1.967172	1.962967	1.962517

Med unntak av $T = 0.05$ for CN gir alle skjemaene bedre approksimasjoner ved finere diskretisering. For FEM, og FEM med ML er absolutt feil stabil med marginale forbedringer ved halvering av tidssteg. For forlengs Euler og CN gir finere inndeling mye bedre approksimasjoner. CN gir for raskest konvergens for tidsutvikling i de første to kolonnene. Konvergensen ved tidsutvikling er tregest for forlengs Euler, hvor (5.4) vokser i verdi. At (5.4) vokser betyr ikke at feilleddene fra steg til steg vokser, men at summen av feilleddene opp til den aktuelle tiden øker. Tidsutviklingen til feilen foreslår FEM med ML som et godt alternativ til forlengs Euler hvis tidsperioden vi modellerer er stor. forholdet mellom feilen i tidsutvikling for FEM og CN bytter om fra T=0.005 til T=0.02, og fra T=0.02 til T=0.005. Det er mulig dette skjer fordi måten de to skjemaene konvergerer ulikt.

Konstant inndeling i tid:

$\Delta t = \frac{1}{10000}$ gir absolutt feil:

T=0.005, absolutt feil $\times 10^{-2}$			
Inndeling	$h = \frac{1}{8}$	$h = \frac{1}{16}$	$= \frac{1}{24}$
Forlengs Euler	0.323933	0.08667	0.478046
Crank-Nicolson	0.044048	0.080155	0.086615
FEM	0.127028	0.060553	0.03856
FEM med ML	0.875692	0.878113	0.872062
T=0.02, absolutt feil $\times 10^{-2}$			
Inndeling	$h = \frac{1}{8}$	$h = \frac{1}{16}$	$= \frac{1}{24}$
Forlengs Euler	0.968379	0.258573	1.420553
Crank-Nicolson	0.134886	0.031029	0.050423
FEM	0.292786	0.139087	0.08873
FEM med ML	2.158787	2.052141	2.022612

T=0.05, absolutt feil $\times 10^{-2}$			
Inndeling	$h = \frac{1}{8}$	$h = \frac{1}{16}$	$= \frac{1}{24}$
Forlengs Euler	1.361122	0.345019	1.898329
Crank-Nicolson	0.238717	0.030428	0.014126
FEM	0.272584	0.131013	0.083716
FEM med ML	2.169565	1.954132	1.916476

Finere inndeling i rom gir raskest konvergens for FEM ved T=0.005 og T=0.02. Vi ser spesielt at FEM beskriver initialbetingelsene godt ved finere inndeling i rom, og har mer nøyaktige approksimasjoner enn CN for korte tidsperioder. CN ser ut til å ha en grense for inndeling i rom, hvor konvergensen blir tregere enn økningen av noder. Forlengs Euler viser igjen en tregere konvergens fremover i tid sammenliknet med de tre andre skjemaene. Forlengs Euler konvergerer nå også tregt for mindre h, hvor gennomsnitlig feil/Node øker. Vedlegg A.3 viser at den største feilen til forlengs Euler stiger, både frem i tid og for finere h. Tabellen for konstant diskretisering av tid foreslår bruk av en FEM metode i korte tidsperioder for liten h. For lengre tidsutvikling har CN raskest konvergens

Proporsjonalt forhold:

I denne seksjonen blir det bare testet for $T = 0.02$:

T=0.02, absolutt feil $\times 10^{-2}$			
Tidssteg	$h = \frac{1}{6}\ \Delta t = \frac{1}{2500}$	$h = \frac{1}{12}\ \Delta t = \frac{1}{5000}$	$h = \frac{1}{18}\ \Delta t = \frac{1}{7500}$
Forlengs Euler	1.287751	0.196809	0.851282
Crank-Nicolson	0.132889	0.07004	0.060103
FEM	0.449606	0.195197	0.122115
FEM med ML	2.223326	2.080271	2.043731

FEM ser ut til å konvergere raskere enn CN ved proposjonal reduksjon av h, og Δt. Det tyder på at vi har to kategorier av skjemaer. Forlengs Euler og FEM med ML gir effektivitet, mens CN og FEM har mer krevende utregninger og konkurrerer på nøyaktighet. Ingen klar vinner er avdekket i noen av kategoriene.

5.4 Avsluttende tanker og diskusjon

I løpet av denne oppgaven har vi verktøy for å approksimere differensialproblemer blitt utforsket. Ved å fokusere på et spesifikt problem (4.1) har vi sett at ulike metoder kan gi stabile løsninger. Et finite element skjema har fått spesiell oppmerksomhet. I behandling av stivhetsmatrisen er det funnet krav som sikrer at den er en M-matrise. En slik stivhetsmatrise er nyttig for andre typer problemer, som elliptiske differensiallikninger, men har den uheldige egenskapen av å begrense valg av parametre. Krav for tidsdiskretisering i FEM er bestemt til halvparten av kravet som gir $\mathbf{M_{ii}} \geq \mathbf{K_{ii}}$, og positiv høyreside i skjemaet. Det er ikke fastslått om forholdet mellom tidsinndelingen for positiv høyreside og stabilitet er tilfeldig. At massematrisen $\mathbf{M}$ bli anvendt to ganger i løsningen av (4.47) antyder at dette kan være grunnen til stabilitetskravet (4.71) istedet for (4.57). Hvis det kan vises at tilsvarende forhold mellom positiv høyreside og stabilitet gjelder for andre elementer enn de kvadratiske, vil det gi en effektivt metode for anslag av stabilitetskrav. Utregningene av feil i skjemaene gjør det

enkelt å se hvorfor Crank-Nicolson er populært som valg av numerisk skjema[6]. Ulempen med CN er at skjemaet kan være mer belastende å løse enn FEM, som også oppnår gode approksimasjoner i forhold til forlengs Euler. FEM er et mindre krevende skjema å løse fordi Cholesky-faktorisering halverer antall operasjoner i LU-faktoriseringen. Det ville vært interessant å videre se på en sentrering av FEM-skjemaet i tid. Hvis et sentrert FEM skjema har den samme effekten som å gå fra forlengs Euler til CN vil det være attraktivt både for å lette på stabilitetskravene, og som et neste steg av nøyaktighet. Det er vurdert positivitetsbevaring for alle numeriske skjemaer i kapittel 4, som en diskret analog til maksimumprinsippene utledet i kapittel 3. For FEM har vi ikke maksimumprinsipper på lik linje forlengs Euler og CN, og mass-lumping gir bare et maksimumprinsipp for et begrenset valg av parametre. Likevel har vi sett at von Neumann stabilitetsanalyse kan gi fungerende stabilitetskrav for metoder med endelige elementer, på lik linje som for endelige differanser. Hvis en sentrert versjon av FEM kan tilfredsstille (4.15) er dette enda en grunn til at et slikt skjema er av interesse.

Appendices

TILLEGG A

Vedlegg A

A.1 Basisfunksjon på randen

For lengder:

$$\int_0^s \int_0^b (\phi_{i,j})(\phi_{l,k})dxdy = \begin{cases} = \frac{h^2}{36} & l = i-1 \text{ eller } i+1 \text{ og } k = j-1 \text{ eller } j+1 \\ = 4\frac{h^2}{36} & l = i \text{ og } k = j-1 \text{ eller } j+1 \\ = 2\frac{h^2}{36} & l = i-1 \text{ eller } i+1 \text{ og } k = j \\ = 8\frac{h^2}{36} & l = i \text{ og } k = j \\ 0 \quad \text{ellers} \end{cases} \tag{A.1}$$

$$\int_0^s \int_0^b (\phi_{i,j})_x(\phi_{l,k})_x dxdy = \begin{cases} = -\frac{1}{6} & l = i-1 \text{ eller } i+1 \text{ og } k = j-1 \text{ eller } j+1 \\ = 2\frac{1}{6} & l = i \text{ og } k = j-1 \text{ eller } j+1 \\ = -2\frac{1}{6} & l = i-1 \text{ eller } i+1 \text{ og } k = j \\ = 4\frac{1}{6} & l = i \text{ og } k = j \\ 0 \quad \text{ellers} \end{cases} \tag{A.2}$$

$$\int_0^s \int_0^b (\phi_{i,j})_y(\phi_{l,k})_y dxdy = \begin{cases} = -\frac{1}{6} & l = i-1 \text{ eller } i+1 \text{ og } k = j-1 \text{ eller } j+1 \\ = -4\frac{1}{6} & l = i \text{ og } k = j-1 \text{ eller } j+1 \\ = \frac{1}{6} & l = i-1 \text{ eller } i+1 \text{ og } k = j \\ = 4\frac{1}{6} & l = i \text{ og } k = j \\ 0 \quad \text{ellers} \end{cases} \tag{A.3}$$

$$\int_0^s \int_0^b (\phi_{i,j})_x(\phi_{l,k}) dxdy = \begin{cases} = \frac{h}{12} & l = i-1 \text{ og } k = j-1 \text{ eller } j+1 \\ = -\frac{h}{12} & l = i+1 \text{ og } k = j-1 \text{ eller } j+1 \\ = 2\frac{h}{12} & l = i-1 \text{ og } k = j \\ = -2\frac{h}{12} & l = i+1 \text{ og } k = j \\ 0 \quad \text{ellers} \end{cases} \tag{A.4}$$

$$\int_0^s \int_0^b (\phi_{i,j})_y(\phi_{l,k})dxdy = \begin{cases} = \frac{h}{12} & l = i-1 \text{ eller } i+1 \text{ og } k = j-1 \\ = -\frac{h}{12} & l = i-1 \text{ eller } i+1 \text{ og } k = j+1 \\ = 4\frac{h}{12} & l = i \text{ og } k = j-1 \\ = -4\frac{h}{12} & l = i \text{ og } k = j+1 \\ = -4\frac{h}{12} & l = i \text{ og } k = 1, \text{Nede!} \\ = 4\frac{h}{12} & l = i \text{ og } k = 1, \text{Oppe!} \\ = -\frac{h}{12} & l = i-1 \text{ eller } i+1 \text{ og } k = j, \text{Nede!} \\ = \frac{h}{12} & l = i-1 \text{ eller } i+1 \text{ og } k = j, \text{Oppe!} \\ 0 \quad \text{ellers} \end{cases} \tag{A.5}$$

