TU Delft

Bacherloreindproject

Technische Wiskunde

Iteratieve oplosmethoden voor de warmtevergelijking

Zoë ten Napel 4590341

14 januari 2020

Abstract

In dit onderzoek is gekeken naar de toepassing van de Scheduled Relaxation Jacobi (SRJ) methode op de tijdsafhankelijke warmtevergelijking. De scheduled Relaxation Jacobi methode is een iteratieve methode die bestaat uit meerdere cycli van verschillende gewichten van iteraties van de gerelaxeerde Jacobi methode. Deze methode is ontworpen om de convergentie van de Jacobi methode te versnellen voor elliptische vergelijkingen. Deze methode is vergeleken met de standaard Jacobi methode om te onderzoeken of deze methode ook tijdsafhankelijke problemen sneller kan oplossen. Zowel de eendimensionele als de tweedimensionele warmtevergelijking is eerst gediscretiseerd met Dirichlet en Neumann randvoorwaarden. Vervolgens zijn de Jacobi en de SRJ methode toegepast op de berekende stelsels. De oplossingen van de iteratieve methoden zijn geanalyseerd aan de hand van amplificatiefactoren die berekend zijn met behulp van de eigenwaarden van de stelselmatrices. Deze analyses zijn vervolgens getest met behulp van MATLAB. Uit de analyses en de resultaten kan worden geconcludeerd dat de SRJ methode tijdsafhankelijke problemen niet sneller oplost.

Inhoudsopgave

Inleiding **2**

1 De warmtevergelijking **3**

2 Discretisatie **3**
2.1 Dirichlet randvoorwaarden . . . 5
2.2 Neumann randvoorwaarden . . . 6
2.3 Lexicografische ordening . . . 6
2.3.1 De stelselmatrix voor de eendimensionele warmtevergelijking . . . 7
2.3.2 De stelselmatrix voor de tweedimensionele warmtevergelijking . . . 8

3 Iteratieve methoden **9**
3.1 De Jacobi methode . . . 10
3.1.1 Convergentie van de Jacobi methode . . . 12
3.2 Von Neumann analyse . . . 14
3.3 De Scheduled Relaxation Jacobi (SRJ) methode . . . 15
3.3.1 De gedempte Jacobi methode . . . 15
3.3.2 $P = 2$, $M = 2$ schema's . . . 17
3.3.3 Grotere schema's . . . 17

4 Resultaten **18**
4.1 Resultaten voor de eendimensionele warmtevergelijking . . . 18
4.1.1 Dirichlet randvoorwaarden . . . 18
4.1.2 Neumann randvoorwaarden . . . 20
4.2 Resultaten voor de tweedimensionele warmtevergelijking . . . 22
4.2.1 Dirichlet randvoorwaarden . . . 22
4.2.2 Neumann randvoorwaarden . . . 24

5 Conclusie **25**

Referenties **26**

Bijlage **26**
MATLAB code voor de eendimensionele warmtevergelijking . . . 26
MATLAB code voor de tweedimensionele warmtevergelijking . . . 30

Inleiding

Vrijwel elk natuurkundig fenomeen kan worden omschreven met een partiële differentiaalvergelijking. Een differentiaalvergelijking omschrijft een situatie met behulp van een functie en de afgeleiden van die functie. De afgeleiden geven hierbij een verandering in een bepaalde richting aan. Deze richting kan zowel een plaats of een tijd aanduiden. Wanneer een differentiaalvergelijking veranderingen in verschillende richtingen omschrijft en dus in de vergelijking afgeleiden van de functie naar verschillende variabelen voorkomen, noemt men dit een partiële differentiaalvergelijking. Een voorbeeld van zo'n vergelijking is de warmtevergelijking. De warmtevergelijking omschrijft de beweging van warmte in een bepaalde ruimte in een bepaalde tijd.

Het exact oplossen van partiële differentiaalvergelijkingen kan veel en ingewikkeld werk zijn. Daarom wordt er vaak voor gekozen om de oplossing iteratief te benaderen. Hierbij wordt de vergelijking eerst omgezet naar een discreet stelsel met behulp van discretisatietmethodes. In dit onderzoek is gebruik gemaakt van de eindige differentiemethode, het gebied wordt onderverdeeld in punten die een bepaalde plaats en tijd aangeven waarbij aan elk punt een functiewaarden wordt aangeschreven. De functiewaarden zijn hierbij afhankelijk van de omliggende punten. Hierdoor ontstaat een lineair stelsel dat kan worden opgelost met iteratieve methoden. De grootte van dit stelsel is afhankelijk van het aantal punten dat wordt gekozen om het gebied te omschrijven, dit wordt ook wel de gridgrootte genoemd. Een grotere gridgrootte zorgt voor een accuratere benadering, maar vaak kost het ook meer tijd en moeite om de benadering te vinden. Iteratieve methoden schatten een oplossing van het stelsel, berekenen de fout tussen deze schatting en de daadwerkelijke oplossing en op basis van analyse van deze fout maken ze een betere schatting. Zo komt een werkende iteratieve methode steeds dichter bij de oplossing. Er wordt dan gesproken van convergentie. Een voorbeeld van zo'n iteratieve methode is de Jacobi methode. De Jacobi methode is een simpel te implenteren methode die de geschatte waarde vermenigvuldigt met een stelsel aan vergelijkingen en vervolgens deze benadering aanpast. Andere methoden, zoals de Gauss-Seidel methode, passen de benadering aan tijdens het vermenigvuldigen met de vergelijkingen. De Jacobi methode wordt gekenmerkt om zijn langzame onvergentie, maar de snelheid van de convergentie is erg afhankelijk van het lineaire stelsel waarop de methode geïmplementeerd wordt.

Xijang en Mittal (2014) hebben een methode ontwikkelt om de convergentie van de Jacobi methode te versnellen voor elliptische vergelijkingen, zoals beschreven in 'Acceleration of the Jacobi iterative method by factors exceeding 100 using scheduled relaxation'. De scheduled Relaxation Jacobi methode is een iteratieve methode die bestaat uit meerdere cycli van iteraties van de gerelaxeerde Jacobi methode. Elliptische vergelijking zijn tijdsonafhankelijke partiële differentiaalvergelijkingen. De warmtevergelijking wordt ook wel een parabolische vergelijking genoemd, het is namelijk een tijdsafhankelijk probleem. In dit onderzoek wordt gekeken naar de convergentie van de Jacobi methode en de Scheduled Relaxation Jacobi methode voor de warmtevergelijking. Xijang en Mittal (2014) hebben geconcludeerd dat de SRJ methode de convergentie van de Jacobi methode significant versnelt voor elliptische vergelijkingen. De Jacobi methode convergeert voor deze vergelijkingen echter een stuk langzamer dan voor parabolische vergelijkingen. De verwachting is hierom dat de convergentie van de SRJ methode niet veel sneller is dan de convergentie van de Jacobi methode toegepast op de warmtevergelijking. In dit onderzoek zal dit eerst analytisch worden onderzocht. Vervolgens zullen deze analyses worden getest met behulp van MATLAB.

1 De warmtevergelijking

Er zal worden gekeken naar de warmtediffussie in een rechtoekig gebied. De zijdes, boven en onderkant van het gebied kunnen wel of niet zijn geïsoleerd. Het gebied heeft een hoogte van lengte H en een breedte van lengte L. De warmtediffusie $u(\mathbf{x}, t)$ is te beschrijven met een partiële differentiaalvergelijking van de vorm:

$$\frac{\partial u}{\partial t} = \alpha \cdot \nabla^2 u \tag{1}$$

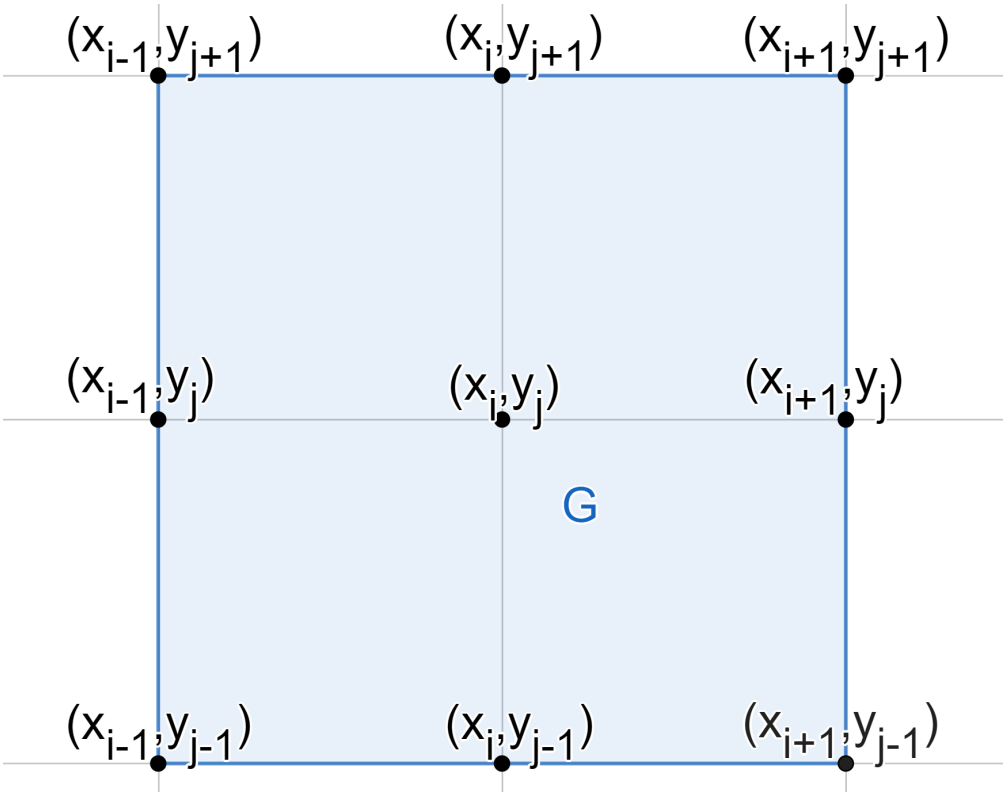
Waarbij de constante α staat voor de thermische diffusiviteit.
Wanneer een zij-, boven- en/of onderkant wordt geïsoleerd ontstaan de volgende randvoorwaarden:

$$\begin{cases} \frac{\partial u}{\partial n}(0, y, t) = 0 & \text{(rechterkant)} \\ \frac{\partial u}{\partial n}(L, y, t) = 0 & \text{(linkerkant)} \\ \frac{\partial u}{\partial n}(x, 0, t) = 0 & \text{(onderkant)} \\ \frac{\partial u}{\partial n}(x, H, t) = 0 & \text{(bovenkant)} \end{cases}$$

Waarbij n de vector is die loodrecht op het oppervlak staat en $\frac{\delta u}{\delta n}$ de flux aangeeft door het oppervlak die bij een perfecte isolatie dus 0 zou moeten zijn.
Wanneer er een warmtebron in de ruimte wordt geplaatst geeft dit de bronterm $f(x, y, t)$. De warmtevergelijking met bronterm is van de volgende vorm:

$$\frac{\partial u}{\partial t} = \alpha \cdot \nabla^2 u + f(x, y, t) \tag{2}$$

2 Discretisatie



Figuur (1) Gediscretiseerd gebied G om punt (x_i, y_j)