For høyder:

$$\int_0^s \int_0^b (\phi_{i,j})(\phi_{l,k})dxdy = \begin{cases} = \frac{h^2}{36} & l = i-1 \text{ eller } i+1 \text{ og } k = j-1 \text{ eller } j+1 \\ = 2\frac{h^2}{36} & l = i \text{ og } k = j-1 \text{ eller } j+1 \\ = 4\frac{h^2}{36} & l = i-1 \text{ eller } i+1 \text{ og } k = j \\ = 8\frac{h^2}{36} & l = i \text{ og } k = j \\ 0 \quad \text{ellers} \end{cases} \tag{A.6}$$

$$\int_0^s \int_0^b (\phi_{i,j})_x(\phi_{l,k})_x dxdy = \begin{cases} = -\frac{1}{6} & l = i-1 \text{ eller } i+1 \text{ og } k = j-1 \text{ eller } j+1 \\ = \frac{1}{6} & l = i \text{ og } k = j-1 \text{ eller } j+1 \\ = -4\frac{1}{6} & l = i-1 \text{ eller } i+1 \text{ og } k = j \\ = 4\frac{1}{6} & l = i \text{ og } k = j \\ 0 \quad \text{ellers} \end{cases} \tag{A.7}$$

$$\int_0^s \int_0^b (\phi_{i,j})_y(\phi_{l,k})_y dxdy = \begin{cases} = -\frac{1}{6} & l = i-1 \text{ eller } i+1 \text{ og } k = j-1 \text{ eller } j+1 \\ = -2\frac{1}{6} & l = i \text{ og } k = j-1 \text{ eller } j+1 \\ = 2\frac{1}{6} & l = i-1 \text{ eller } i+1 \text{ og } k = j \\ = 4\frac{1}{6} & l = i \text{ og } k = j \\ 0 \quad \text{ellers} \end{cases} \tag{A.8}$$

$$\int_0^s \int_0^b (\phi_{i,j})_x(\phi_{l,k}) dxdy = \begin{cases} = \frac{h}{12} & l = i-1 \text{ og } k = j-1 \text{ eller } j+1 \\ = -\frac{h}{12} & l = i+1 \text{ og } k = j-1 \text{ eller } j+1 \\ = 4\frac{h}{12} & l = i-1 \text{ og } k = j \\ = -4\frac{h}{12} & l = i+1 \text{ og } k = j \\ = 4\frac{h}{12} & l = i \text{ og } k = j, \text{Høyre!} \\ = -4\frac{h}{12} & l = i \text{ og } k = j, \text{Venstre!} \\ = 1\frac{h}{12} & l = i \text{ og } k = j-1 \text{ eller } j+1, \text{Høyre!} \\ = -1\frac{h}{12} & l = i \text{ og } k = j-1 \text{ eller } j+1, \text{Venstre} \\ 0 \quad \text{ellers} \end{cases} \tag{A.9}$$

$$\int_0^s \int_0^b (\phi_{i,j})_y(\phi_{l,k}) dxdy = \begin{cases} = \frac{h}{12} & l = i-1 \text{ eller } i+1 \text{ og } k = j-1 \\ = -\frac{h}{12} & l = i-1 \text{ eller } i+1 \text{ og } k = j+1 \\ = 2\frac{h}{12} & l = i \text{ og } k = j-1 \\ = -2\frac{h}{12} & l = i \text{ og } k = j+1 \\ 0 \quad \text{ellers} \end{cases} \tag{A.10}$$

For Hjørner!:

$$\int_0^s \int_0^b (\phi_{i,j})(\phi_{l,k})dxdy = \begin{cases} = \frac{h^2}{36} & l = i-1 \text{ eller } i+1 \text{ og } k = j-1 \text{ eller } j+1 \\ = 2\frac{h^2}{36} & l = i \text{ og } k = j-1 \text{ eller } j+1 \\ = 2\frac{h^2}{36} & l = i-1 \text{ eller } i+1 \text{ og } k = j \\ = 4\frac{h^2}{36} & l = i \text{ og } k = j \\ 0 \quad \text{ellers} \end{cases} \tag{A.11}$$

$$\int_0^s \int_0^b (\phi_{i,j})_x(\phi_{l,k})_x dxdy = \begin{cases} = -\frac{1}{6} & l = i-1 \text{ eller } i+1 \text{ og } k = j-1 \text{ eller } j+1 \\ = \frac{1}{6} & l = i \text{ og } k = j-1 \text{ eller } j+1 \\ = -2\frac{1}{6} & l = i-1 \text{ eller } i+1 \text{ og } k = j \\ = 2\frac{1}{6} & l = i \text{ og } k = j \\ 0 \quad \text{ellers} \end{cases} \tag{A.12}$$

$$\int_0^s \int_0^b (\phi_{i,j})_y(\phi_{l,k})_y dxdy = \begin{cases} = -\frac{1}{6} & l = i-1 \text{ eller } i+1 \text{ og } k = j-1 \text{ eller } j+1 \\ = -2\frac{1}{6} & l = i \text{ og } k = j-1 \text{ eller } j+1 \\ = \frac{1}{6} & l = i-1 \text{ eller } i+1 \text{ og } k = j \\ = 2\frac{1}{6} & l = i \text{ og } k = j \\ 0 \quad \text{ellers} \end{cases} \tag{A.13}$$

$$\int_0^s \int_0^b (\phi_{i,j})_x(\phi_{l,k})dxdy = \begin{cases} = \frac{h}{12} & l = i-1 \text{ og } k = j-1 \text{ eller } j+1 \\ = -\frac{h}{12} & l = i+1 \text{ og } k = j-1 \text{ eller } j+1 \\ = 2\frac{h}{12} & l = i-1 \text{ og } k = j \\ = -2\frac{h}{12} & l = i+1 \text{ og } k = j \\ = -2\frac{h}{12} & l = i \text{ og } k = j, \text{venstre hjørner!} \\ = 2\frac{h}{12} & l = i \text{ og } k = j, \text{høyre hjørner!} \\ = -\frac{h}{12} & l = i \text{ og } k = j-1 \text{ eller } j+1, \text{Venstre hjørner} \\ = +\frac{h}{12} & l = i \text{ og } k = j-1 \text{ eller } j+1, \text{Høyre hjørner} \\ 0 \quad \text{ellers} \end{cases} \tag{A.14}$$

$$\int_0^s \int_0^b (\phi_{i,j})_y(\phi_{l,k})dxdy = \begin{cases} = \frac{h}{12} & k = i-1 \text{ eller } i+1 \text{ og } r = j-1 \\ = -\frac{h}{12} & k = i-1 \text{ eller } i+1 \text{ og } r = j+1 \\ = 2\frac{h}{12} & k = i \text{ og } r = j-1 \\ = -2\frac{h}{12} & k = i \text{ og } r = j+1 \\ = -2\frac{h}{12} & k = i \text{ og } r = j, \text{nedre hjørner!} \\ = 2\frac{h}{12} & k = i \text{ og } r = j, \text{øvre hjørner!} \\ = -\frac{h}{12} & k = i-1 \text{ eller } i+1 \text{ og } r = j, \text{nedre hjørner!} \\ = \frac{h}{12} & k = i-1 \text{ eller } i+1 \text{ og } r = j, \text{øvre hjørner!} \\ 0 \quad \text{ellers} \end{cases} \tag{A.15}$$

A.2 Alternativt bevis for parabler

Beviset under har jeg med fordi jeg finner det elegant. Samtidig er jeg noe usikker på lovligheten til noen av argumentene underveis, og det kommer dermed som et supplement til beviset som allerede er ført:

Hvis vi velger et punkt (a,b) og en linje langs x-aksen kan vi sette opp:

$$|(x,y) - (a,b)| = |(x,y) - (st,0)|$$

$$\sqrt{x^2 - 2ax + a^2 + y^2 - 2by + b^2} = \sqrt{x^2 - 2sxt + s^2t^2 + y^2}$$

$$a^2 + b^2 - 2ax - 2by = (st)^2 - 2stx$$

For å få en normal fra linjen på x-aksen opp til punktet må vi ha $st = x$:

$$a^2 + b^2 - 2ax - 2by = -x^2$$

Herfra ser vi enkelt at den resulterende funksjonen kan skrives:

$$f(x) = a`x^2 + b`x + c`$$

Denne likningen er kjent og kjær, men vi kan bruke andre linjer enn den som går langs x-aksen. Ved innsetting av en vilkårlig linje har vi:

$$f(cx + fy + g) = a`(cx + fy + g)^2 + b`(cx + fy + g) + c` =$$

$$a`(c^2x^2 + 2cfxy + f^2y^2 + 2cgx + 2fgy + f^2) + b`(cx + fy + g) + c` = c`$$

Fra uttrykket får vi ut en samling av konstanter, førstegradsledd. og andregradsledd. For sistnevnte ser vi at det er en sammenheng mellom leddene:

$$c^2x^2 + 2cfxy + f^2y^2 \rightarrow Ax^2 + Bxy + Cy^2 = Ax^2 + 2\sqrt{A * C}xy + Cy^2$$

Resten er nå identisk med det eksisterende beviset.

A.3 Maks feil i skjemaer:

For $h = \frac{1}{15}$

T=0.005, størst feil $\times 10^{-2}$			
Tidssteg	$\Delta t = \frac{1}{2800}$	$\Delta t = \frac{1}{5000}$	$\Delta t = \frac{1}{10000}$
Forlengs Euler	0.405709	0.006436	0.006242
Crank-Nicolson	0.009414	0.003584	0.003584
FEM	0.417513	0.399148	0.380935
FEM med ML	0.398318	0.38399	0.374887

T=0.02, Største feil $\times 10^{-2}$			
Tidssteg	$\Delta t = \frac{1}{2800}$	$\Delta t = \frac{1}{5000}$	$\Delta t = \frac{1}{10000}$
Forlengs Euler	2.534967	1.082402	0.01548
Crank-Nicolson	0.004215	0.005939	0.007054
FEM	0.648946	0.634898	0.619884
FEM med ML	0.632814	0.622495	0.615923

T=0.05, største feil $\times 10^{-2}$			
Tidssteg	$\Delta t = \frac{1}{2800}$	$\Delta t = \frac{1}{5000}$	$\Delta t = \frac{1}{10000}$
Forlengs Euler	3.410005	1.893511	0.008624
Crank-Nicolson	0.008038	0.00811	0.008162
FEM	0.513517	0.505491	0.496955
FEM med ML	0.499135	0.493187	0.489396

For $\Delta t = \frac{1}{10000}$

T=0.005, størst feil $\times 10^{-2}$			
Inndeling	$h = \frac{1}{8}$	$h = \frac{1}{16}$	$= \frac{1}{24}$
Forlengs Euler	0.113772	0.00778	0.694712
Crank-Nicolson	0.016609	0.12866	0.00056
FEM	0.548266	0.346721	0.179493
FEM med ML	0.788319	0.340609	0.177001

T=0.02, størst feil $\times 10^{-2}$			
Inndeling	$h = \frac{1}{8}$	$h = \frac{1}{16}$	$= \frac{1}{24}$
Forlengs Euler	0.293102	0.006762	1.586192
Crank-Nicolson	0.077526	0.005643	0.001914
FEM	1.246555	0.555726	0.268463
FEM med ML	1.562665	0.551369	0.266527

T=0.05, størst feil $\times 10^{-2}$			
Inndeling	$h = \frac{1}{8}$	$h = \frac{1}{16}$	$= \frac{1}{24}$
Forlengs Euler	0.348582	0.404388	1.922696
Crank-Nicolson	0.090866	0.006468	0.001838
FEM	1.154625	0.442914	0.20855
FEM med ML	1.316466	0.436299	0.206631

Varierende h og Δt med konstant forhold:

T=0.02, største feil $\times 10^{-2}$			
Inndeling	$h = \frac{1}{6}$ $\Delta t = \frac{1}{2500}$	$h = \frac{1}{12}$ $\Delta t = \frac{1}{5000}$	$h = \frac{1}{18}$ $\Delta t = \frac{1}{7500}$
Forlengs Euler	0.711805	0.039134	1.318686
Crank-Nicolson	0.159496	0.012997	0.003818
FEM	2.268635	0.921183	0.455617
FEM med ML	2.087412	0.899308	0.450169

TILLEGG B

Vedlegg B

B.1 Kode til forlengs euler:

```
import numpy as np

"""
Likningen vi lÃÿser er fÃÿlgende:
du/dt +a_1(x,y,t)*du/dx^2 + a_1(x,y,t)*du/dy^2
  +b_1(x,y,t)*(du/dx)+b_2*(du/dy)=f(x,y,t)
Dette er varmelikningen med tillat transport i systemet, men uten
dannelse/tap av energi (gjennom f.eks kjemiske reaksjoner).
"""
""" Definerer omrÃĕde """
x_start=0 ; x_slutt=1    #rektangulÃęrt omrÃĕde.
y_start=0 ; y_slutt=1
T=0.02                  #sluttid
h=1/12
dt=1/2500
"""Visning av lÃÿsning"""
animasjon=0   #1=animasjon, 0= ikke animasjon
u_min=0  ;  u_max=1
K=15          # inndelinger i visning av tidsutviklingen,
              # visningen hopper flere steg av gangen for hver
              # inndeling.
oppdatering=0.3     # 1/fps
finhet_anim=200     #Ãÿker opplÃÿsning pÃĕ bilde, men koster
finhet_statisk=500  # en del prosessorkraft!

"""FUNKSJONER OG RANDBETINGELSER"""

a_1=1            #Diffusjonsledd 1=dxx, 2=dyy
a_2=1         ### Velg diffusjonsledd positive! ###

b_1=1              #transportledd 1=dx , 2=dy
b_2=1

def f(t,x,y):             #inhomogenitet
    del1=np.pi*np.exp(-2*t*np.pi**2)
    del2=(np.cos(np.pi*x)*np.sin(np.pi*y)+np.sin(np.pi*x)*np.cos(np.pi*y))
    return(del1*del2)
def g(x,y):                #starttilstand
    return(np.sin(np.pi*x)*np.sin(np.pi*y))

#Randbetingelser
def i_1(t,y):  #HÃÿyre
    return 0
```

```
def i_2(t,y):  #Venstre
    return 0
def i_3(t,x):  #Nede
    return 0
def i_4(t,x):  #Oppe
    return 0

###KODE UNDER DETTE PUNKTET TRENGER IKKE ENDRES###
# ved visning av U er matrisenummer du ulike tidene,
# kolonner er x, og rader er y.
"""korrigering av deltaer"""

epsilon=1/1000
h_sjekk=round(min(a_1/(abs(b_1)+epsilon),a_2/(abs(b_2)+epsilon)),15)
if h > h_sjekk:
    print("h er korrigert til",h)
    h=h_sjekk
dt_sjekk=(h**2)/(2*(a_1+a_2)+h*abs(b_1+b_2)+epsilon)
if dt > dt_sjekk:
    print("dt er korrigert til",dt)
    dt=dt_sjekk

""" Oppretter grid"""
M=int(T/dt +1)
N_1=int((x_slutt-x_start)/h +1)
N_2=int((y_slutt-y_start)/h +1)
U=np.zeros((N_1,N_2))
U_next=np.zeros((N_1,N_2))
F=np.zeros((N_1,N_2))

"""Setter inn randverdier."""
for k in range(0,N_2):
    for n in range(0,N_1):
        U[k,n]=(g(x_start+h*n,y_start+h*k))
        F[k,n]=f(0,x_start+n*h,y_start+k*h)
"""LÃ¸ser differensialproblemet"""
def u_next(i,j):
    d_1=U[i,j]*(1+(dt/(h**2))*(-2*(a_1+a_2)+h*(b_1+b_2)))
    d_2=U[i+1,j]*((dt/(h**2))*(a_1-h*b_1))
    d_3=U[i,j+1]*((dt/(h**2))*(a_2-h*b_2))
    d_4=U[i-1,j]*(a_1*dt/(h**2))
    d_5=U[i,j-1]*(a_2*dt/(h**2))
    d_6=F[i,j]*dt
    return(d_1+d_2+d_3+d_4+d_5+d_6)

"""oppretter plot , grenser og type plot"""
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(111, projection='3d')
z_nedre=u_min ; z_ovre=u_max
x_axis=(np.linspace(x_start,x_slutt,N_1))
y_axis=(np.linspace(y_start,y_slutt,N_2))
X,Y=(np.meshgrid(y_axis,x_axis))
ax.set_zlim(z_nedre,z_ovre)

if animasjon==0:
    for m in range(1,M):
        for k in range(1,N_1-1):
            for s in range(1,N_2-1):
                U_next[k,s]=u_next(k,s)
        for n in range(0,N_1):
            U[n,0]=i_3(m*dt,x_start+n*h)
            U[n,N_2-1]=i_4(m*dt,x_start+n*h,)
```

```
        for k in range(1,N_2-1):
            U[0,k]=i_2(m*dt,y_start+k*h)
            U[N_1-1,k]=i_1(m*dt,y_start+k*h)
        for n in range(0,N_1):
            for k in range(0,N_2):
                F[n,k]=f(m*dt,x_start+n*h,y_start+k*h)
        U=U_next
    np.savetxt("U_differanser",U)
    ax = fig.add_subplot(111, projection='3d')
    ax.set_xlabel('x')
    ax.set_ylabel('y')
    ax.set_zlabel('z')
    ax.set_zlim(z_nedre,z_ovre)
    ax.contour(X,Y,U,finhet_statisk)

sumtid=0
if animasjon==1:
    for t in range(1,K-1):
        ax.clear()
        tid=round((0.02*M/(1.02**K -1))*1.02**t)
        sumtid+=tid
        for m in range(0,tid):
            for k in range(1,N_1-1):
                for s in range(1,N_2-1):
                    U_next[k,s]=u_next(k,s)
            for n in range(0,N_1):
                U[n,0]=i_3(m*dt,x_start+n*h)
                U[n,N_2-1]=i_4(m*dt,x_start+n*h,)
            for k in range(1,N_2-1):
                U[0,k]=i_2(m*dt,y_start+k*h)
                U[N_1-1,k]=i_1(m*dt,y_start+k*h)
            for k in range(0,N_2):
                for n in range(0,N_1):
                    F[k,n]=f((m+sumtid)*dt,x_start+n*h,y_start+k*h)
            U=U_next
        ax = fig.add_subplot(111, projection='3d')
        ax.set_xlabel('x')
        ax.set_ylabel('y')
        ax.set_zlabel('z')
        ax.set_zlim(z_nedre,z_ovre)
        ax.contour(X,Y,U,finhet_anim)
        plt.pause(oppdatering)
plt.show()
```

B.2 Crank Nicolson-skjema

```
import matplotlib.pyplot as plt
import numpy as np
from scipy import sparse
import scipy

"""
Likningen vi lÃÿser er fÃÿlgende:
du/dt +a_1(x,y,t)*du/dx^2 + a_1(x,y,t)*du/dy^2
  +b_1(x,y,t)*(du/dx)+b_2*(du/dy)=f(x,y,t)
Dette er varmelikningen med tillat transport i systemet, men uten
dannelse/tap av energi (gjennom f.eks kjemiske reaksjoner).

"""
""" Definerer omrÃĕde """
```

```
x_start=0 ; x_slutt=1    #rektangulÃęrt omrÃĕde.
y_start=0 ; y_slutt=1
T=0.02                  #sluttid
h=1/12
dt=1/2500

"""Visning av lÃÿsning"""
u_min=0  ;  u_max=1
K=15          # inndelinger i visning av tidsutviklingen,
              # visningen hopper flere steg av gangen for hver
              # inndeling.
finhet_statisk=500  # en del prosessorkraft!

"""FUNKSJONER OG RANDBETINGELSER"""
a_1=1          #Diffusjonsledd
a_2=1

b_1=1          #transportledd
b_2=1

def f(t,x,y):             #inhomogenitet
    del1=np.pi*np.exp(-2*t*np.pi**2)
    del2=(np.cos(np.pi*x)*np.sin(np.pi*y)+np.sin(np.pi*x)*np.cos(np.pi*y))
    return(del1*del2)
def g(x,y):                #starttilstand
    return(np.sin(np.pi*x)*np.sin(np.pi*y))



"""
    if x>=1/4 and y>=1/4:
        if x<=3/4 and y<=3/4:
            return(1)
        else:
            return(0)
    else:
        return(0)
"""
#Randbetingelser
def i_1(t,y):  #HÃÿyre
    return 0
def i_2(t,y):  #Venstre
    return 0
def i_3(t,x):  #Nede
    return 0
def i_4(t,x):  #Oppe
    return 0

###KODE UNDER DETTE PUNKTET TRENGER IKKE ENDRES###
# ved visning av U er matrisenummer du ulike tidene,
# kolonner er x, og rader er y.
"""korrigering av deltaer"""

epsilon=1/1000
h_sjekk=round(min(2*a_1/(abs(b_1)+epsilon),2*a_2/(abs(b_2)+epsilon)),15)
if h > h_sjekk:
    print("h er korrigert til",h)
    h=h_sjekk
dt_sjekk=(h**2)/(a_1+a_2+epsilon)
if dt > dt_sjekk:
    print("dt er korrigert til",dt)
    dt=dt_sjekk
```

```
""" Oppretter grid"""
M=int(T/dt +1)
N_11=int((x_slutt-x_start)/h +1)
N_22=int((y_slutt-y_start)/h +1)
U_now=np.zeros((N_11,N_22))
F=np.zeros((N_11,N_22))

"""Setter inn randverdier."""
for k in range(0,N_22):
    for n in range(0,N_11):
        U_now[k,n]=(g(x_start+h*n,y_start+h*k))
        F[k,n]=f(0.5*dt,x_start+n*h,y_start+k*h)
"""LÃ¸ser differensialproblemet"""

v_u=np.zeros((2,N_11*N_22-N_11))
v_m1=np.zeros((2,N_11*N_22-1))
v_0=np.zeros((2,N_11*N_22))
v_p1=np.zeros((2,N_11*N_22-1))
v_o=np.zeros((2,N_11*N_22-N_11))

sigma_1=dt*(2*a_1+h*b_1)/(4*h**2)
sigma_2=dt*(2*a_1-h*b_1)/(4*h**2)
omega_1=dt*(2*a_2+h*b_2)/(4*h**2)
omega_2=dt*(2*a_2-h*b_2)/(4*h**2)
alpha=dt*(a_1+a_2)/(h**2)

vy=1
for s in range(0,2):
    v_0[s,0]=vy
    v_0[s,N_11-1]=vy
    v_0[s,N_11*(N_22-1)]=vy
    v_0[s,N_11*N_22-1]=vy
    for k in range(1,N_11-1):
        v_0[s,k]=vy
        v_0[s,N_11*(N_22-1)+k]=vy
    for n in range(1,N_22-1):
        v_0[s,n*N_11]=vy
        v_0[s,(n+1)*N_11-1]=vy
for n in range(1,N_22-1):
    for k in range(1,N_11-1):
        v_0[0,n*N_11+k]=1+alpha
        v_0[1,n*N_11+k]=1-alpha
for n in range(1,N_22-1):
    for k in range(1,N_11-1):
        v_m1[0,n*N_11+k-1]=-sigma_1
        v_p1[0,n*N_11+k]=-sigma_2
        v_m1[1,n*N_11+k-1]=sigma_1
        v_p1[1,n*N_11+k]=sigma_2
for n in range(0,N_22-2):
    for k in range(1,N_11-1):
        v_u[0,n*N_11+k]=-omega_1
        v_o[0,(n+1)*N_11+k]=-omega_2
        v_u[1,n*N_11+k]=omega_1
        v_o[1,(n+1)*N_11+k]=omega_2

diag_v=[-N_11,-1,0,1,N_11]
K_venstre=sparse.diags([v_u[0,:],v_m1[0,:],v_0[0,:],v_p1[0,:],v_o[0,:]],diag_v)
K_hÃ¸yre=sparse.diags([v_u[1,:],v_m1[1,:],v_0[1,:],v_p1[1,:],v_o[1,:]],diag_v)


K_sparse_v=sparse.csc_matrix(K_venstre)
```

```
K_sparse_h=sparse.csc_matrix(K_hÃÿyre)
F_m=np.reshape(F, (N_11*N_22))
U_now=np.reshape(U_now, (N_11*N_22))
#Kv_inv=scipy.sparse.linalg.inv(K_venstre)
#oppretter plot , grenser og type plot
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(111, projection='3d')
z_nedre=u_min ; z_ovre=u_max
x_axis=(np.linspace(x_start,x_slutt,N_11))
y_axis=(np.linspace(y_start,y_slutt,N_22))
X,Y=(np.meshgrid(x_axis,y_axis))
ax.set_zlim(z_nedre,z_ovre)


for m in range(0,M):
    b=K_sparse_h.dot(U_now)+dt*F_m
    U_next=scipy.sparse.linalg.spsolve(K_sparse_v,b)
    for n in range(0,N_11):
        U_next[n]=0
        U_next[(N_22-1)*N_11+n]=0
    for k in range(0,N_22):
        U_next[k*N_11]=0
        U_next[((k+1)*N_11)-1]=0
        for n in range(0,N_11):
            F[k,n]=f((0.5+m)*dt,x_start+n*h,y_start+k*h)
        F_m=np.reshape(F, (N_11*N_22))
    U_now=U_next
Omformet=U_now.reshape(N_22, N_11)
np.savetxt("U_CN",Omformet)
ax = fig.add_subplot(111, projection='3d')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
ax.set_zlim(z_nedre,z_ovre)
ax.contour(X,Y,Omformet,finhet_statisk)

plt.show()
```