Om de verkregen partiële differentiale vergelijking iteratief te kunnen oplossen, moet deze gediscretiseerd worden. Dit kan worden gedaan met behulp van de eindige differentiemethode. Zoals de naam al zegt, draait het bij deze methode om het benaderen van de afgeleide op een eindig aantal punten. Ten eerste, wordt het gebied gediscretisseerd door het continue gebied te verdelen in discrete punten met een vaste afstand h tussen de punten. Voor de discretisatie van het tweedimensionele model wordt een netwerk van N elementen uit het domeingebied gecreëerd. Hierbij ontstaat een uniform rooster G_h, $h = \frac{1}{N}$ van $(N+1)^2$ punten inclusief de punten op de rand, deze is als volgt gedefinieerd:

$$G_h = \{(x_i, y_j) \mid x_i = ih, y_j = jh; h = \frac{1}{N}, 1 \leq i, j \leq N+1; N \in \mathbb{N}\}$$

Vervolgens zal elk punt in het gebied apart worden benaderd door te kijken naar de vier omliggende punten. Kijk bijvoorbeeld naar figuur 1. Om punt (x_i, y_j) te benaderen wordt gebruik gemaakt van de punten (x_{i-1}, y_j), (x_{i+1}, y_j), (x_i, y_{j-1}) en (x_i, y_{j+1}). Door gebruik te maken van de gecentreerde afstandsformules kan de warmtediffusie in punt (x_i, y_j) als volgt worden benaderd:

$$\frac{\partial^2 u}{\partial x^2}(x_i, y_j, t) \approx \frac{u(x_{i-1}, y_j, t) - 2u(x_i, y_j, t) + u(x_{i+1}, y_j, t)}{h^2} \tag{3}$$

$$\frac{\partial^2 u}{\partial y^2}(x_i, y_j, t) \approx \frac{u(x_i, y_{j-1}, t) - 2u(x_i, y_j) + u(x_i, y_{j+1}, t)}{h^2} \tag{4}$$

Op een analoge manier kan het tijdsafhankelijke deel van de warmtevergelijking kan worden benaderd met behulp van de Euler voorwaartse afstandsformule:

$$\frac{\partial u}{\partial t}(x_i, y_j, t_m) \approx \frac{u(x_i, y_j, t_{m+1}) - u(x_i, y_j, t_m)}{\Delta t} \tag{5}$$

waarbij Δt de tijdstap aangeeft en er geldt $t_m = t_{m-1} + \Delta t$. Deze benadering kan gebruikt worden voor alle inwendige punten van G_h. Dit geeft het volgende voor de partiële differentiaalvergelijkingen

$$\begin{cases} \nabla^2 u_{i,j} \approx \dfrac{u^m_{i-1,j} + u^m_{i,j-1} - 4u^m_{i,j} + u^m_{i+1,j} + u^m_{i,j+1}}{h^2} & \text{voor } 2 \leq i, j \leq N, \\ \dfrac{\partial u}{\partial t}(x_i, y_j, t_m) \approx \dfrac{u^{m+1}_{i,j} - u^m_{i,j}}{\Delta t} & \text{voor } m \geq 1. \end{cases}$$

Het combineren van de gediscretiseerde partiële differentiaalvergelijkingen geeft

$$\frac{u^{m+1}_{i,j} - u^m_{i,j}}{\Delta t} \approx \alpha \frac{u^m_{i-1,j} + u^m_{i,j-1} - 4u^m_{i,j} + u^m_{i+1,j} + u^m_{i,j+1}}{h^2} \tag{6}$$

De benadering van de warmtevergelijking gediscretiseerd op de inwendige punten van G_h kan nu worden geschreven als

$$u^{m+1}_{i,j} = u^m_{i,j} + \frac{\alpha \Delta t}{h^2}(u^m_{i-1,j} + u^m_{i,j-1} - 4u^m_{i,j} + u^m_{i+1,j} + u^m_{i,j+1}) + f^m_{i,j} \tag{7}$$

Deze benadering geeft echter geen stelsel vergelijkingen en werkt alleen voor een kleine Δt, als deze te groot is kunnen instabiele benaderingen worden gecreeërd en zal de fout tussen de oplossing en de benadering divergeren. Een betere methode om het tijdsafhankelijke deel van de warmtevergelijking te benaderen is de Euler achterwaartse methode.

$$\frac{\partial u}{\partial t}(x_i, y_j, t_m) \approx \frac{u(x_i, y_j, t_m) - u(x_i, y_j, t_{m-1})}{\Delta t} \tag{8}$$

Dit kan weer gecombineerd worden met de gediscretiseerde Laplacian om de gediscretiseerde warmtevergelijking als volgt te omschrijven

$$\frac{u^m_{i,j} - u^{m-1}_{i,j}}{\Delta t} \approx \alpha \frac{u^m_{i-1,j} + u^m_{i,j-1} - 4u^m_{i,j} + u^m_{i+1,j} + u^m_{i,j+1}}{h^2} + f^m_{i,j} \tag{9}$$

Oftewel

$$u_{i,j}^{m-1} + f_{i,j}^m = u_{i,j}^m + \frac{\alpha\Delta t}{h^2}(-u_{i-1,j}^m - u_{i,j-1}^m + 4u_{i,j}^m - u_{i+1,j}^m - u_{i,j+1}^m) \tag{10}$$

De discrete vergelijking op deze punten kan dan worden gerepresenteerd met de stencilnotatie:

$$\begin{pmatrix} 0 & -p & 0 \\ -p & 1+4p & -p \\ 0 & -p & 0 \end{pmatrix}, \quad p = \frac{\alpha\Delta t}{h^2} \tag{11}$$

De middelste rij geeft hier de benadering aan van het punt met de punten links en rechts van het te benaderen punt en de midelste kolom representeert de benadering met de punten onder en boven het te benaderen punt.
Voor verdere uitwerking van het gediscretiseerde porbleem zal eerst gekeken worden naar de randvoorwaarden.

2.1 Dirichlet randvoorwaarden

Dirichlet randvoorwaarden worden ook wel gefixeerde randvoorwaarden genoemd, dit vanwege de vorm. Dirichlet randvoorwaarden 'fixeerd' namelijk de rand en geeft het een constante waarde. Bij de warmtevergelijking 1 zeggen Dirichlet randvoorwaarden iets over de temperatuur van de rand. Randvoorwaarden van de volgende vorm worden Dirichlet randvoorwaarden genoemd:

$$\begin{cases} u(0,y,t) = u_{1,j}^m & = b_{1,j} & \text{(rechterkant)} \\ u(L,y,t) = u_{N+1,j}^m & = b_{N+1,j} & \text{(linkerkant)} \\ u(x,0,t) = u_{i,1}^m & = b_{i,1} & \text{(onderkant)} \\ u(x,H,t) = u_{i,N+1}^m & = b_{i,N+1} & \text{(bovenkant)} \end{cases}$$

Er zijn twee manieren om te zorgen dat het discrete probleem voldoet aan de Dirichlet randvoorwaarden. De eerste methode is zonder eliminatie van de randvoorwaarden, hierbij worden de onbekende randpunten deel van het lineaire stelsel. Er wordt voor elk punt op de Dirichlet rand een vergelijking toegevoegd waarbij op al deze punten de randwaarden $b_{i,j}^m$ aan de bronterm $f_{i,j}^m$ wordt toegevoegd en de punten het volgende stencil krijgen toegewezen:

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{12}$$

Deze vergelijkingen kunnen vervolgens weer gecombineerd worden met de vergelijkingen voor de inwendige punten. Dan ontstaan er $(N+1)^2$ lineaire vergelijkingen voor de $(N+1)^2$ onbekenden $\{u_{i,j}^m \mid 1 \leq i,j \leq N+1\}$ waar uiteindelijk de volledige stelselmatrix mee kan worden gedefinieerd. Om de symmetrie in deze stelselmatrix te bewaren moet het gewicht $\frac{\alpha\Delta t}{h^2}$ van een inwendig punt (i,j) naar een randpunt van de rechterkant worden gebracht. Voor een inwendig punt $(2,j)$ verbonden met de linkerrand $x=0$ wordt $f_{2,j}^m$ overschreven door $f_{2,j}^m + \frac{\alpha\Delta t}{h^2} b_{1_j}$ en het stencil van dit punt komt er als volgt uit te zien:

$$\begin{pmatrix} 0 & -p & 0 \\ 0 & 1+4p & -p \\ 0 & -p & 0 \end{pmatrix} \tag{13}$$

Voor een inwendig punt in een hoek waarnaast zich aan twee kanten een randpunt bevindt, zoals het punt $(2,2)$ met aan de linkerkant en de onderkant een randpunt wordt $f_{2,2}^m$ overschreven door $f_{2,2}^m + \frac{\alpha\Delta t}{h^2} b_{1,2} + \frac{\alpha\Delta t}{h^2} b_{2,1}$ en het stencil van dit punt wordt:

$$\begin{pmatrix} 0 & -p & 0 \\ 0 & 1+4p & -p \\ 0 & 0 & 0 \end{pmatrix} \tag{14}$$

De tweede methode is met eliminatie van de randvoorwaarden, hierbij worden de inwendige punten dicht bij de rand op dezelfde manier vervangen als bij de eerste methoden en worden geen stencils voor de randpunten toegevoegd. Deze methode resulteerd in $(N-1)^2$ vergelijkingen voor de onbekenden $\{u_{i,j}^m \mid 2 \leq i,j \leq N\}$.