B.3 FEM-skjema, vanlig+mass lumped

```
# -*- coding: utf-8 -*-
import matplotlib.pyplot as plt
import numpy as np
from scipy import sparse
from sksparse.cholmod import cholesky

""" HUSK ÃĚ KJÃŸRE "oppsett FEM" FÃŸRST"""
kompansert=1  #Kompanserer for konstante kjente randverdier.
""" visning"""
u_min=0  ;  u_max=1
animasjon=0   #1=animasjon, 0= ikke animasjon
K=15          # inndelinger i visning av tidsutviklingen,
              # visningen hopper flere steg av gangen for hver
              # inndeling.
oppdatering=0.3     # 1/fps
finhet_anim=200     #Ãŷker opplÃŷsning pÃě bilde, men koster
finhet_statisk=500  # en del prosessorkraft!


""" slutt pÃě oppsett"""
```

```
"""FUNKSJONER, PARAMETRE OG RANDBETINGELSER"""
phi_master=np.loadtxt("phi_master")
R=np.loadtxt("Konstanter")
A_1=R[0] ; A_2=R[1]
B_1=R[2] ; B_2=R[3]
dt=R[4] ; h=R[5]
T=R[6]
x_start=R[7] ; x_slutt=R[8]
y_start=R[9] ; y_slutt=R[10]
mass_lumping=R[11]

U_now=np.loadtxt("U_now")
F=np.loadtxt("F")

M=int(T/dt +1)
N_11=int((x_slutt-x_start)/h +1)
N_22=int((y_slutt-y_start)/h +1)

### SPARSE MATRIXES###

#Oppretter diagonaler
v_um1=np.zeros((5,N_11*N_22-N_11-1))
v_u=np.zeros((5,N_11*N_22-N_11))
v_up1=np.zeros((5,N_11*N_22-N_11+1))
v_m1=np.zeros((5,N_11*N_22-1))
v_p1=np.zeros((5,N_11*N_22-1))
v_0=np.zeros((5,N_11*N_22))
v_om1=np.zeros((5,N_11*N_22-N_11+1))
v_o=np.zeros((5,N_11*N_22-N_11))
v_op1=np.zeros((5,N_11*N_22-N_11-1))

phi_master=phi_master.reshape(5,9,9)
phi_master[0,:,:]=phi_master[0,:,:]*h**2
phi_master[3,:,:]=phi_master[3,:,:]*h
phi_master[4,:,:]=phi_master[4,:,:]*h



if kompansert==1:
   vy=1
   v_0[0,0]=vy
   v_0[0,N_11-1]=vy
   v_0[0,N_11*(N_22-1)]=vy
   v_0[0,N_11*N_22-1]=vy
   for k in range(1,N_11-1):
       v_0[0,k]=vy
       v_0[0,N_11*(N_22-1)+k]=vy
   for n in range(1,N_22-1):
       v_0[0,n*N_11]=vy
       v_0[0,(n+1)*N_11-1]=vy
   for s in range(0,5):
       for n in range(1,N_22-1):
           for k in range(1,N_11-1):
               v_0[s,n*N_11+k]=phi_master[s,4,4]

       for n in range(1,N_22-1):
           for k in range(1,N_11-1):
               v_p1[s,n*N_11+k]=phi_master[s,3,4]
               v_m1[s,n*N_11+k-1]=phi_master[s,4,3]
####
       for n in range(0,N_22-2):
           for k in range(1,N_11-1):
```

```
                v_om1[s,(n+1)*N_11+k]=phi_master[s,1,3]
                v_up1[s,n*N_11+k+1]=phi_master[s,3,1]
                v_u[s,n*N_11+k]=phi_master[s,4,1]
                v_o[s,(n+1)*N_11+k]=phi_master[s,1,4]
        for n in range(0,N_22-2):
            for k in range(0,N_11-2):
                v_op1[s,(n+1)*N_11+1+k]=phi_master[s,0,4]
                v_um1[s,n*N_11+k]=phi_master[s,4,0]

if kompansert==0:
   for s in range(0,5):
        v_0[s,0]=phi_master[s,0,0]
        v_0[s,N_11-1]=phi_master[s,2,2]
        v_0[s,N_11*(N_22-1)]=phi_master[s,6,6]
        v_0[s,N_11*N_22-1]=phi_master[s,8,8]
        for k in range(1,N_11-1):
            v_0[s,k]=phi_master[s,1,1]
            v_0[s,N_11*(N_22-1)+k]=phi_master[s,7,7]
        for n in range(1,N_22-1):
            v_0[s,n*N_11]=phi_master[s,3,3]
            v_0[s,(n+1)*N_11-1]=phi_master[s,5,5]
            for k in range(1,N_11-1):
                v_0[s,n*N_11+k]=phi_master[s,4,4]

        for k in range(0,N_11-1):
            v_p1[s,k]=phi_master[s,0,1]
            v_m1[s,k]=phi_master[s,1,0]
            v_p1[s,(N_22-1)*N_11+k]=phi_master[s,6,7]
            v_m1[s,(N_22-1)*N_11+k]=phi_master[s,7,6]
        for n in range(1,N_22-1):
            for k in range(0,N_11-1):
                v_p1[s,n*N_11+k]=phi_master[s,3,4]
                v_m1[s,n*N_11+k]=phi_master[s,4,3]

        for n in range(0,N_22-1):
            for k in range(1,N_11):
                v_om1[s,n*N_11+k]=phi_master[s,1,3]
                v_up1[s,n*N_11+k]=phi_master[s,3,1]

        for n in range(0,N_22-1):
            v_o[s,n*N_11]=phi_master[s,0,3]
            v_u[s,n*N_11]=phi_master[s,3,0]
            v_o[s,(n+1)*N_11-1]=phi_master[s,2,5]
            v_u[s,(n+1)*N_11-1]=phi_master[s,5,2]
            for k in range(1,N_11-1):
                v_o[s,n*N_11+k]=phi_master[s,1,4]
                v_u[s,n*N_11+k]=phi_master[s,4,1]

        for n in range(0,N_22-1):
            for k in range(0,N_11-1):
                v_op1[s,n*N_11+k]=phi_master[s,0,4]
                v_um1[s,n*N_11+k]=phi_master[s,4,0]

diag_v=[-(N_11+1),-N_11,-(N_11-1),-1,0,1,N_11-1,N_11,N_11+1]


M_sparse_pp=sparse.diags([v_um1[0,:],v_u[0,:],v_up1[0,:],v_m1[0,:],v_0[0,:]
                                ,v_p1[0,:],v_om1[0,:],v_o[0,:],v_op1[0,:]],diag_v)
M_sparse_pxpx=sparse.diags([v_um1[1,:],v_u[1,:],v_up1[1,:],v_m1[1,:],v_0[1,:]
                                ,v_p1[1,:],v_om1[1,:],v_o[1,:],v_op1[1,:]],diag_v)
M_sparse_pypy=sparse.diags([v_um1[2,:],v_u[2,:],v_up1[2,:],v_m1[2,:],v_0[2,:]
                                ,v_p1[2,:],v_om1[2,:],v_o[2,:],v_op1[2,:]],diag_v)
```

```
M_sparse_pxp=sparse.diags([v_um1[3,:],v_u[3,:],v_up1[3,:],v_m1[3,:],v_0[3,:]
                                ,v_p1[3,:],v_om1[3,:],v_o[3,:],v_op1[3,:]],diag_v)
M_sparse_pyp=sparse.diags([v_um1[4,:],v_u[4,:],v_up1[4,:],v_m1[4,:],v_0[4,:]
                                ,v_p1[4,:],v_om1[4,:],v_o[4,:],v_op1[4,:]],diag_v)

#plt.spy
#pp_inv=scipy.sparse.linalg.inv(M_sparse_pp)
#Om du gidder sett av tid til Ãě bygge disse fra bunn
M_sparse_pp=sparse.csc_matrix(M_sparse_pp)
M_sparse_pxpx=sparse.csc_matrix(M_sparse_pxpx)
M_sparse_pypy=sparse.csc_matrix(M_sparse_pypy)
M_sparse_pxp=sparse.csc_matrix(M_sparse_pxp)
M_sparse_pyp=sparse.csc_matrix(M_sparse_pyp)

###Grid til plot:###
x_axis=(np.linspace(x_start,x_slutt,N_11))
y_axis=(np.linspace(y_start,y_slutt,N_22))
X,Y=(np.meshgrid(x_axis,y_axis))
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(111, projection='3d')
z_nedre=u_min ; z_ovre=u_max
ax.set_zlim(z_nedre,z_ovre)

Ledd=A_1*M_sparse_pxpx+A_2*M_sparse_pypy+B_1*M_sparse_pxp+B_2*M_sparse_pyp
#Valg av metode:
def metode(y,x,z):
    if mass_lumping==1:
        return(z/h**2)
    if mass_lumping==0:
        return(y(x))
if animasjon==0:
    for m in range(0,M):
        b=M_sparse_pp.dot(U_now)-dt*(Ledd.dot(U_now))+dt*F[m,:]
        c=h**2*U_now-dt*(Ledd.dot(U_now))+dt*F[m,:]
        faktor = cholesky(M_sparse_pp)
        U_now=metode(faktor,b,c)
        for n in range(0,N_11):
            U_now[n]=0
            U_now[(N_22-1)*N_11+n]=0
        for k in range(0,N_22):
            U_now[k*N_11]=0
            U_now[((k+1)*N_11)-1]=0
    Omformet=U_now.reshape(N_11, N_22)

    if mass_lumping==0:
       np.savetxt("U_FEM",Omformet)
    if mass_lumping==1:
       np.savetxt("U_FEM_ML",Omformet)
    ax = fig.add_subplot(111, projection='3d')
    ax.set_xlabel('x')
    ax.set_ylabel('y')
    ax.set_zlabel('z')
    ax.set_zlim(z_nedre,z_ovre)
    ax.contour(X,Y,Omformet,finhet_statisk)
if animasjon==1:
    Omformet=np.frombuffer(U_now).reshape(N_11, N_22)
    for t in range(1,K):
        ax.clear()
        tid=round((0.02*M/(1.02**K -1))*1.02**t)
        for m in range(0,tid):
            b=M_sparse_pp.dot(U_now)-dt*(Ledd.dot(U_now))+dt*F[m,:]
            c=h**2*U_now-dt*(Ledd.dot(U_now))+dt*F[m,:]
```

```
            faktor = cholesky(M_sparse_pp)
            U_now=metode(faktor,b,c)
            for n in range(0,N_11):
                U_now[n]=0
                U_now[(N_22-1)*N_11+n]=0
            for k in range(0,N_22):
                U_now[k*N_11]=0
                U_now[((k+1)*N_11)-1]=0
        Omformet=np.frombuffer(U_now).reshape(N_22, N_11)
        ax = fig.add_subplot(111, projection='3d')
        ax.set_xlabel('x')
        ax.set_ylabel('y')
        ax.set_zlabel('z')
        ax.set_zlim(z_nedre,z_ovre)
        ax.contour(X,Y,Omformet,finhet_anim)
        plt.pause(oppdatering)

plt.show()
```

B.4 Oppsett til FEM-skjema

```
# -*- coding: utf-8 -*-
import numpy as np

"""FUNKSJONER OG RANDBETINGELSER"""
x_start=0 ; x_slutt=1    #rektangulÃęrt omrÃëde.
y_start=0 ; y_slutt=1
T=0.05                  #sluttid
h=1/15
dt=1/10000

A_1=1            #Diffusjonsledd 1=dxx, 2=dyy
A_2=1         ### Velg diffusjonsledd positive! ###

B_1=1              #transportledd 1=dx , 2=dy
B_2=1
def f(t,x,y):             #inhomogenitet
    del1=np.pi*np.exp(-2*t*np.pi**2)
    del2=(np.cos(np.pi*x)*np.sin(np.pi*y)+np.sin(np.pi*x)*np.cos(np.pi*y))
    return(del1*del2)
def g(x,y):                #starttilstand
    return(np.sin(np.pi*x)*np.sin(np.pi*y))


"""starterr hjelper til Ãë finne krav som begrenser stÃÿrste
mulige feil av den linÃęrt interpolerte versjonen
av g(x,y) i forhold til den kontinuerlige g(x,y)
 Velg starterr=0 for Ãë ignorere denne delen av koden."""
starterr=0

"""mass_lumping bestemmer om du skal lÃÿse FEM med
standard metode (=0) , eller med mass-lumped
massematrise (=1)"""
mass_lumping=0

"""Koden er ikke moden for inhomogene randfunksjoner enda
   behold disse funksjonene som 0"""
def i_1(t,y):  #HÃÿyre
    return 0
def i_2(t,y):  #Venstre
```

```
    return 0
def i_3(t,x):  #Nede
    return 0
def i_4(t,x):  #Oppe
    return 0
""" slutt pÃě oppsett"""

"""korrigering og test av parametre"""
if A_1/2 > A_2:
   print("K er ikke M-matrise, programmet kan fortsatt kjÃÿres")
if A_1 > 2*A_2:
   print("K er ikke M-matrise, programmet kan fortsatt kjÃÿres")
epsilon=1/10000
if B_1==B_2==0:
    test_1=1
else:
    if mass_lumping==0:
        test_1=min(3*A_1/(abs(B_2)+epsilon),3*A_2/(abs(B_1)+epsilon),2*(A_1+A_2)/(abs(B_1+B_2)+epsilon))
    if mass_lumping==1:
        test_1=min(2*A_1/(abs(B_2)+epsilon),2*A_2/(abs(B_1)+epsilon))
if h > test_1:
    print("h er korrigert til",test_1)
    h=test_1
###Test for dt###
if mass_lumping==0:
   test_2=h**2 /(6*(A_1+A_2)+epsilon)
if mass_lumping==1:
   test_2=min((h**2) /(4*A_1+epsilon),(h**2) /(4*A_2+epsilon))
if dt > test_2:
    print("dt er korrigert til",test_2)
    dt=test_2

M=int(T/dt +1)
N_11=int((x_slutt-x_start)/h +1)
N_22=int((y_slutt-y_start)/h +1)
U_now=np.zeros(N_11*N_22)
F=np.zeros((M,N_11*N_22))

"""Setter inn rand og funksjonsverdier."""
for k in range(1,N_22):
    for n in range(1,N_11):
        U_now[k*N_11+n]=(g(x_start+h*n,y_start+h*k))
for n in range(0,N_11):
    U_now[n]=i_3(0,x_start+n*h)
    U_now[(N_22-1)*N_11+n]=i_4(0,x_start+n*h,)
for k in range(0,N_22):
    U_now[k*N_11]=i_2(0,y_start+k*h)
    U_now[((k+1)*N_11)-1]=i_1(0,y_start+k*h)
"Numerisk integrasjon av F:"
#Vekttall:
w_1=25/81 ; w_2=40/81 ; w_3=64/81
# Posisjoner:
first=h*(5-np.sqrt(15))/10
second=h/2
third=h*(5+np.sqrt(15))/10
# Verdier thil phi:
p_1=(1-((5-np.sqrt(15))/10))*(1-((5-np.sqrt(15))/10))
p_2=p_4=(1/2)*(1-((5-np.sqrt(15))/10))
p_3=p_7=(1-((5+np.sqrt(15))/10))*(1-((5-np.sqrt(15))/10))
p_5=(1/2)*(1/2)
p_6=p_8=(1/2)*(1-((5+np.sqrt(15))/10))
p_9=(1-((5+np.sqrt(15))/10))*(1-((5+np.sqrt(15))/10))
```

```
### HjÃ¿rner

for t in range(0,M):
    #nede venstre:
    a_1=w_1*(f(t,first,first)*p_1+f(t,third,first)*p_3+f(t,first,third)*p_7
             +f(t,third,third)*p_9)
    b_1=w_2*(f(t,second,first)*p_2+f(t,second,third)*p_8+f(t,first,second)*p_4
             +f(t,third,second)*p_6)
    c_1=w_3*f(t,second,second)*p_5
    F[t,0]=(h**2 / 4)*(a_1+b_1+c_1)
    #nede hÃ¿yre:
    a_2=w_1*(f(t,h*(N_11-2)+first,first)*p_3+f(t,h*(N_11-2)+third,first)*p_1
             +f(t,h*(N_11-2)+first,third)*p_9+f(t,h*(N_11-2)+third,third)*p_7)
    b_2=w_2*(f(t,h*(N_11-2)+second,first)*p_2+f(t,h*(N_11-2)+second,third)*p_8
             +f(t,h*(N_11-2)+first,second)*p_6+f(t,h*(N_11-2)+third,second)*p_4)
    c_2=w_3*f(t,h*(N_11-2)+second,second)*p_5
    F[t,N_11-1]=(h**2 / 4)*(a_2+b_2+c_2)
    #oppe venstre:
    a_3=w_1*(f(t,first,h*(N_22-2)+first)*p_7+f(t,third,h*(N_22-2)+first)*p_9
             +f(t,first,h*(N_22-2)+third)*p_1+f(t,third,h*(N_22-2)+third)*p_3)
    b_3=w_2*(f(t,second,h*(N_22-2)+first)*p_8+f(t,second,h*(N_22-2)+third)*p_2
             +f(t,first,h*(N_22-2)+second)*p_4+f(t,third,h*(N_22-2)+second)*p_6)
    c_3=w_3*f(t,second,h*(N_22-2)+second)*p_5
    F[t,N_11*(N_22-1)]=(h**2 / 4)*(a_3+b_3+c_3)
    #oppe hÃ¿yre:
    a_4=w_1*(f(t,h*(N_11-2)+first,h*(N_22-2)+first)*p_9+f(t,h*(N_11-2)
            +third,h*(N_22-2)+first)*p_7+f(t,h*(N_11-2)
            +first,h*(N_22-2)+third)*p_3+f(t,h*(N_11-2)+third,h*(N_22-2)+third)*p_1)
    b_4=w_2*(f(t,h*(N_11-2)+second,h*(N_22-2)+first)*p_8+f(t,h*(N_11-2)+second,
            h*(N_22-2)+third)*p_2+f(t,h*(N_11-2)+first,h*(N_22-2)+second)*p_6
            +f(t,h*(N_11-2)+third,h*(N_22-2)+second)*p_4)
    c_4=w_3*f(t,h*(N_11-2)+second,h*(N_22-2)+second)*p_5
    F[t,N_11*N_22-1]=(h**2 / 4)*(a_4+b_4+c_4)

    #lengder:
    for n in range(1,N_11-1):
        a=w_1*(f(t,(n-1)*h+first,first)*p_3+f(t,(n-1)*h+third,first)*p_1
               +f(t,(n-1)*h+first,third)*p_9+f(t,(n-1)*h+third,third)*p_7)
        b=w_2*(f(t,(n-1)*h+second,first)*p_2+f(t,(n-1)*h+second,third)*p_8
               +f(t,(n-1)*h+first,second)*p_6+f(t,(n-1)*h+third,second)*p_4)
        c=w_3*(f(t,(n-1)*h+second,second)*p_5)
        d=w_1*(f(t,n*h+first,first)*p_1+f(t,n*h+third,first)*p_3
               +f(t,n*h+first,third)*p_7+f(t,n*h+third,third)*p_9)
        e=w_2*(f(t,n*h+second,first)*p_2+f(t,n*h+second,third)*p_8
               +f(t,n*h+first,second)*p_4+f(t,n*h+third,second)*p_6)
        g=w_3*(f(t,n*h+second,second)*p_5)
        F[t,n]=(h**2 / 4)*(a+b+c+d+e+g)
        a_1=w_1*(f(t,(n-1)*h+first,h*(N_22-2)+first)*p_9
                 +f(t,(n-1)*h+third,h*(N_22-2)+first)*p_7+f(t,(n-1)*h+first,
                 h*(N_22-2)+third)*p_3+f(t,(n-1)*h+third,h*(N_22-2)+third)*p_1)
        b_1=w_2*(f(t,(n-1)*h+second,h*(N_22-2)+first)*p_8+f(t,(n-1)*h+second,
                 h*(N_22-2)+third)*p_2+f(t,(n-1)*h+first,h*(N_22-2)+second)*p_6
                 +f(t,(n-1)*h+third,h*(N_22-2)+second)*p_4)
        c_1=w_3*(f(t,(n-1)*h+second,h*(N_22-2)+second)*p_5)
        d_1=w_1*(f(t,n*h+first,h*(N_22-2)+first)*p_7+f(t,n*h+third,
                 h*(N_22-2)+first)*p_9+f(t,n*h+first,h*(N_22-2)+third)*p_1
                 +f(t,n*h+third,h*(N_22-2)+third)*p_3)
        e_1=w_2*(f(t,n*h+second,h*(N_22-2)+first)*p_8+f(t,n*h+second,h*(N_22-2)
                 +third)*p_2+f(t,n*h+first,h*(N_22-2)+second)*p_4
                 +f(t,n*h+third,h*(N_22-2)+second)*p_6)
        g_1=w_3*(f(t,n*h+second,h*(N_22-2)+second)*p_5)
```

```
        F[t,N_11*(N_22-1)+n]=(h**2 / 4)*(a_1+b_1+c_1+d_1+e_1+g_1)
    #Bredder:
    for n in range(1,N_22-1):
        a=w_1*(f(t,first,(n-1)*h+first)*p_7+f(t,third,(n-1)*h+first)*p_9
               +f(t,first,(n-1)*h+third)*p_1+f(t,third,(n-1)*h+third)*p_3)
        b=w_2*(f(t,second,(n-1)*h+first)*p_8+f(t,second,(n-1)*h+third)*p_2
               +f(t,first,(n-1)*h+second)*p_4+f(t,third,(n-1)*h+second)*p_6)
        c=w_3*f(t,second,(n-1)*h+second)*p_5
        d=w_1*(f(t,first,n*h+first)*p_1+f(t,third,n*h+first)*p_3
               +f(t,first,n*h+third)*p_7+f(t,third,n*h+third)*p_9)
        e=w_2*(f(t,second,n*h+first)*p_2+f(t,second,n*h+third)*p_8
               +f(t,first,n*h+second)*p_4+f(t,third,n*h+second)*p_6)
        g=w_3*f(t,second,n*h+second)*p_5
        F[t,n*N_11]=(h**2 / 4)*(a+b+c+d+e+g)
        a_1=w_1*(f(t,(N_11-1)*h+first,(n-1)*h+first)*p_9+f(t,(N_11-1)*h+third,
                (n-1)*h+first)*p_7+f(t,(N_11-1)*h+first,(n-1)*h+third)*p_3
                +f(t,(N_11-1)*h+third,(n-1)*h+third)*p_1)
        b_1=w_2*(f(t,(N_11-1)*h+second,(n-1)*h+first)*p_8+f(t,(N_11-1)*h+second,
                (n-1)*h+third)*p_2+f(t,(N_11-1)*h+first,(n-1)*h+second)*p_6
                +f(t,(N_11-1)*h+third,(n-1)*h+second)*p_4)
        c_1=w_3*f(t,(N_11-1)*h+second,(n-1)*h+second)*p_5
        d_1=w_1*(f(t,(N_11-1)*h+first,n*h+first)*p_3+f(t,(N_11-1)*h+third,
                n*h+first)*p_1+f(t,(N_11-1)*h+first,n*h+third)*p_9