2.2 Neumann randvoorwaarden

In het geval van Neumann randvoorwaarden moeten er wel stencils voor de randpunten worden gedefinieerd. Neumann randvoorwaarden zeggen wat over de normale afgeleiden van het probleem, in het geval van de warmtevergelijking geven Neumann randvoorwaarden iets weer over de warmteoverdracht aan de rand en zijn daarom van de volgende vorm:

$$\frac{\partial u(x,y)}{\partial n} = \nabla u(x,y) \cdot \mathbf{n} = b(x,y) \tag{15}$$

Om een Neumann rand te discretiseren wordt onderscheid gemaakt tussen de hoekpunten en de overige randpunten. Voor de linkerrand waar $x = 0$ kan het randpunt $u^m_{1,j}$ met $2 \leq j \leq N$ worden benaderd door een fictief punt $u^m_{0,j}$ die een afstand h links van het randpunt ligt, te definiëren. De normale afgeleide kan nu worden benaderd met behulp van de gecentreerde afstandsformules:

$$\frac{\partial u}{\partial n}(0,y_j) = -\frac{\partial u}{\partial x}(0,y_j) = -\frac{u^m_{2,j} - u^m_{0,j}}{2h} + \mathcal{O}(h^2) \tag{16}$$

Door nu de randvoorwaarden hier in te verwerken krijgt $u^m_{0,j}$ de volgende vergelijking:

$$u^m_{0,j} = 2h \cdot b_{1,j} + u^m_{2,j} \tag{17}$$

Het stencil voor de randpunten $u^m_{1,j}$ met $2 \leq j \leq N$ wordt nu:

$$\begin{pmatrix} 0 & p & 0 \\ 0 & 1+4p & -2p \\ 0 & -p & 0 \end{pmatrix} \tag{18}$$

Terwijl $f^m_{1,j}$ moet worden overschreven door $f^m_{1,j} + \frac{2\alpha\Delta t}{h} b_{1,j}$. Om symmetrie in de stelselmatrix te bewaren moet de vergelijking voor dit punt door twee worden gedeeld en zo ontstaat het volgende stencil:

$$\begin{pmatrix} 0 & -\frac{1}{2}p & 0 \\ 0 & \frac{1}{2}+2p & p \\ 0 & -\frac{1}{2}p & 0 \end{pmatrix} \tag{19}$$

De hoekpunten worden op dezelfde manier gedefinieerd, maar nu gebeuren de stappen in zowel de x- als de y-richting. Hierdoor wordt aan het einde niet door twee maar door vier gedeeld. Kijk bijvoorbeeld naar het hoekpunt $u^m_{1,1}$. Hiervoor moet de bronterm $f^m_{1,1}$ overschreven worden door $\frac{1}{4} f^m_{1,1} + \frac{\alpha\Delta t}{h} b_{1,1}$ en wordt het stencil als volgt gedefinieerd:

$$\begin{pmatrix} 0 & -\frac{1}{2}p & 0 \\ 0 & \frac{1}{4}+p & -\frac{1}{2}p \\ 0 & 0 & 0 \end{pmatrix} \tag{20}$$

2.3 Lexicografische ordening

Om uiteindelijk de stelselmatrix te definiëren voor de onbekende punten is het gunstig om eerst de volgorde van de punten te definiëren. Er wordt hier gebruik gemaakt van lexicographische ordening zonder eliminatie van de randvoorwaarden, deze ordening kan dus gebruikt wordt voor zowel Dirichlet als Neumann randvoorwaarden.

De punten krijgen nu een globale index toegewezen: het punt (i,j) krijgt index $I = i + (j-1)(N+1)$ voor $1 \leq i,j \leq N+1$. Zodat alle $(N+1)^2$ een index toegewezen krijgen. Nu kunnen de onbekende waarden

van $u_{i,j}^m$ en $f_{i,j}^m$ in kolomvectoren $\mathbf{u}^m$ en $\mathbf{f}^m$ van lengte $(N+1)^2$ worden opgeslagen zodat:

$$\mathbf{u}^m = \begin{pmatrix} u_{1,1}^m \\ u_{1,2}^m \\ \vdots \\ u_{1,N+1}^m \\ u_{2,1}^m \\ \vdots \\ u_{2,N+1}^m \\ \vdots \\ u_{N+1,1}^m \\ \vdots \\ u_{N+1,N+1}^m \end{pmatrix} \qquad \mathbf{f}^m = \begin{pmatrix} f_{1,1}^m \\ f_{1,2}^m \\ \vdots \\ f_{1,N+1}^m \\ f_{2,1}^m \\ \vdots \\ f_{2,N+1}^m \\ \vdots \\ f_{N+1,1}^m \\ \vdots \\ f_{N+1,N+1}^m \end{pmatrix}.$$

Nu moet alleen nog de stelselmatrix gedefinieerd worden om het probleem om te zetten naar een lineaire stelsel. De discrete differentiaalmatrix A is afhankelijk van de dimensie en de randvoorwaarden van het probleem.

2.3.1 De stelselmatrix voor de eendimensionele warmtevergelijking

In het geval van Dirichlet randvoorwaarden geldt:

$$A = \begin{pmatrix} 1 & 0 & \cdots & \cdots & \cdots & 0 \\ -p & 1+2p & -p & 0 & \cdots & 0 \\ 0 & -p & 1+2p & -p & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -p & 1+2p & -p \\ 0 & \cdots & \cdots & 0 & 0 & 1 \end{pmatrix} \in \mathcal{R}^{(N+1)\times(N+1)}. \tag{21}$$

Deze matrix kan symmetrisch worden gemaakt door de verbanden tussen de meest linker en meest rechter inwendige punten en de linker en rechter randen te vertalen naar de meest rechtervector zodat

$$A = \begin{pmatrix} 1 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & 1+2p & -p & 0 & \cdots & 0 \\ 0 & -p & 1+2p & -p & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -p & 1+2p & 0 \\ 0 & \cdots & \cdots & 0 & 0 & 1 \end{pmatrix} \in \mathcal{R}^{(N+1)\times(N+1)}. \tag{22}$$

Met bronterm

$$\mathbf{f}^m = \begin{pmatrix} f_1^m \\ f_2^m + \frac{\alpha\Delta t}{h^2} b_1 \\ f_3^m \\ \vdots \\ f_{N-1}^m \\ f_N^m + \frac{\alpha\Delta t}{h^2} b_{N+1} \\ f_{N+1}^m \end{pmatrix} \in \mathcal{R}^{(N+1)\times(N+1)}. \tag{23}$$

In het geval van Neumann randvoorwaarden geldt:

$$A = \begin{pmatrix} p & -p & \cdots & \cdots & \cdots & 0 \\ -p & 1+2p & -p & 0 & \cdots & 0 \\ 0 & p & 1+2p & -p & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -p & 1+2p & -p \\ 0 & \cdots & \cdots & 0 & -p & p \end{pmatrix} \in \mathcal{R}^{(N+1)\times(N+1)}. \tag{24}$$

Met bronterm

$$\mathbf{f}^m = \begin{pmatrix} \frac{1}{2}f_1^m + \frac{\alpha\Delta t}{h}b_1 \\ f_2^m \\ \vdots \\ f_N^m \\ \frac{1}{2}f_{N+1}^m + \frac{\alpha\Delta t}{h}b_{N+1} \end{pmatrix} \in \mathcal{R}^{(N+1)\times(N+1)}. \tag{25}$$

2.3.2 De stelselmatrix voor de tweedimensionele warmtevergelijking

In het tweedimensionele geval wordt A opgebouwd uit meerdere matrices. Daarvoor moeten eerste de volgende matrices gedefineerd worden: I_{N-1}^m als de identiteitsmatrix op $\mathcal{R}^{(N-1)\times(N-1)}$, I_{N+1}^m als de identiteitsmatrix op$\mathcal{R}^{(N+1)\times(N+1)}$, de matrix T^m

$$T^m = \begin{pmatrix} 1+4p & -p & \cdots & \cdots & \cdots & 0 \\ -p & 1+4p & -p & 0 & \cdots & 0 \\ 0 & -p & 1+4p & -p & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -p & 1+4p & -p \\ 0 & \cdots & \cdots & 0 & -p & 1+4p \end{pmatrix} \in \mathcal{R}^{(N-1)\times(N-1)}$$

en de matrices $\hat{I}^m$ en $\hat{T}^m$

$$\hat{I}^m = \begin{pmatrix} 0 & 0 & 0 \\ 0 & I_{N-1}^m & 0 \\ 0 & 0 & 0 \end{pmatrix} \in \mathcal{R}^{(N+1)\times(N+1)} \qquad \hat{T}^m = \begin{pmatrix} 1 & 0 & 0 \\ 0 & T^m & 0 \\ 0 & 0 & 1 \end{pmatrix} \in \mathcal{R}^{(N+1)\times(N+1)}.$$

In het geval van Dirichlet randvoorwaarden geldt nu

$$A = \begin{pmatrix} I_{N+1}^m & 0 & \cdots & \cdots & \cdots & 0 & 0 \\ 0 & \hat{T}^m & -p\hat{I}^m & 0 & \cdots & \cdots & 0 \\ 0 & -p\hat{I}^m & \hat{T}^m & -p\hat{I}^m & & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & & \vdots \\ 0 & \cdots & 0 & -p\hat{I}^m & \hat{T}^m & -p\hat{I}^m & 0 \\ 0 & \cdots & \cdots & 0 & -p\hat{I}^m & \hat{T}^m & 0 \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & I_{N+1}^m \end{pmatrix} \in \mathcal{R}^{(N+1)^2\times(N+1)^2}. \tag{26}$$

en in het geval van Neumann randvoorwaarden geldt

$$A = \begin{pmatrix} pI^m_{N+1} & -p\hat{I}^m & \cdots & \cdots & \cdots & 0 & 0 \\ -p\hat{I}^m & \hat{T}^m & -p\hat{I}^m & 0 & \cdots & \cdots & 0 \\ 0 & -p\hat{I}^m & \hat{T}^m & -p\hat{I}^m & & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & & \vdots \\ 0 & \cdots & 0 & -p\hat{I}^m & \hat{T}^m & -p\hat{I}^m & 0 \\ 0 & \cdots & \cdots & 0 & -p\hat{I}^m & \hat{T}^m & -p\hat{I}^m \\ 0 & \cdots & \cdots & \cdots & \cdots & -p\hat{I}^m & pI^m_{N+1} \end{pmatrix} \in \mathcal{R}^{(N+1)^2 \times (N+1)^2}. \tag{27}$$

Voor het gemak wordt er een oplossingsvector $\mathbf{s}^m = \mathbf{u}^{m-1} + \mathbf{f}^m$ gedefinieerd. Nu kan de oplossing van de warmtevergelijking worden benaderd door het lineaire stelsel $A\mathbf{u}^m = \mathbf{s}^m$ op te lossen. Om de oplossing zo accuraat mogelijk te laten zijn, is het gunstig om een grote N te nemen. Vanwege de grote van het stelsel bij een grote N zullen iteratieve methoden gebruikt worden om de oplossing van het stelsel te benaderen.

3 Iteratieve methoden

Iteratieve methoden benaderen de oplossing door middel van 'trial and error', er wordt eerst een beginoplossing bepaald die vervolgens wordt geanalyseerd en zo wordt met de eerder geschatte oplossing een betere schatting bepaald. Dit gebeurt een aantal stappen waarbij bij een effectieve methode per iteratie de schatting steeds dichter bij de echte oplossing komt. Vanwege de notatie zal in dit hoofdstuk de tijds- en plaatsindex achterwege worden gelaten.