                +f(t,(N_11-1)*h+third,n*h+third)*p_7)
        e_1=w_2*(f(t,(N_11-1)*h+second,n*h+first)*p_2+f(t,(N_11-1)*h+second,
                n*h+third)*p_8+f(t,(N_11-1)*h+first,n*h+second)*p_6
                +f(t,(N_11-1)*h+third,n*h+second)*p_4)
        g_1=w_3*f(t,(N_11-1)*h+second,n*h+second)*p_5
        F[t,N_11*(n+1)-1]=(h**2 / 4)*(a_1+b_1+c_1+d_1+e_1+g_1)
    #Midtpunkter
    for m in range(1,N_22-1):
        for n in range(1,N_11-1):
            a_1=w_1*(f(t,(n-1)*h+first,(m-1)*h+first)*p_9+f(t,(n-1)*h+third,
                    (m-1)*h+first)*p_7+f(t,(n-1)*h+first,(m-1)*h+third)*p_3
                    +f(t,(n-1)*h+third,(m-1)*h+third)*p_1)
            b_1=w_2*(f(t,(n-1)*h+second,(m-1)*h+first)*p_8+f(t,(n-1)*h+second,
                    (m-1)*h+third)*p_2+f(t,(n-1)*h+first,(m-1)*h+second)*p_6
                    +f(t,(n-1)*h+third,(m-1)*h+second)*p_4)
            c_1=w_3*f(t,(n-1)*h+second,(m-1)*h+second)*p_5
            a_2=w_1*(f(t,n*h+first,(m-1)*h+first)*p_7+f(t,n*h+third,(m-1)*h
                    +first)*p_9+f(t,n*h+first,(m-1)*h+third)*p_1+f(t,n*h+third,
                    (m-1)*h+third)*p_3)
            b_2=w_2*(f(t,n*h+second,(m-1)*h+first)*p_8+f(t,n*h+second,(m-1)*h
                    +third)*p_2+f(t,n*h+first,(m-1)*h+second)*p_4+f(t,n*h+third,
                    (m-1)*h+second)*p_6)
            c_2=w_3*f(t,n*h+second,(m-1)*h+second)*p_5
            a_3=w_1*(f(t,(n-1)*h+first,m*h+first)*p_3+f(t,(n-1)*h+third,m*h+first)*p_1
                    +f(t,(n-1)*h+first,m*h+third)*p_9+f(t,(n-1)*h+third,m*h+third)*p_7)
            b_3=w_2*(f(t,(n-1)*h+second,m*h+first)*p_2+f(t,(n-1)*h+second,m*h+third)*p_8
                    +f(t,(n-1)*h+first,m*h+second)*p_6+f(t,(n-1)*h+third,m*h+second)*p_4)
            c_3=w_3*f(t,(n-1)*h+second,m*h+second)*p_5
            a_4=w_1*(f(t,n*h+first,m*h+first)*p_1+f(t,n*h+third,m*h+first)*p_3
                    +f(t,n*h+first,m*h+third)*p_7+f(t,n*h+third,m*h+third)*p_9)
            b_4=w_2*(f(t,n*h+second,m*h+first)*p_2+f(t,n*h+second,m*h+third)*p_8
                    +f(t,n*h+first,m*h+second)*p_4+f(t,n*h+third,m*h+second)*p_6)
            c_4=w_3*f(t,n*h+second,m*h+second)*p_5
            F[t,m*N_11+n]=(h**2 / 4)*(a_1+a_2+a_3+a_4+b_1+b_2+b_3+b_4+c_1+c_2+c_3+c_4)
###test for glatthet-supplementerende krav for gode approksimasjoner###

if starterr !=0:
   C=0
   for k in range(0,N_22-1):
```

```
       for n in range(0,N_11-1):
           utest=max(abs(U_now[k*N_11+n]-U_now[k*N_11+(n+1)]),
                  abs(U_now[k*N_11+n]-U_now[(k+1)*N_11+(n)]),
                  abs(U_now[k*N_11+n]-U_now[(k+1)*N_11+(n+1)]),
                  abs(U_now[k*N_11+(n+1)]-U_now[(k+1)*N_11+n]),
                  abs(U_now[k*N_11+(n+1)]-U_now[(k+1)*N_11+(n+1)]),
                  abs(U_now[(k+1)*N_11+n]-U_now[(k+1)*N_11+(n+1)]))
           if utest>C:
               C=utest
   if starterr < C*h**2:
       print("vurder en finere inndelinger av h")



av=30
N_1=3
N_2=3

phi_phi=np.zeros((N_1*N_2,N_1*N_2))
phiX_phiX=np.zeros((N_1*N_2,N_1*N_2))
phiY_phiY=np.zeros((N_1*N_2,N_1*N_2))
phiX_phi=np.zeros((N_1*N_2,N_1*N_2))
phiY_phi=np.zeros((N_1*N_2,N_1*N_2))


##HjÃ¥rner##
phi_phi[0,0]=round(4/36,av)
phi_phi[N_1-1,N_1-1]=round(4/36,av)
phi_phi[N_1*(N_2-1),N_1*(N_2-1)]=round(4/36,av)
phi_phi[N_1*N_2-1,N_1*N_2-1]=round(4/36,av)

phi_phi[1,0]=round(2/36,av)
phi_phi[N_1-2,N_1-1]=round(2/36,av)
phi_phi[N_1*(N_2-1)+1,N_1*(N_2-1)]=round(2/36,av)
phi_phi[N_1*N_2-2,N_1*N_2-1]=round(2/36,av)

phi_phi[N_1,0]=round(2/36,av)
phi_phi[N_1-1+N_1,N_1-1]=round(2/36,av)
phi_phi[N_1*(N_2-2),N_1*(N_2-1)]=round(2/36,av)
phi_phi[N_1*(N_2-1)-1,N_1*N_2-1]=round(2/36,av)

phi_phi[N_1+1,0]=round(1/36,av)
phi_phi[N_1-2+N_1,N_1-1]=round(1/36,av)
phi_phi[N_1*(N_2-2)+1,N_1*(N_2-1)]=round(1/36,av)
phi_phi[N_1*(N_2-1)-2,N_1*N_2-1]=round(1/36,av)
##Lengder##
for m in range(1,N_1-1):
    phi_phi[m,m]=round(8/36,av)
    phi_phi[m-1,m]=round(2/36,av)
    phi_phi[m+1,m]=round(2/36,av)
    phi_phi[m+N_1,m]=round(4/36,av)
    phi_phi[m+N_1-1,m]=round(1/36,av)
    phi_phi[m+N_1+1,m]=round(1/36,av)
for m in range(N_1*(N_2-1)+1,N_1*N_2-1):
    phi_phi[m,m]=round(8/36,av)
    phi_phi[m-1,m]=round(2/36,av)
    phi_phi[m+1,m]=round(2/36,av)
    phi_phi[m-N_1,m]=round(4/36,av)
    phi_phi[m-N_1-1,m]=round(1/36,av)
    phi_phi[m-N_1+1,m]=round(1/36,av)
##HÃ¥yder##
for m in range(1,N_2-1):
```

```
    phi_phi[m*N_1,m*N_1]=round(8/36,av)
    phi_phi[m*N_1+1,m*N_1]=round(4/36,av)
    phi_phi[(m-1)*N_1,m*N_1]=round(2/36,av)
    phi_phi[(m+1)*N_1,m*N_1]=round(2/36,av)
    phi_phi[(m-1)*N_1+1,m*N_1]=round(1/36,av)
    phi_phi[(m+1)*N_1+1,m*N_1]=round(1/36,av)
for m in range(1,N_2-1):
    phi_phi[(m+1)*N_1-1,(m+1)*N_1-1]=round(8/36,av)
    phi_phi[(m+1)*N_1-2,(m+1)*N_1-1]=round(4/36,av)
    phi_phi[(m)*N_1-1,(m+1)*N_1-1]=round(2/36,av)
    phi_phi[(m+2)*N_1-1,(m+1)*N_1-1]=round(2/36,av)
    phi_phi[(m)*N_1-2,(m+1)*N_1-1]=round(1/36,av)
    phi_phi[(m+2)*N_1-2,(m+1)*N_1-1]=round(1/36,av)
##Indre punkter##
for m in range(1,N_2-1):
    for n in range(1,N_1-1):
        phi_phi[m*N_1+n,m*N_1+n]=round(16/36,av)
        phi_phi[m*N_1+n-1,m*N_1+n]=round(4/36,av)
        phi_phi[m*N_1+n+1,m*N_1+n]=round(4/36,av)
        phi_phi[(m+1)*N_1+n,m*N_1+n]=round(4/36,av)
        phi_phi[(m-1)*N_1+n,m*N_1+n]=round(4/36,av)
        phi_phi[(m+1)*N_1+n-1,m*N_1+n]=round(1/36,av)
        phi_phi[(m+1)*N_1+n+1,m*N_1+n]=round(1/36,av)
        phi_phi[(m-1)*N_1+n-1,m*N_1+n]=round(1/36,av)
        phi_phi[(m-1)*N_1+n+1,m*N_1+n]=round(1/36,av)

#Neste
##HjÃ¿rner##
phiX_phiX[0,0]=round(2/6,av)
phiX_phiX[N_1-1,N_1-1]=round(2/6,av)
phiX_phiX[N_1*(N_2-1),N_1*(N_2-1)]=round(2/6,av)
phiX_phiX[N_1*N_2-1,N_1*N_2-1]=round(2/6,av)

phiX_phiX[1,0]=round(-2/6,av)
phiX_phiX[N_1-2,N_1-1]=round(-2/6,av)
phiX_phiX[N_1*(N_2-1)+1,N_1*(N_2-1)]=round(-2/6,av)
phiX_phiX[N_1*N_2-2,N_1*N_2-1]=round(-2/6,av)

phiX_phiX[N_1,0]=round(1/6,av)
phiX_phiX[N_1-1+N_1,N_1-1]=round(1/6,av)
phiX_phiX[N_1*(N_2-2),N_1*(N_2-1)]=round(1/6,av)
phiX_phiX[N_1*(N_2-1)-1,N_1*N_2-1]=round(1/6,av)

phiX_phiX[N_1+1,0]=round(-1/6,av)
phiX_phiX[N_1-2+N_1,N_1-1]=round(-1/6,av)
phiX_phiX[N_1*(N_2-2)+1,N_1*(N_2-1)]=round(-1/6,av)
phiX_phiX[N_1*(N_2-1)-2,N_1*N_2-1]=round(-1/6,av)
##Lengder##
for m in range(1,N_1-1):
    phiX_phiX[m,m]=round(4/6,av)
    phiX_phiX[m-1,m]=round(-2/6,av)
    phiX_phiX[m+1,m]=round(-2/6,av)
    phiX_phiX[m+N_1,m]=round(2/6,av)
    phiX_phiX[m+N_1-1,m]=round(-1/6,av)
    phiX_phiX[m+N_1+1,m]=round(-1/6,av)
for m in range(N_1*(N_2-1)+1,N_1*N_2-1):
    phiX_phiX[m,m]=round(4/6,av)
    phiX_phiX[m-1,m]=round(-2/6,av)
    phiX_phiX[m+1,m]=round(-2/6,av)
    phiX_phiX[m-N_1,m]=round(2/6,av)
    phiX_phiX[m-N_1-1,m]=round(-1/6,av)
    phiX_phiX[m-N_1+1,m]=round(-1/6,av)
```

```
##HÃÿyder##
for m in range(1,N_2-1):
    phiX_phiX[m*N_1,m*N_1]=round(4/6,av)
    phiX_phiX[m*N_1+1,m*N_1]=round(-4/6,av)
    phiX_phiX[(m-1)*N_1,m*N_1]=round(1/6,av)
    phiX_phiX[(m+1)*N_1,m*N_1]=round(1/6,av)
    phiX_phiX[(m-1)*N_1+1,m*N_1]=round(-1/6,av)
    phiX_phiX[(m+1)*N_1+1,m*N_1]=round(-1/6,av)
for m in range(1,N_2-1):
    phiX_phiX[(m+1)*N_1-1,(m+1)*N_1-1]=round(4/6,av)
    phiX_phiX[(m+1)*N_1-2,(m+1)*N_1-1]=round(-4/6,av)
    phiX_phiX[(m)*N_1-1,(m+1)*N_1-1]=round(1/6,av)
    phiX_phiX[(m+2)*N_1-1,(m+1)*N_1-1]=round(1/6,av)
    phiX_phiX[(m)*N_1-2,(m+1)*N_1-1]=round(-1/6,av)
    phiX_phiX[(m+2)*N_1-2,(m+1)*N_1-1]=round(-1/6,av)
##Indre punkter##
for m in range(1,N_2-1):
    for n in range(1,N_1-1):
        phiX_phiX[m*N_1+n,m*N_1+n]=round(8/6,av)
        phiX_phiX[m*N_1+n-1,m*N_1+n]=round(-4/6,av)
        phiX_phiX[m*N_1+n+1,m*N_1+n]=round(-4/6,av)
        phiX_phiX[(m+1)*N_1+n,m*N_1+n]=round(2/6,av)
        phiX_phiX[(m-1)*N_1+n,m*N_1+n]=round(2/6,av)
        phiX_phiX[(m+1)*N_1+n-1,m*N_1+n]=round(-1/6,av)
        phiX_phiX[(m+1)*N_1+n+1,m*N_1+n]=round(-1/6,av)
        phiX_phiX[(m-1)*N_1+n-1,m*N_1+n]=round(-1/6,av)