De rij iteraties wordt genoteerd als $(\mathbf{u}^k)_{k\geq 0}$, de beginwaarde als $\mathbf{u}^0$ en de exacte, te benaderende oplossing als $\mathbf{u}^*$ waarbij voor $k \to \infty \quad \mathbf{u}^k \to \mathbf{u}^*$. De fout van de benadering is gelijk aan de afstand tussen de exacte oplossing en de benadering, dit geeft een foutvector

$$\mathbf{e}^k = \mathbf{u}^* - \mathbf{u}^k. \tag{28}$$

De kwaliteit van de oplossing op de k^e iteratie wordt aangegeven met de residuvector

$$\mathbf{r}^k = \mathbf{s} - A\mathbf{u}^k. \tag{29}$$

De volgende relatie tussen de fout en de residuvector geldt

$$A\mathbf{e}^k = \mathbf{r}^k \tag{30}$$

In dit onderzoek zal de kwaliteit van de oplossing op de k^e iteratie worden aangegeven met de relatieve fout $\mathbf{r_f}$ gedefinieerd door

$$\mathbf{r_f}^k = \frac{\|\mathbf{r}^k\|_2}{\|\mathbf{s}\|_2} \tag{31}$$

De stelselmatrix kan worden opgesplitst in de niet-singuliere matrix M en een matrix $N = M - A$ zodat $A = M - N$. Nu kan het lineare stelsel $A\mathbf{u} = \mathbf{s}$ geschreven worden als $M\mathbf{u} = N\mathbf{u} + \mathbf{s}$ en dit geeft het volgende iteratieve schema

$$\begin{aligned} \mathbf{u}^{k+1} &= M^{-1}N\mathbf{u}^k + M^{-1}\mathbf{s} \\ &= M^{-1}(M - A)\mathbf{u}^k + M^{-1}\mathbf{s} \\ &= \mathbf{u}^k + M^{-1}(\mathbf{s} - A\mathbf{u}^k) \\ &= \mathbf{u}^k + M^{-1}\mathbf{r}^k \end{aligned} \tag{32}$$

Voor de foutvector geldt

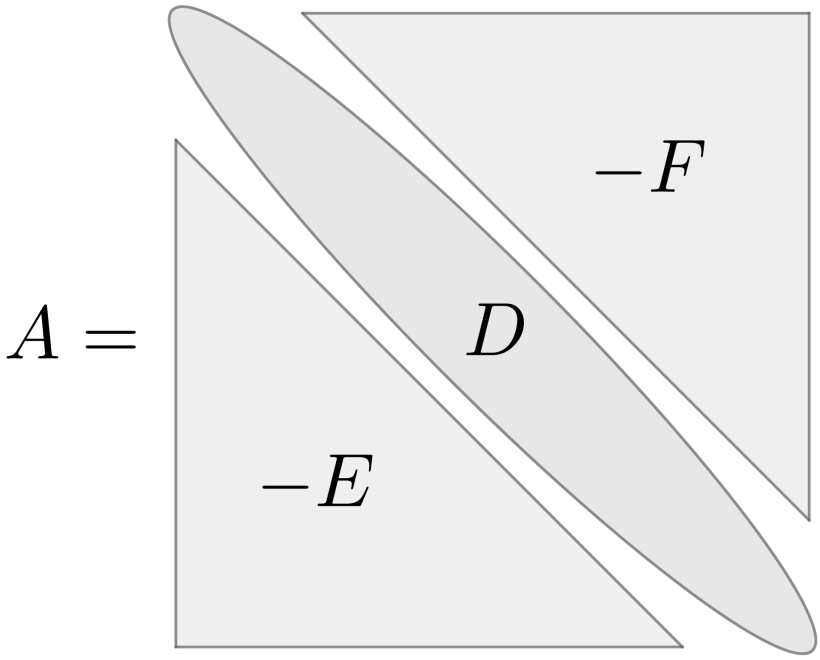
$$\begin{aligned} \mathbf{e}^{k+1} &= \mathbf{u}^* - \mathbf{u}^{k+1} \\ &= \mathbf{u}^* - \mathbf{u}^k - M^{-1}\mathbf{r}^k) \\ &= \mathbf{e}^k - M^{-1}A\mathbf{e}^k \\ &= (I - M^{-1}A)\mathbf{e}^k. \end{aligned} \tag{33}$$

en voor de residuvector geldt

$$\begin{aligned}\mathbf{r}^{k+1} &= \mathbf{s} - A\mathbf{u}^{k+1} \\ &= \mathbf{s} - A\mathbf{u}^{k} - AM^{-1}\mathbf{r}^{k} \\ &= \mathbf{r}^{k} - AM^{-1}\mathbf{r}^{k} \\ &= (I - AM^{-1})\mathbf{r}^{k}.\end{aligned} \tag{34}$$

De iteratiematrix $B = I - M^{-1}A$ kan nu worden gebruikt voor het schatten van de kwaliteit van het iteratieve schema.

Voordat er naar specifieke methoden wordt gekeken, zal eerst de stelselmatrix worden opgedeeld in een diagonaal- en twee stricte driehoeksmatrices zodat $A = D - E - F \in \mathbb{R}^{n\times n}$



Figuur (2) Opdelen van de stelselmatrix A zodat $A = D - E - F$

3.1 De Jacobi methode

De Jacobi methode definieert de diagonaalmatrix van de stelselmatrix als de niet-singuliere matrix zodat:

$$\left.\begin{aligned} M &= D \\ N &= E + F \end{aligned}\right\} \qquad \mathbf{u}^{k+1} = D^{-1}((E+F)\mathbf{u}^{k} + \mathbf{s}) \tag{35}$$

De kentallen van $\mathbf{u}$ kunnen nu per iteratie berekend worden door:

$$u_i^{k+1} = \frac{1}{a_{ii}}\left(s_i - \sum_{j=1, j\neq i}^{n} a_{ij}u_j^{k}\right) \tag{36}$$

Voorbeeld

Ter illustratie zal het volgende lineaire stelsel worden opgelost met behulp van de Jacobi methode.

$$\begin{cases} 6x_1 + x_2 + x_3 = 11 \\ x_1 + 4x_2 + x_3 = 12 \\ -x_1 - x_2 + 3x_3 = 6 \end{cases} \tag{37}$$

In de eerste vergelijking x_1, in de tweede vergelijking x_2 en in de derde vergelijking x_3 isoleren geeft het volgende iteratieve schema:

$$\begin{cases} x_1^{k+1} = \dfrac{1}{6}(11 - x_2^k - x_3^k) \\ x_2^{k+1} = \dfrac{1}{4}(12 - x_1^k - x_3^k) \\ x_3^{k+1} = \dfrac{1}{3}(6 + x_1^k + x_2^k) \end{cases} \tag{38}$$

Dit is gelijk aan de Jacobi methode. Voor grotere stelsels en voor implementatie in MATLAB is het gunstig om gebruik te maken van de matrixnotatie.

$$\begin{pmatrix} 6 & 1 & 1 \\ 1 & 4 & 1 \\ -1 & -1 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 11 \\ 12 \\ 6 \end{pmatrix} \tag{39}$$

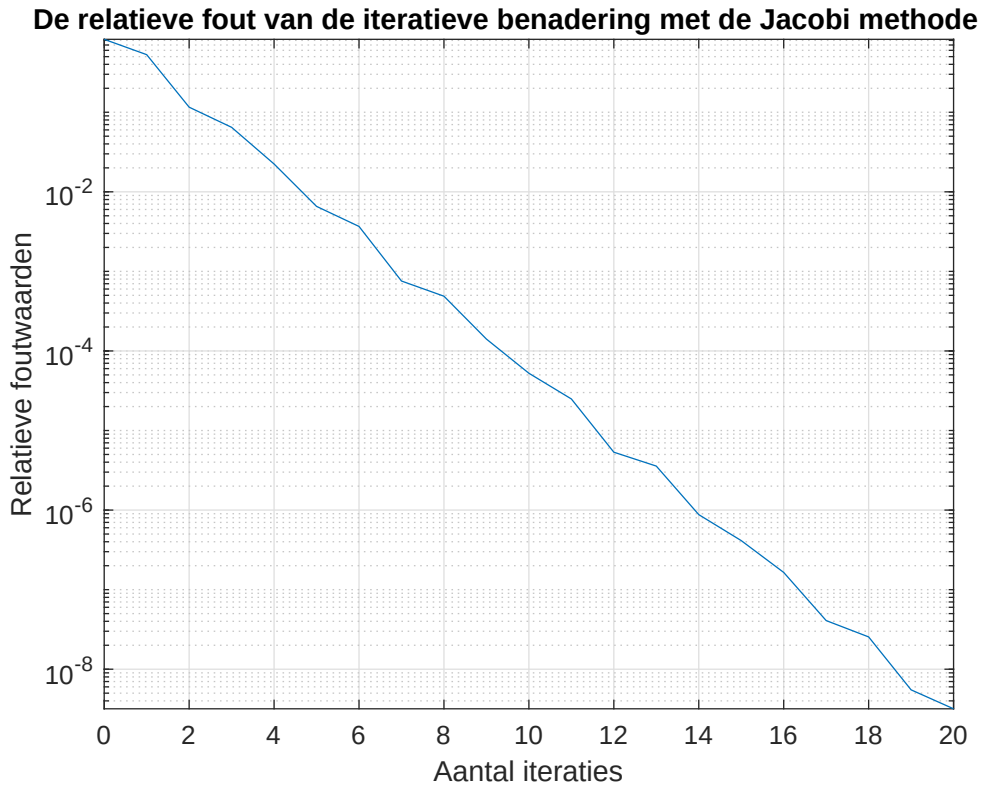
Voor de Jacobi methode geldt voor de matrices M en N:

$$M = D = \begin{pmatrix} 6 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 3 \end{pmatrix} \qquad\qquad N = E + F = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ -1 & -1 & 0 \end{pmatrix}$$

zodat

$$\begin{aligned} \mathbf{u}^{k+1} &= \begin{pmatrix} \frac{1}{6} & 0 & 0 \\ 0 & \frac{1}{4} & 0 \\ 0 & 0 & \frac{1}{3} \end{pmatrix} \left(\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ -1 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} 11 \\ 12 \\ 6 \end{pmatrix} \right) \\ &= \begin{pmatrix} \frac{1}{6}(x_2 + x_3 + 11) & 0 & 0 \\ 0 & \frac{1}{4}(x_1 + x_3 + 12) & 0 \\ 0 & 0 & \frac{1}{3}(-x_1 - x_2 + 6) \end{pmatrix} \end{aligned} \tag{40}$$

Door het nulpunt als beginwaarde te nemen geeft dit het volgende resultaat voor de relatieve fout van de oplossing per iteratie.



Figuur (3) De relatieve fout per iteratie van de te benaderende oplossing van het voorbeeldstelsel met de Jacobi methode met beginwaarde $(0,0,0)^T$

Voor dit onderzoek zal een oplossing een goede benadering worden genoemd wanneer de relatieve fout kleiner is dan 10^{-6}. De benadering zal hiervoor moeten convergeren naar de oplossing. Aan de grafiek is te zien dat de Jacobi methode de oplossing van het stelsel vanaf de 14^{e} iteratie zo goed benaderd. Dit stelsel kan dus binnen 14 iteraties worden opgelost met de Jacobi methode. De volgende sectie zal verder ingaan op de convergentie van de iteratieve fout van de Jacobi methode.

3.1.1 Convergentie van de Jacobi methode

Wegens de eerder bekeken formule voor de foutvector in formule 33, geldt voor de iteratieve fout:

$$\begin{aligned}\mathbf{e}^k &= (I - M^{-1}A)\mathbf{e}^{k-1} \\ &= (I - M^{-1}A)^2\mathbf{e}^{k-2} \\ &= \dots \\ &= (I - M^{-1}A)^k\mathbf{e}^0 \\ &= B^k\mathbf{e}^0.\end{aligned} \tag{41}$$

De fout na k iteraties kan dus berekend door de iteratiematrix $B = I - M^{-1}A$ k keer toe te passen. Hieruit volgt dat de Jacobi methode convergeert voor rijdiagonaaldominante matrices met een snellere convergentie voor sterkere dominantie. Dit wordt intuïtief duidelijk door B te analyseren.
Laat A een willekeurige matrix $n \times n$ zijn.