        phiX_phiX[(m-1)*N_1+n+1,m*N_1+n]=round(-1/6,av)

#Neste
##HjÃÿrner##
phiY_phiY[0,0]=round(2/6,av)
phiY_phiY[N_1-1,N_1-1]=round(2/6,av)
phiY_phiY[N_1*(N_2-1),N_1*(N_2-1)]=round(2/6,av)
phiY_phiY[N_1*N_2-1,N_1*N_2-1]=round(2/6,av)

phiY_phiY[1,0]=round(1/6,av)
phiY_phiY[N_1-2,N_1-1]=round(1/6,av)
phiY_phiY[N_1*(N_2-1)+1,N_1*(N_2-1)]=round(1/6,av)
phiY_phiY[N_1*N_2-2,N_1*N_2-1]=round(1/6,av)

phiY_phiY[N_1,0]=round(-2/6,av)
phiY_phiY[N_1-1+N_1,N_1-1]=round(-2/6,av)
phiY_phiY[N_1*(N_2-2),N_1*(N_2-1)]=round(-2/6,av)
phiY_phiY[N_1*(N_2-1)-1,N_1*N_2-1]=round(-2/6,av)

phiY_phiY[N_1+1,0]=round(-1/6,av)
phiY_phiY[N_1-2+N_1,N_1-1]=round(-1/6,av)
phiY_phiY[N_1*(N_2-2)+1,N_1*(N_2-1)]=round(-1/6,av)
phiY_phiY[N_1*(N_2-1)-2,N_1*N_2-1]=round(-1/6,av)

##Lengder##
for m in range(1,N_1-1):
    phiY_phiY[m,m]=round(4/6,av)
    phiY_phiY[m-1,m]=round(1/6,av)
    phiY_phiY[m+1,m]=round(1/6,av)
    phiY_phiY[m+N_1,m]=round(-4/6,av)
    phiY_phiY[m+N_1-1,m]=round(-1/6,av)
    phiY_phiY[m+N_1+1,m]=round(-1/6,av)
for m in range(N_1*(N_2-1)+1,N_1*N_2-1):
    phiY_phiY[m,m]=round(4/6,av)
    phiY_phiY[m-1,m]=round(1/6,av)
    phiY_phiY[m+1,m]=round(1/6,av)
```

```
    phiY_phiY[m-N_1,m]=round(-4/6,av)
    phiY_phiY[m-N_1-1,m]=round(-1/6,av)
    phiY_phiY[m-N_1+1,m]=round(-1/6,av)
##HÃÿyder##
for m in range(1,N_2-1):
    phiY_phiY[m*N_1,m*N_1]=round(4/6,av)
    phiY_phiY[m*N_1+1,m*N_1]=round(2/6,av)
    phiY_phiY[(m-1)*N_1,m*N_1]=round(-2/6,av)
    phiY_phiY[(m+1)*N_1,m*N_1]=round(-2/6,av)
    phiY_phiY[(m-1)*N_1+1,m*N_1]=round(-1/6,av)
    phiY_phiY[(m+1)*N_1+1,m*N_1]=round(-1/6,av)
for m in range(1,N_2-1):
    phiY_phiY[(m+1)*N_1-1,(m+1)*N_1-1]=round(4/6,av)
    phiY_phiY[(m+1)*N_1-2,(m+1)*N_1-1]=round(2/6,av)
    phiY_phiY[(m)*N_1-1,(m+1)*N_1-1]=round(-2/6,av)
    phiY_phiY[(m+2)*N_1-1,(m+1)*N_1-1]=round(-2/6,av)
    phiY_phiY[(m)*N_1-2,(m+1)*N_1-1]=round(-1/6,av)
    phiY_phiY[(m+2)*N_1-2,(m+1)*N_1-1]=round(-1/6,av)
##Indre punkter##
for m in range(1,N_2-1):
    for n in range(1,N_1-1):
        phiY_phiY[m*N_1+n,m*N_1+n]=round(8/6,av)
        phiY_phiY[m*N_1+n-1,m*N_1+n]=round(2/6,av)
        phiY_phiY[m*N_1+n+1,m*N_1+n]=round(2/6,av)
        phiY_phiY[(m+1)*N_1+n,m*N_1+n]=round(-4/6,av)
        phiY_phiY[(m-1)*N_1+n,m*N_1+n]=round(-4/6,av)
        phiY_phiY[(m+1)*N_1+n-1,m*N_1+n]=round(-1/6,av)
        phiY_phiY[(m+1)*N_1+n+1,m*N_1+n]=round(-1/6,av)
        phiY_phiY[(m-1)*N_1+n-1,m*N_1+n]=round(-1/6,av)
        phiY_phiY[(m-1)*N_1+n+1,m*N_1+n]=round(-1/6,av)

#Neste
##HjÃyrner##
phiX_phi[0,0]=round(-2/12,av)
phiX_phi[N_1-1,N_1-1]=round(2/12,av)
phiX_phi[N_1*(N_2-1),N_1*(N_2-1)]=round(-2/12,av)
phiX_phi[N_1*N_2-1,N_1*N_2-1]=round(2/12,av)

phiX_phi[1,0]=round(-2/12,av)
phiX_phi[N_1-2,N_1-1]=round(2/12,av)
phiX_phi[N_1*(N_2-1)+1,N_1*(N_2-1)]=round(-2/12,av)
phiX_phi[N_1*N_2-2,N_1*N_2-1]=round(2/12,av)

phiX_phi[N_1,0]=round(-1/12,av)
phiX_phi[N_1-1+N_1,N_1-1]=round(1/12,av)
phiX_phi[N_1*(N_2-2),N_1*(N_2-1)]=round(-1/12,av)
phiX_phi[N_1*(N_2-1)-1,N_1*N_2-1]=round(1/12,av)

phiX_phi[N_1+1,0]=round(-1/12,av)
phiX_phi[N_1-2+N_1,N_1-1]=round(1/12,av)
phiX_phi[N_1*(N_2-2)+1,N_1*(N_2-1)]=round(-1/12,av)
phiX_phi[N_1*(N_2-1)-2,N_1*N_2-1]=round(1/12,av)
##Lengder##
for m in range(1,N_1-1):
    phiX_phi[m-1,m]=round(2/12,av)
    phiX_phi[m+1,m]=round(-2/12,av)
    phiX_phi[m+N_1-1,m]=round(1/12,av)
    phiX_phi[m+N_1+1,m]=round(-1/12,av)
for m in range(N_1*(N_2-1)+1,N_1*N_2-1):
    phiX_phi[m-1,m]=round(2/12,av)
    phiX_phi[m+1,m]=round(-2/12,av)
    phiX_phi[m-N_1-1,m]=round(1/12,av)
```

```
    phiX_phi[m-N_1+1,m]=round(-1/12,av)
##HÃÿyder##
for m in range(1,N_2-1):
    phiX_phi[m*N_1,m*N_1]=round(-4/12,av)
    phiX_phi[m*N_1+1,m*N_1]=round(-4/12,av)
    phiX_phi[(m-1)*N_1,m*N_1]=round(-1/12,av)
    phiX_phi[(m+1)*N_1,m*N_1]=round(-1/12,av)
    phiX_phi[(m-1)*N_1+1,m*N_1]=round(-1/12,av)
    phiX_phi[(m+1)*N_1+1,m*N_1]=round(-1/12,av)
for m in range(1,N_2-1):
    phiX_phi[(m+1)*N_1-1,(m+1)*N_1-1]=round(4/12,av)
    phiX_phi[(m+1)*N_1-2,(m+1)*N_1-1]=round(4/12,av)
    phiX_phi[(m)*N_1-1,(m+1)*N_1-1]=round(1/12,av)
    phiX_phi[(m+2)*N_1-1,(m+1)*N_1-1]=round(1/12,av)
    phiX_phi[(m)*N_1-2,(m+1)*N_1-1]=round(1/12,av)
    phiX_phi[(m+2)*N_1-2,(m+1)*N_1-1]=round(1/12,av)
##Indre punkter##
for m in range(1,N_2-1):
    for n in range(1,N_1-1):
        phiX_phi[m*N_1+n-1,m*N_1+n]=round(4/12,av)
        phiX_phi[m*N_1+n+1,m*N_1+n]=round(-4/12,av)
        phiX_phi[(m+1)*N_1+n-1,m*N_1+n]=round(1/12,av)
        phiX_phi[(m+1)*N_1+n+1,m*N_1+n]=round(-1/12,av)
        phiX_phi[(m-1)*N_1+n-1,m*N_1+n]=round(1/12,av)
        phiX_phi[(m-1)*N_1+n+1,m*N_1+n]=round(-1/12,av)

#Neste
##HjÃÿrner##
phiY_phi[0,0]=round(-2/12,av)
phiY_phi[N_1-1,N_1-1]=round(-2/12,av)
phiY_phi[N_1*(N_2-1),N_1*(N_2-1)]=round(2/12,av)
phiY_phi[N_1*N_2-1,N_1*N_2-1]=round(2/12,av)

phiY_phi[1,0]=round(-1/12,av)
phiY_phi[N_1-2,N_1-1]=round(-1/12,av)
phiY_phi[N_1*(N_2-1)+1,N_1*(N_2-1)]=round(1/12,av)
phiY_phi[N_1*N_2-2,N_1*N_2-1]=round(1/12,av)

phiY_phi[N_1,0]=round(-2/12,av)
phiY_phi[N_1-1+N_1,N_1-1]=round(-2/12,av)
phiY_phi[N_1*(N_2-2),N_1*(N_2-1)]=round(2/12,av)
phiY_phi[N_1*(N_2-1)-1,N_1*N_2-1]=round(2/12,av)

phiY_phi[N_1+1,0]=round(-1/12,av)
phiY_phi[N_1-2+N_1,N_1-1]=round(-1/12,av)
phiY_phi[N_1*(N_2-2)+1,N_1*(N_2-1)]=round(1/12,av)
phiY_phi[N_1*(N_2-1)-2,N_1*N_2-1]=round(1/12,av)
##Lengder##
for m in range(1,N_1-1):
    phiY_phi[m,m]=round(-4/12,av)
    phiY_phi[m-1,m]=round(-1/12,av)
    phiY_phi[m+1,m]=round(-1/12,av)
    phiY_phi[m+N_1,m]=round(-4/12,av)
    phiY_phi[m+N_1-1,m]=round(-1/12,av)
    phiY_phi[m+N_1+1,m]=round(-1/12,av)
for m in range(N_1*(N_2-1)+1,N_1*N_2-1):
    phiY_phi[m,m]=round(4/12,av)
    phiY_phi[m-1,m]=round(1/12,av)
    phiY_phi[m+1,m]=round(1/12,av)
    phiY_phi[m-N_1,m]=round(4/12,av)
    phiY_phi[m-N_1-1,m]=round(1/12,av)
    phiY_phi[m-N_1+1,m]=round(1/12,av)
```

```
##HÃÿyder##
for m in range(1,N_2-1):
    phiY_phi[(m-1)*N_1,m*N_1]=round(2/12,av)
    phiY_phi[(m+1)*N_1,m*N_1]=round(-2/12,av)
    phiY_phi[(m-1)*N_1+1,m*N_1]=round(1/12,av)
    phiY_phi[(m+1)*N_1+1,m*N_1]=round(-1/12,av)
for m in range(1,N_2-1):
    phiY_phi[(m)*N_1-1,(m+1)*N_1-1]=round(2/12,av)
    phiY_phi[(m+2)*N_1-1,(m+1)*N_1-1]=round(-2/12,av)
    phiY_phi[(m)*N_1-2,(m+1)*N_1-1]=round(1/12,av)
    phiY_phi[(m+2)*N_1-2,(m+1)*N_1-1]=round(-1/12,av)
##Indre punkter##
for m in range(1,N_2-1):
    for n in range(1,N_1-1):
        phiY_phi[(m+1)*N_1+n,m*N_1+n]=round(-4/12,av)
        phiY_phi[(m-1)*N_1+n,m*N_1+n]=round(4/12,av)
        phiY_phi[(m+1)*N_1+n-1,m*N_1+n]=round(-1/12,av)
        phiY_phi[(m+1)*N_1+n+1,m*N_1+n]=round(-1/12,av)
        phiY_phi[(m-1)*N_1+n-1,m*N_1+n]=round(1/12,av)
        phiY_phi[(m-1)*N_1+n+1,m*N_1+n]=round(1/12,av)
"""
#Test av positivitet ved Ãě se pÃě pivots
def row_echelon(A):
    r, c = A.shape
    if r == 0 or c == 0:
        return A
    for i in range(len(A)):
        if A[i,0] != 0:
            break
    else:
        B = row_echelon(A[:,1:])
        return np.hstack([A[:,:1], B])
    if i > 0:
        ith_row = A[i].copy()
        A[i] = A[0]
        A[0] = ith_row
    A[0] = A[0] / A[0,0]
    A[1:] -= A[0] * A[1:,0:1]
    B = row_echelon(A[1:,1:])
    return np.vstack([A[:1], np.hstack([A[1:,:1], B]) ])

A=row_echelon(phi_phi)
epsilon=0.0001
for n in range(N_1*N_2):
    for i in range(N_1*N_2):
        if abs(A[n,i])>epsilon:
            if A[n,i]<epsilon:
                print("matrisen er ikke positiv definit")
"""
phi_phi=phi_phi.reshape([N_1**2*N_2**2])
phiX_phiX=phiX_phiX.reshape([N_1**2*N_2**2])
phiY_phiY=phiY_phiY.reshape([N_1**2*N_2**2])
phiX_phi=phiX_phi.reshape([N_1**2*N_2**2])
phiY_phi=phiY_phi.reshape([N_1**2*N_2**2])
phi_master=[phi_phi,phiX_phiX,phiY_phiY,phiX_phi,phiY_phi]

np.savetxt("phi_master",phi_master)
np.savetxt("Konstanter",[A_1,A_2,B_1,B_2,dt,h,T,x_start,x_slutt,y_start,y_slutt,mass_lumping])
np.savetxt("U_now",U_now)
np.savetxt("F",F)
```