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \tag{42}$$

Gegeven de Jacobi methode geldt $M = D$, dus geldt nu voor de iteratiematrix:

$$\begin{aligned} B &= I - M^{-1}A = I - D^{-1}A \\ &= \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} - \begin{pmatrix} \frac{1}{a_{11}} & 0 & \cdots & 0 \\ 0 & \frac{1}{a_{22}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{a_{nn}} \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \\ &= - \begin{pmatrix} 0 & \frac{a_{12}}{a_{11}} & \cdots & \frac{a_{1n}}{a_{11}} \\ \frac{a_{21}}{a_{22}} & 0 & \cdots & \frac{a_{2n}}{a_{22}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{a_{n1}}{a_{nn}} & \frac{a_{n2}}{a_{nn}} & \cdots & 0 \end{pmatrix} \end{aligned} \tag{43}$$

Laat A nu sterk rijdiagonaaldominant zijn. Dan geldt

$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}| \qquad \forall i = 1, \ldots, n.$$

Gezien de kentallen van B zal B^k nu naar de nulmatrix convergeren en dus zal de iteratieve fout naar nul convergeren.

Dat de convergentie van de Jacobi methode afhankelijk is van rijdiagonaaldominantie van de stelselmatrix A kan ook worden bewezen met behulp van de spectrale straal en een aantal stellingen.

Definitie 1. *De spectrale straal $\rho(A)$ van een matrix $A \in \mathbb{R}^{n\times n}$ is gedefinieerd als*

$$\rho(A) = \max_{i=1,\ldots,n} \{|\lambda_i| : \lambda_i \in \sigma(A)\}$$

waarbij $\sigma(A)$ de verzameling van eigenwaarden van A is.

Stelling 1. *Gegeven een willekeurige submultiplicatieve matrixnorm $\|\cdot\|$, dan geldt*

$$\rho(A) \leq \|A\|.$$

Bewijs. Laat λ een willekeurige eigenwaarde van A zijn met eigenvector $\mathbf{u}$. Dan geldt $A\mathbf{u} = \lambda\mathbf{u}$ en vanwege de submultiplicatieve eigenschap volgt

$$\|\lambda\mathbf{u}\| = |\lambda|\|\mathbf{u}\| = \|A\mathbf{u}\| \leq \|A\|\|\mathbf{u}\| \Rightarrow |\lambda| \leq \|A\|.$$

Aangezien λ willekeurig geldt dit voor iedere $|\lambda_i| : \lambda_i \in \sigma(A)$ □

Stelling 2. *Gegeven een matrix $A \in \mathbb{R}^{n\times n}$*

$$\rho(A) < 1 \Leftrightarrow \lim_{k\to\infty} A^k = \emptyset_{n\times n}.$$

Bewijs. $\Rightarrow$ Laat J de Jordaan normaalvorm zijn van de matrix A zodat er een inverteerbare matrix P is waarvoor geldt $A = PJP^{-1}$. Dan geldt voor de k^e macht van A dat $A^k = PJ^kP^{-1}$ zodat als $\rho(A) < 1$ geldt dat $lim_{k\to\infty}A^k = \emptyset_{n\times n}$
$\Leftarrow$ Laat $\mathbf{u}_n$ een genormaliseerde eigenvector zijn van de eigenwaarde met maximale modulus zodat $A\mathbf{u}_n = \lambda_n\mathbf{u}_n$. Oftewel $A^k\mathbf{u}_n = \lambda_n^k\mathbf{u}_n$. Door aan beiden kanten de tweenorm te nemen volgt nu

$$|\lambda_n|^k = \|A^k\mathbf{u}_n\|_2 \to 0.$$

Hieruit volgt dat $\rho(A) < 1$. □

Hieruit volgt direct de volgende stelling voor de iteratiematrix.

Stelling 3.

$$\rho(B) = \rho(I - M^{-1}A) < 1 \Leftrightarrow \{\mathbf{u}^k\}_{k=1}^{\infty} \textit{ convergeert.}$$

De I-norm van de iteratiematrix van de Jacobi methode wordt gegeven door

$$\|B\|_1 = \max_{1 \leq j \leq n} \sum_{i=1, i \neq j}^{n} \frac{|a_{ij}|}{|a_{ii}|}.$$

Nu volgt uit stellingen 1 en 3 dat de benadering van de oplossing met de Jacobi methode convergeert als

$$\sum_{j=1, j \neq i}^{n} |a_{ij}| < |a_{ii}| \quad \forall i = 1, \ldots, n.$$

Oftewel wanneer de stelselmatrix A rijdiagonaaldominant is. Bovendien volgt dat de convergentie van de een algemene iteratieve methode afhankelijk is van de eigenwaarden van de iteratiematrix B.

3.2 Von Neumann analyse

Met behulp van de Von Neumann analyse toegepast op de spectrale straal van B kan de convergentie van algemene methoden worden geanalyseerd. De methode is gebaseerd op Fourier decompositie van de iteratieve fout.
Laat $v_1, v_2, \ldots, v_n$ een orthonormale basis van eigenvectoren zijn voor $\mathbb{R}^n$ en definieer $\mathbf{e}^0 = c_1 v_1 + c_2 v_2 + \cdots + c_n v_n$ met $c_1, \ldots, c_n \in \mathbb{R}$. Wegens 41 geldt nu

$$\begin{aligned} \mathbf{e}^k &= B^k \mathbf{e}^0 \\ \mathbf{e}^k &= c_1 \lambda_1^k v_1 + c_2 \lambda_2^k v_2 + \cdots + c_n \lambda_n^k v_n \end{aligned} \tag{44}$$

Hieraan is te zien dat de afname van de fout van een iteratieve methode afhankelijk is van de eigenwaarden van de iteratiematrix. De term die het langzaamst daalt is de term met de grootste absolute eigenwaarde λ_n. Voor grote k is het voldoende om alleen naar die term te kijken en kan de fout worden benaderd met:

$$\mathbf{e}^k \approx c_n \lambda_n^k v_n \tag{45}$$

Vanwege de constante diagonaal d van de stelselmatrix is een eigenvector van A correspondent met de eigenwaarde $\lambda(A)$ ook een eigenvector van de iteratiematrix B met eigenwaarde

$$\lambda(B) = 1 - \frac{1}{d}\lambda(A). \tag{46}$$

Laat nu

$$\mathbf{v}_m = \begin{pmatrix} sin(\pi m h) \\ sin(\pi m 2h) \\ \vdots \\ sin(\pi m (N-1) h) \end{pmatrix}$$

een eigenvector zijn van A. De derde rij van de stelselmatrix geeft nu:

$$\begin{aligned} A\mathbf{v}_m &= -psin(\pi m h) + (1+2p)sin\pi m 2h) - psin(\pi m 3h) \\ &= -p(sin(\pi m h) + sin(\pi m 3h)) + (1+2p)sin(\pi m 2h) \\ &= -p(sin(\pi m 2h - \pi m h) + sin(\pi m 2h + \pi m h)) + (1+2p)sin(\pi m 2h) \\ &= -p(2sin(\pi m 2h)cos(\pi m h)) + (1+2p)sin(\pi m 2h) \\ &= sin(\pi m 2h)(1 + 2p - 2pcos(\pi m h)). \end{aligned} \tag{47}$$

Dit geeft dus de volgende eigenwaarde

$$\lambda = 1 + 2p - 2pcos(\pi m h) \tag{48}$$

Waarbij gebruik wordt gemaakt van de volgende goniometrische som- en verschilformules:

$$\begin{aligned} sin(\alpha+\beta) &= sin(\alpha)sin(\beta)+cos(\alpha)cos(\beta) \\ sin(\alpha-\beta) &= sin(\alpha)sin(\beta)-cos(\alpha)cos(\beta). \end{aligned} \tag{49}$$

De overige interne punten van het stelsel geven vergelijkbare berekeningen. De eigenwaarden van de stelselmatrix een eendimensionele warmtevergelijking met Dirichlet randvoorwaarden worden nu gegeven door:

$$\lambda_m(A) = 1 + 2p - 2pcos(\pi mh).$$

Voor de eigenwaarden van de bijbehorende iteratiematrix B geldt nu vanwege vergelijking 46:

$$\lambda_m(B) = 1 - \frac{1}{1+2p}\lambda_m(A)$$

Met behulp van deze eigenwaarden kan een amplificatiefactor $\mathbf{X}$ gedefinieerd worden waarvoor geldt:

$$e^{k+1} = \mathbf{X}(m)e^k \qquad \text{met } e^k \text{ de fout in de k-de iteratie.} \tag{50}$$

De amplificatiefactor van de gehele Jacobi methode is gelijk aan de grootste absolute eigenwaarde van B. Voor het eendimensionele geval met Dirichlet randvoorwaarden geldt:

$$1 \leq |\lambda_m(A)| \leq 1 + 4p.$$

Oftewel

$$|\lambda_m(B)| \leq \frac{2p}{1+2p}.$$

Om de Jacobi methode te optimaliseren moet de amplificatiefactor voor elke eigenwaarde zo klein mogelijk worden gemaakt. Hiervoor zal dus worden gekeken naar de amplificatiefactor per eigenwaarde

$$\begin{aligned} \mathbf{X}_{JAC}(m) &= 1 - \frac{1}{1+2p}(1+2p-2pcos(\pi mh)) \\ &= \frac{2p}{1+2p}cos(\pi mh) \end{aligned} \tag{51}$$

Hieruit volgt dat de Jacobi methode snel convergeert voor kleine waarden van $p = \frac{\alpha\Delta t}{h^2}$, zoals gedefinieerd in 11. Deze waarde is minimaal voor een kleine tijdstap in verhouding met een grote plaatsstap, oftewel een kleine gridgrootte.

3.3 De Scheduled Relaxation Jacobi (SRJ) methode

De Scheduled Relaxation Jacobi (SRJ) methode is een iteratieve methode gebaseerd op de Jacobi methode waarbij relaxatiegewichten worden gebruikt om de convergentie van de methode te versnellen. In 'Acceleration of the Jacobi iterative method by factors exceeding 100 using scheduled relaxation' beschrijven Yang en Mittal de SRJ methode en het vinden van de optimale parameters voor elliptische vergelijkingen (2014). Om deze methode toe te passen op de warmtevergelijking zal eerst worden gekeken naar de gerelaxeerde Jacobi methode, ook wel de gedempte Jacobi methode genoemd.