Bibliografi

[1] Ball, W. W. R. (W. R. *A short account of the history of mathematics*. eng. New York, Dover Publications, 1960. URL: http://archive.org/details/shortaccountofhi0000ball (sjekket 19.04.2023).

[2] C.H, E. *The historical development of the calculus*. Engelsk. New York: Springer-Verlag, 1979.

[3] Crank, J. og Nicolson, P. «A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type». en. I: *Advances in Computational Mathematics* årg. 6, nr. 1 (des. 1996), s. 207–226. URL: https://doi.org/10.1007/BF02127704 (sjekket 14.05.2023).

[4] *Crank Nicolson Scheme for the Heat Equation*. URL: https://people.sc.fsu.edu/~jpeterson/5-CrankNicolson.pdf (sjekket 24.04.2023).

[5] *Distance between Point and Line | Brilliant Math & Science Wiki*. en-us. URL: https://brilliant.org/wiki/distance-between-point-and-line/ (sjekket 20.04.2023).

[6] Duffy, D. J. «A critique of the crank nicolson scheme strengths and weaknesses for financial instrument pricing». en. I: *Wilmott* årg. 2004, nr. 4 (jul. 2004), s. 68–76. URL: http://doi.wiley.com/10.1002/wilm.42820040417 (sjekket 13.05.2023).

[7] Dunham, W. *The Genius of Euler: Reflections on his Life and Work*. en. Google-Books-ID: UMH2DwAAQBAJ. American Mathematical Soc., aug. 2020.

[8] Evans, L. C. *Partial differential equations*. eng. 2nd ed. Bd. vol. 19. Graduate studies in mathematics. Providence, R.I: American Mathematical Society, 2010.

[9] Friedman, A. *Partial Differential Equations of Parabolic Type*. en. Google-Books-ID: e0HDAgAAQBAJ. Courier Corporation, aug. 2013.

[10] *Galerkin Finite Element Methods for Parabolic Problems*. en. Bd. 25. Springer Series in Computational Mathematics. Berlin, Heidelberg: Springer, 2006. URL: http://link.springer.com/10.1007/3-540-33122-0 (sjekket 21.04.2023).

[11] Glinowiecka-Cox, M. «Analytic Solution of 1D Diffusion-Convection Equation with Varying Boundary Conditions». en. Bachelor of Science in Mathematics and University Honors. Portland State University, jun. 2022. URL: https://pdxscholar.library.pdx.edu/honorstheses/1182 (sjekket 02.05.2023).

[12] Herivel, J. *Joseph Fourier : the man and the physicist*. eng. Oxford : Clarendon Press, 1975. URL: http://archive.org/details/josephfourierman0000heri (sjekket 13.05.2023).

[13] Horn, R. A. og Johnson, C. R. *Matrix Analysis*. Cambridge: Cambridge University Press, 1985. URL: https://www.cambridge.org/core/books/matrix-analysis/9CF2CB491C9E97948B15FAD835EF9A8B (sjekket 20.04.2023).

[14] *Jean-Baptiste Biot - Biography*. en. URL: https://mathshistory.st-andrews.ac.uk/Biographies/Biot/ (sjekket 13.05.2023).

[15] Leonhardi Euleri. *Institutiones_calculi_integrals_volumen-Primum*. fre. Lipsiae Et Berolini. URL: http://archive.org/details/institutionescal020326mbp (sjekket 14.05.2023).

[16] Marsden, J. mfl., red. *Introduction to Partial Differential Equations*. en. Bd. 29. Texts in Applied Mathematics. Berlin/Heidelberg: Springer-Verlag, 2005. URL: http://link.springer.com/10.1007/b138016 (sjekket 21.04.2023).

[17] *Numerical recipes in C : the art of scientific computing*. eng. Cambridge ; New York : Cambridge University Press, 1995. URL: http://archive.org/details/numericalrecipes0865unse (sjekket 15.05.2023).

[18] *Overview — scikit-sparse 0.4.3+9.ge35b764 documentation*. URL: https://scikit-sparse.readthedocs.io/en/latest/overview.html#download (sjekket 24.04.2023).

[19] Phillips, G. M. og Taylor, P. J. «Chapter 7 - NUMERICAL INTEGRATION AND DIFFERENTIATION». en. I: *Theory and Applications of Numerical Analysis (Second Edition)*. Red. av Phillips, G. M. og Taylor, P. J. London: Academic Press, jan. 1996, s. 160–195. URL: https://www.sciencedirect.com/science/article/pii/B9780125535601500082 (sjekket 24.04.2023).

[20] Plemmons, R. J. «M-matrix characterizations.I—nonsingular M-matrices». en. I: *Linear Algebra and its Applications* årg. 18, nr. 2 (jan. 1977), s. 175–188. URL: https://www.sciencedirect.com/science/article/pii/0024379577900738 (sjekket 23.04.2023).

[21] Press, W. H. og Teukolsky, S. A. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. en. Google-Books-ID: 1aAOdzK3FegC. Cambridge University Press, sep. 2007.

[22] *Scala Graduum Caloris. Calorum Descriptiones & Figna*. lat. Philosophical Transactions (1683-1775), jan. 1700. URL: http://archive.org/details/jstor-102813 (sjekket 19.04.2023).

[23] *Sparse matrices (scipy.sparse) — SciPy v1.10.1 Manual*. URL: https://docs.scipy.org/doc/scipy/reference/sparse.html (sjekket 12.05.2023).

[24] *The Euler Archive*. URL: http://eulerarchive.maa.org/ (sjekket 13.05.2023).