3.3.1 De gedempte Jacobi methode

Definieer ω als het relaxatiegewicht en laat $\bar{\mathbf{u}}^{k+1}$ het iteratieve schema van de standaard Jacobi methode zijn, zoals omgeschreven in vergelijking . De gedempte Jacobi methode kan nu worden omgeschreven door

$$\mathbf{u}^{k+1} = (1-\omega)\mathbf{u}^k + \omega\bar{\mathbf{u}}^{k+1}. \tag{52}$$

Voor $\omega = 1$ is de methode gelijk aan de standaard Jacobi methode. Voor $\omega < 1$ krijgt de vorige iteratie een groter gewicht dan de met de Jacobi methode geschatte waarde, er wordt in dat geval gesproken over een onderrelaxatie. In het geval van $\omega > 1$ wordt gesproken over een overrelaxatie aangezien de geschatte waarde

een groter gewicht krijgt. Het uitwerken van vergelijking 52 met behulp van de definitie voor de residuvector 29 en de vergelijking voor de standaard Jacobi methode 35, geeft voor de gedempte Jacobi methode

$$\begin{aligned}\mathbf{u}^{k+1} &= (1-\omega)\mathbf{u}^k + \omega\bar{\mathbf{u}}^{k+1} \\ &= (1-\omega)\mathbf{u}^k + \omega D^{-1}((E+F)\mathbf{u}^k + s) \\ &= (1-\omega)\mathbf{u}^k + \omega D^{-1}s + \omega D^{-1}(E+F)\mathbf{u}^k \\ &= (1-\omega)\mathbf{u}^k + \omega D^{-1}\mathbf{r}^k + \omega D^{-1}A\mathbf{u}^k + wD^{-1}(E+F)\mathbf{u}^k \\ &= (1-\omega)\mathbf{u}^k + \omega D^{-1}\mathbf{r}^k + \omega\mathbf{u}^k \\ &= \mathbf{u}^k + \omega D^{-1}\mathbf{r}^k.\end{aligned} \tag{53}$$

zodat de niet-singuliere matrix en de iteratiematrix worden gegeven door

$$M = \frac{1}{\omega}D, \qquad\qquad B = I - \omega D^{-1}A.$$

Om het optimale relaxatiegewicht te vinden wordt gekeken naar de amplificatiefactor $\mathbf{X}(m)$ van de methode, voor de gedempte Jacobi methode geldt nu

$$\lambda(B) = 1 - \frac{\omega}{d}\lambda(A), \quad \text{met } \omega > 0. \tag{54}$$

Dit geeft de volgende amplificatiefactor voor de gedempte Jacobi methode:

$$\begin{aligned}\mathbf{X}_{dJAC}(m) &= 1 - \frac{\omega}{1+2p}(1+2p-2pcos(\pi mh)) \\ &= 1 - \omega + \frac{\omega\cdot 2p}{1+2p}cos(\pi mh), \quad \text{met } \omega > 0.\end{aligned} \tag{55}$$

Voor convergentie moet gelden $|X| < 1$ en het relaxatiegewicht is optimaal wanneer de absolute amplificatiefactor minimaal is, zodat de oplossing zo snel mogelijk convergeert. De gedempte Jacobi methode is het relaxatiegewicht dus optimaal voor

$$|\omega - \frac{\omega\cdot 2p}{1+2p}cos(\pi mh)| \to 1. \tag{56}$$

Door vergelijking 51 van de amplificatiefactor van de standaard Jacobi methode in te vullen geeft dit:

$$|\omega - \omega\mathbf{X}_{JAC}| \to 1. \tag{57}$$

De standaard Jacobi methode convergeert het snelst bij een minimale waarde voor p. Bij het optimaliseren van de Jacobi methode zal de gedempte Jacobi methode optimaal worden voor $|\omega| \to 1$. Hieruit volgt dat de standaard Jacobi methode optimaal is voor een warmtevergelijking waarbij de tijdstap in verhouding staat met de gridgrootte. Wanneer de verhouding tussen de tijdstap en de gridgrootte in de warmtevergelijking verschuift naar een grote tijdstap en een grotere gridgrootte, zal de vergelijking steeds minder afhankelijk worden van de tijd en de vorm krijgen van een elliptische vergelijking. Dan kan er een optimale ω worden gevonden waarvoor de gedempte Jacobi methode sneller convergeert dan de Jacobi methode. Voor de elliptische vergelijking is dit onderzocht en bewezen door Xiyang en Mittal (2014). Hierbij convergeert de SRJ methode niet zozeer sneller, maar convergeert de Jacobi methode voornamelijk veel langzamer aangezien de diagonaal van de stelselmatrix veel minder dominant wordt.

De SRJ methode past een cyclus van M iteraties van de gedempte Jacobi methode toe met verchillende relaxatiegewichten. Deze cyclus wordt herhaald tot convergentie. Het aantal verchillende relaxatiegewichten wordt het aantal levels van het SRJ schema genoemd, genoteerd met P. De verschillende relaxatiegewichten worden opgeslagen in de vector $\Omega = \{\omega_1, \omega_2, \ldots, \omega_P\}$ waarbij $\omega_1 > \omega_2 > \cdots > \omega_P$. Het aantal keer dat een relaxatiegewicht ω_i wordt herhaald in een SRJ cyclus wordt genoteerd met q_i, deze waarden worden opgeslagen in de vector $Q = \{q_1, q_2, \ldots, q_P\}$. Daarnaast geeft $\beta_i = \frac{q_i}{M}$ de verhouding tussen het aantal keer

dat een relaxatiegewicht voorkomt q_i en het aantal iteraties in de cyclus M waarbij $B = \{\beta_1, \beta_2, \ldots, \beta_P\}$. Een Scheduled Relaxation Jacobi schema wordt nu gedefinieerd door P, Ω en B. Hiervoor kunnen vervolgens optimale waarden berekend worden waarvoor de SRJ methode het snelst convergeert. Door Xiyang en Mittal (2014) zijn bovendien optimale waardes gevonden voor elliptische vergelijkingen. De verwachting is dat dezelfde waarden voor de warmtevergelijking waarbij een grote tijdstap in verhouding staat met een grote gridgrootte, de Jacobi methode optimaliseert. Bij een kleinere tijdstap die meer in verhouding staat met de gridstap zal naar verwachting de Jacobi methode optimaal zijn vanwege de amplificatiefactor van de gedempte Jacobimethode.

3.3.2 $P = 2$, $M = 2$ schema's

Een gesloten oplossing voor de optimale waarden kan alleen gevonden worden voor $P = 2$ en $M = 2$. Dit schema bestaat uit een cyclus van twee iteraties waardoor $q_1 = q_2 = 1$, immers wanneer $q_1 \neq q_2$ zou maar één relaxatiegewicht worden gebruikt en zou de methode gelijk zijn aan de gedempte Jacobi methode. Bovendien geldt dan $\beta_1 = \beta_2 = \frac{1}{2}$. In het geval van twee relaxatiegewichten worden de eigenwaarden van de iteratiematrix nu gegeven door

$$\lambda_m(B) = (1 - \omega_1 \frac{1}{d}\lambda_m(A))^{\frac{1}{2}}(1 - \omega_2 \frac{1}{d}\lambda_m(A))^{\frac{1}{2}}. \tag{58}$$

Definieer het golfgetal κ

$$\kappa = \frac{1}{d}\lambda_m(A) = \frac{1}{1+2p}(1 + 2p - 2pcos(\pi mh)). \tag{59}$$

Dan wordt de amplificatiefactor gegeven door

$$X = \sqrt{|1 - \omega_1\kappa||1 - \omega_2\kappa|} = \sqrt{|1 - \frac{\omega_2\kappa}{\eta}||1 - \omega_2\kappa|} \tag{60}$$

met $\eta = \frac{\omega_2}{\omega_1}$. Door te gebruiken dat het lokale minimum van de amplificatiefactor X gelijk is aan het lokale minimum van $\log X$ kunnen de volgende vergelijkingen gevonden worden om de optimale waarde van ω_1 en ω_2 te vinden. Zoals beschreven in [5].

$$\eta = \frac{-b + \sqrt{b^2 + 4}}{2} \qquad\qquad \omega_2 = \frac{\eta + 1}{2 + \kappa_{min}}, \tag{61}$$

$$\text{met} \qquad b = \frac{2\kappa_{min} - 6(2 + \kappa_{min})^2}{\kappa_{min} + (2 + \kappa_{min})^2} \tag{62}$$

3.3.3 Grotere schema's

Voor $P = 2$, $M > 2$ geldt het volgende. Neem $\Omega = \{\omega_1, \omega_2\}$ en $Q = \{q_1, q_2\}$. Er geldt $q_1 + q_2 = M$. Definieer nu $\beta_1 = \frac{q_1}{M} = \beta$ en $\beta_2 = \frac{q_2}{M} = \frac{M - q_1}{M} = 1 - \beta$. Voor de amplificatiefactor geldt

$$X = |1 - \omega_1\kappa|^{\beta}|1 - \omega_2\kappa|^{1-\beta}. \tag{63}$$

Hier kan op analoge wijze als bij kleinere M per gridgrootte optimale waarden van de relaxatiegewichten worden gevonden. Dit is al gebeurd door Yang(2014) en deze waarden zullen dus worden overgenomen.
Voor $P > 2$ schema's is de amplificatiefactor van de vorm:

$$X = \prod_{i=1}^{P} |1 - \omega_i\kappa|^{\beta_i}. \tag{64}$$

Om voor grotere schema's met meer relaxatiegewichten de optimale waarden te vinden, zal een groot stelsel van vergelijkingen moeten worden opgelost. Voor dit onderzoek zal worden gekeken naar $P = 5$ en hiervoor zullen wederom de al berekende optimale waarden uit Yang(2014) worden gehaald.

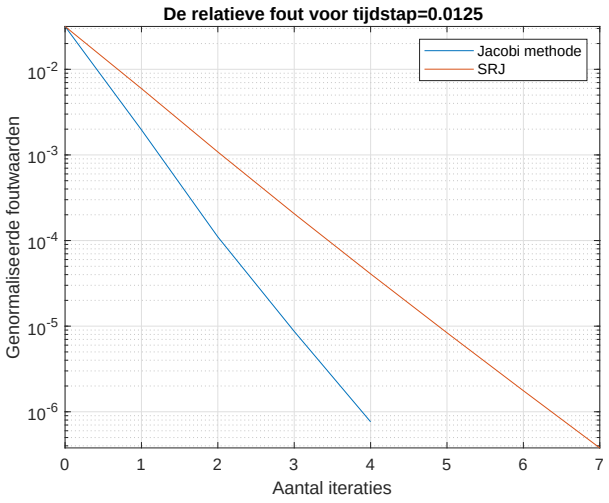
4 Resultaten

De eis voor het stopcriterium is dat de genormaliseerde foutwaarde kleiner is dan 10^{-6}. De genormaliseerde foutwaarde wordt berekend door het residu te delen door de norm van Au, waarbij u de oplossing van het stelsel en A de stelselmatrix is. Hieronder staan de resultaten voor de P=2, M=2 schema's waarbij de optimale relaxatiegewichten en parameters berekend zijn met de vergelijkingen 58 en 59. Voor schema's met grotere waarde van P of M werkt de SRJ methode niet voor de warmtevergelijking. De methode divergeert namelijk na enkele iteraties. Dit zou kunnen komen doordat de optimale waarde van de relaxatiegewichten en parameters berekend zijn voor tijdsonafhankelijke problemen. Deze waarden zijn blijkbaar niet geschikt voor een tijdsafhankelijk probleem zoals de warmtevergelijking.

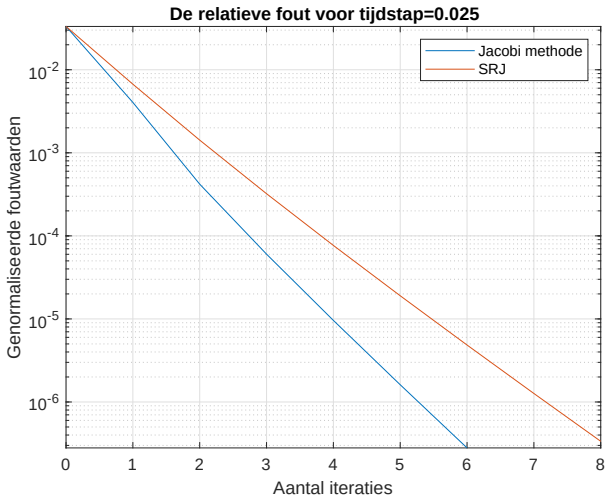
4.1 Resultaten voor de eendimensionele warmtevergelijking

4.1.1 Dirichlet randvoorwaarden

Ten eerste zal worden gekeken naar de genormaliseerde foutwaarde voor Dirichlet randvoorwaarden $u(0,t) = 30$ en $u(L,t) = 30$ met een gridgrootte van 512. De initiële benadering is gelijk aan de nulvector. Er zal worden gekeken naar de convergentiesnelheid van de foutwaarden voor verschillende tijdstappen. Aan de hand van de hierboven beschreven analyse is de verwachting dat de Jacobi methode sneller convergeeert voor een kleine tijdstap dan voor een grote tijdstap. Bovendien is de verwachting dat de Jacobi methode sneller convergeert dan de Scheduled Relaxation Jacobi methode. De SRJ methode zal waarschijnlijk echter minder in snelheid afnemen voor grotere tijdstappen dan de Jacobi methode waardoor de SRJ methode sneller zal zijn voor grote tijdstappen. Hierbij is vooral de verhouding van de tijdstap en de plaatsstap van belang. Voor een gridgrootte $N = 512$ geldt voor de plaatsstap $h = \frac{1}{512} \approx 0,002$.



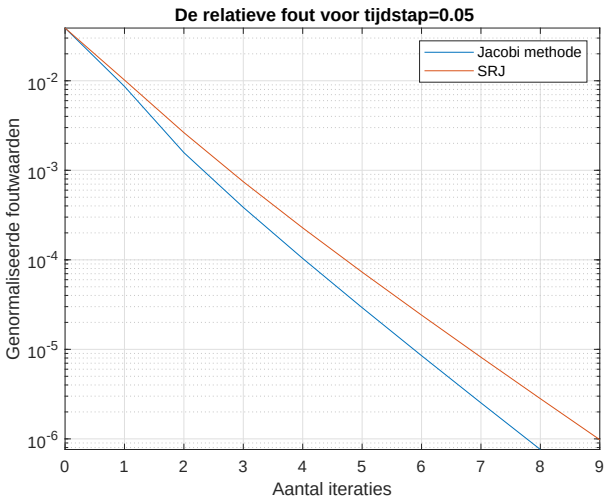
(a) De relatieve fout per iteratie van de te benaderende oplossing met een gridgrootte $N = 512$ en een tijdstap $\Delta t = 0,0125$ met Dirichlet randvoorwaarden.



(b) De relatieve fout per iteratie van de te benaderende oplossing met een gridgrootte $N = 512$ en een tijdstap $\Delta t = 0,025$ met Dirichlet randvoorwaarden.

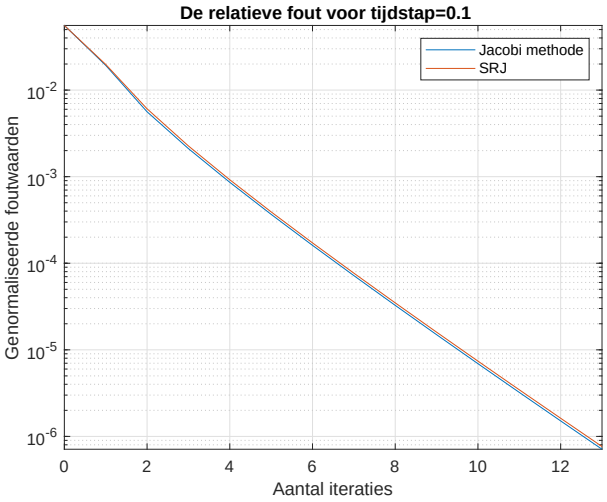
Methode	Iteraties	Genormaliseerde Fout
Jacobi	5	7.6342e-07
SRJ	8	3.7825e-07

Methode	Iteraties	Genormaliseerde Fout
Jacobi	7	2.7988e-07
SRJ	9	3.3611e-07



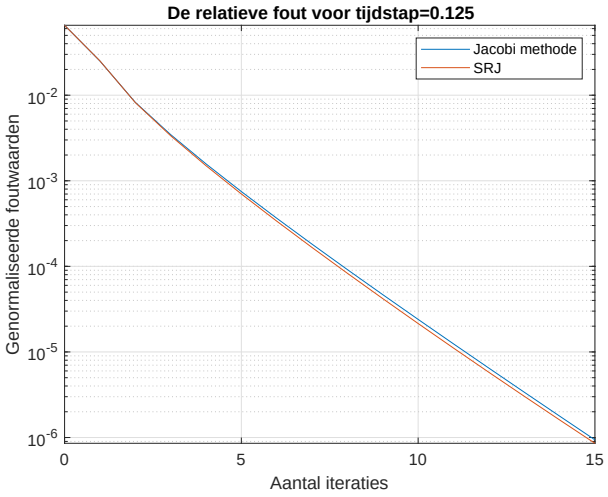
(a) De relatieve fout per iteratie van de te benaderende oplossing met een gridgrootte $N = 512$ en een tijdstap $\Delta t = 0,05$ met Dirichlet randvoorwaarden.

Methode	Iteraties	Genormaliseerde Fout
Jacobi	9	7.6093e-07
SRJ	10	9.7942e-07



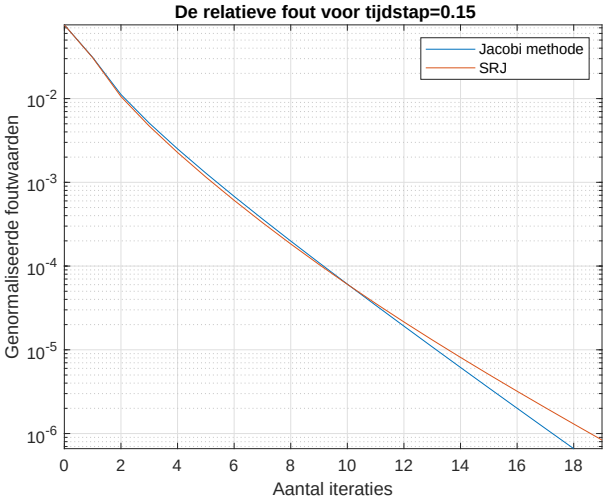
(b) De relatieve fout per iteratie van de te benaderende oplossing met een gridgrootte $N = 512$ en een tijdstap $\Delta t = 0,1$ met Dirichlet randvoorwaarden.

Methode	Iteraties	Genormaliseerde Fout
Jacobi	14	7.1096e-07
SRJ	14	7.6435e-07



(a) De relatieve fout per iteratie van de te benaderende oplossing met een gridgrootte $N = 512$ en een tijdstap $\Delta t = 0,125$ met Dirichlet randvoorwaarden.

Methode	Iteraties	Genormaliseerde Fout
Jacobi	16	9.3927e-07
SRJ	16	8.5594e-07



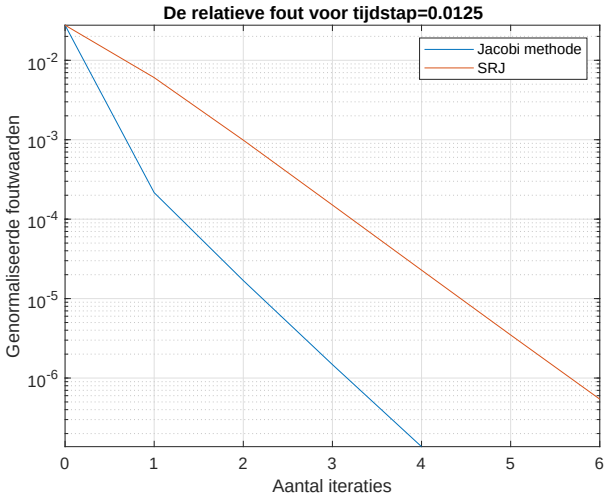
(b) De relatieve fout per iteratie van de te benaderende oplossing met een gridgrootte $N = 512$ en een tijdstap $\Delta t = 0,15$ met Dirichlet randvoorwaarden.

Methode	Iteraties	Genormaliseerde Fout
Jacobi	19	6.5997e-07
SRJ	20	8.3616e-07

Aan de resultaten is te zien dat de Jacobi methode inderdaad sneller convergeert voor een kleinere tijdstap. Merk hierbij op dat de tijdstap nog steeds erg groot is in verhouding met de plaatsstap voor de gekozen gridgrootte. Voor een tijdstap die in verhouding staat met de plaatsstap behaalt de Jacobi methode het stopcriterium al binnen één iteratie. Voor de tijdstap $\Delta t = 0,125$ is te zien dat de SRJ methode net iets sneller convergeert dan de Jacobi methode. Voor de tijdstap $\Delta t = 0,15$ divergeert de SRJ methode na een aantal iteraties, dit is te verklaren doordat de amplificatiefactor groter wordt dan 1 voor een te grote waarde van $p = \frac{\alpha \Delta t}{h^2}$. Oftewel voor een te grote tijdstap in verhouding met de plaatsstap.

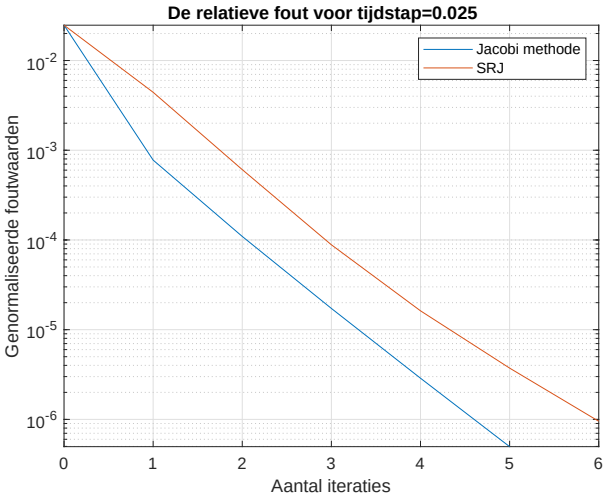
4.1.2 Neumann randvoorwaarden

Vervolgens zullen de iteratieve methoden getest worden voor de eendimensionele warmtevergelijking met Neumann randvoorwaarden $\frac{\partial u(x)}{\partial n} = 30$. De gridgrootte blijft $N = 512$ en de initiële benadering blijft de nulvector. Er zal worden gekeken naar dezelfde tijdstapgrootte.



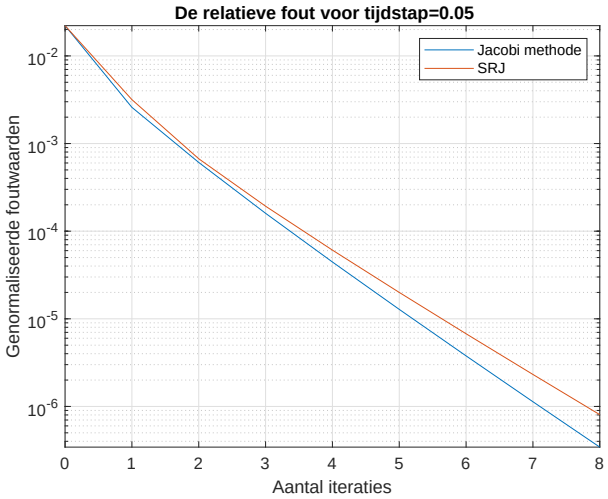
(a) De relatieve fout per iteratie van de te benaderende oplossing met een gridgrootte $N = 512$ en een tijdstap $\Delta t = 0,0125$ met Neumann randvoorwaarden.

Methode	Iteraties	Genormaliseerde Fout
Jacobi	5	1.3667e-07
SRJ	8	1.7322e-07



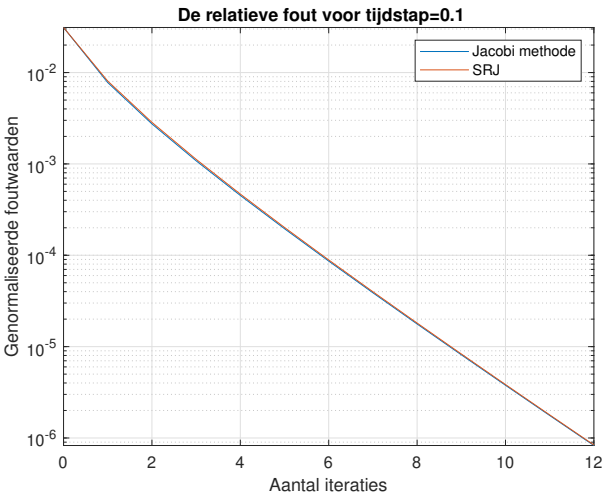
(b) De relatieve fout per iteratie van de te benaderende oplossing met een gridgrootte $N = 512$ en een tijdstap $\Delta t = 0,025$ met Neumann randvoorwaarden.

Methode	Iteraties	Genormaliseerde Fout
Jacobi	6	4.9614e-07
SRJ	7	9.9202e-07



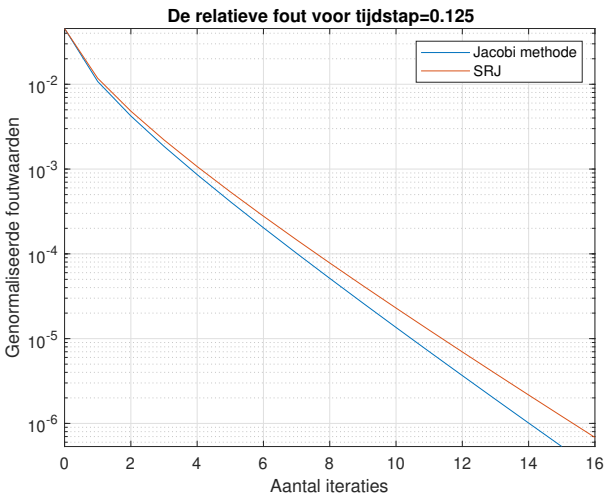
(a) De relatieve fout per iteratie van de te benaderende oplossing met een gridgrootte $N = 512$ en een tijdstap $\Delta t = 0,05$ met Neumann randvoorwaarden.

Methode	Iteraties	Genormaliseerde Fout
Jacobi	9	3.4295e-07
SRJ	9	8.0482e-07

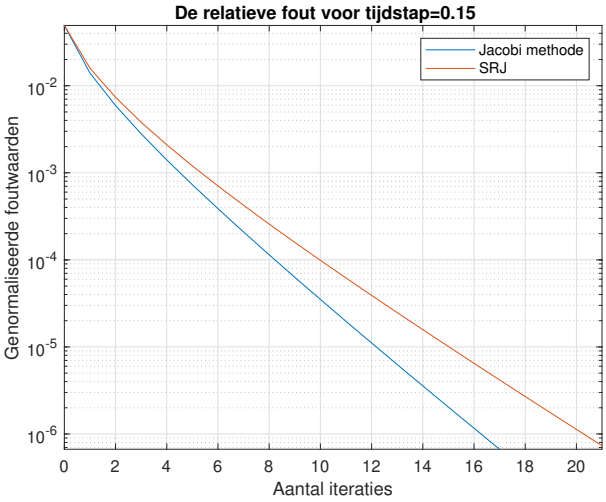


(b) De relatieve fout per iteratie van de te benaderende oplossing met een gridgrootte $N = 512$ en een tijdstap $\Delta t = 0,1$ met Neumann randvoorwaarden.

Methode	Iteraties	Genormaliseerde Fout
Jacobi	13	8.2605e-07
SRJ	13	8.3716e-07



(a) De relatieve fout per iteratie van de te benaderende oplossing met een gridgrootte $N = 512$ en een tijdstap $\Delta t = 0,125$ met Neumann randvoorwaarden.



(b) De relatieve fout per iteratie van de te benaderende oplossing met een gridgrootte $N = 512$ en een tijdstap $\Delta t = 0,15$ met Neumann randvoorwaarden.

Methode	Iteraties	Genormaliseerde Fout
Jacobi	16	5.3359e-07
SRJ	17	6.8363e-07

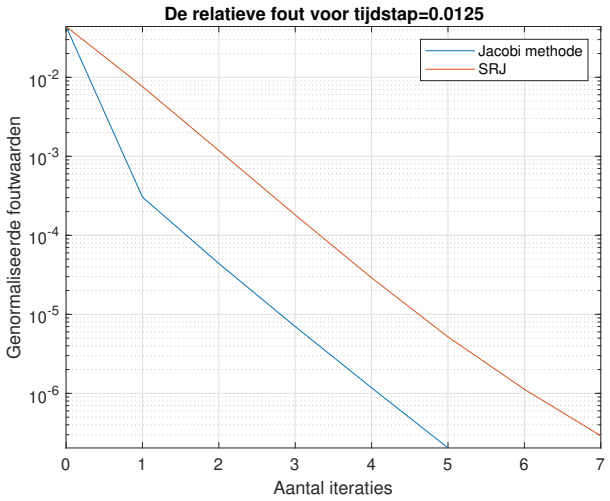
Methode	Iteraties	Genormaliseerde Fout
Jacobi	18	6.6814e-07
SRJ	22	7.3359e-07

De resultaten van de vergeliijking met de Neumann randvoorwaarden zijn vergelijkbaar met de resultaten van de vergelijking met Dirichlet randvoorwaarden, maar bij de tijdstappen $\Delta t = 0,125$ en $\Delta t = 0,15$ is te zien dat de SRJ methode divergeert voor kleinere tijdstappen bij Neumann randvoorwaarden dan bij Dirichlet randvoorwaarden.

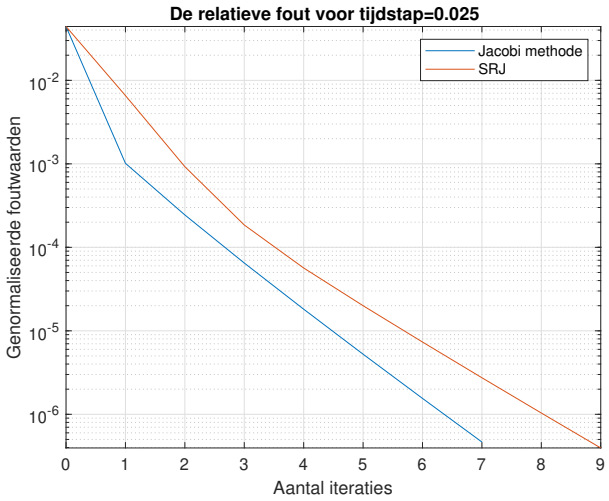
4.2 Resultaten voor de tweedimensionele warmtevergelijking

4.2.1 Dirichlet randvoorwaarden

Naast de eendimensionele warmtevergelijking, zijn ook de genormaliseerde foutwaarden van de tweedimensionele warmtevergelijking gemodellerd. Eerst zal worden gekeken naar de genormaliseerde foutwaarde voor Dirichlet randvoorwaarden $u(0, y, t) = 30$, $u(L, y, t) = 30$, $u(x, 0, t) = 30$ en $u(x, H, t) = 30$ met wederom een gridgrootte van 512. De initiële benadering is gelijk aan de nulvector. De verwachting is dat de Jacobi methode sneller convergeert voor kleine tijdstappen dan voor grotere tijdstappen en dat de Jacobi methode voor kleinere tijdstappen sneller convergeert dan de SRJ methode. Daarnaast zal de Scheduled Relaxation Jacobi methode waarschijnlijk minder in snelheid afnemen bij grotere tijdstappen dan de Jacobi methode waardoor de SRJ methode voor grotere tijdstappen sneller zal convergeren dan de Jacobi methode. voor een grote tijdstap. Naar verwachting zullen de resultaten vergelijkbaar zijn met de eendimensionele warmtevergelijking met Dirichlet randvoorwaarden, hierbij wordt wel een iets langzamere convergentie verwacht vanwege de vergroting van het stelsel.



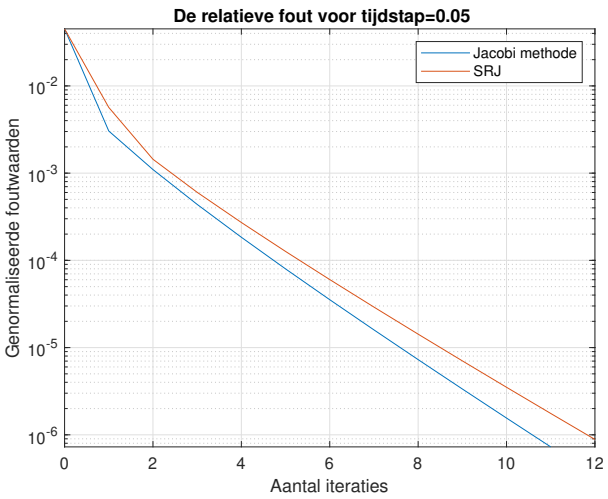
(a) De relatieve fout per iteratie van de te benaderende oplossing met een gridgrootte $N = 512$ en een tijdstap $\Delta t = 0,0125$ met Dirichlet randvoorwaarden.



(b) De relatieve fout per iteratie van de te benaderende oplossing met een gridgrootte $N = 512$ en een tijdstap $\Delta t = 0,025$ met Dirichlet randvoorwaarden.

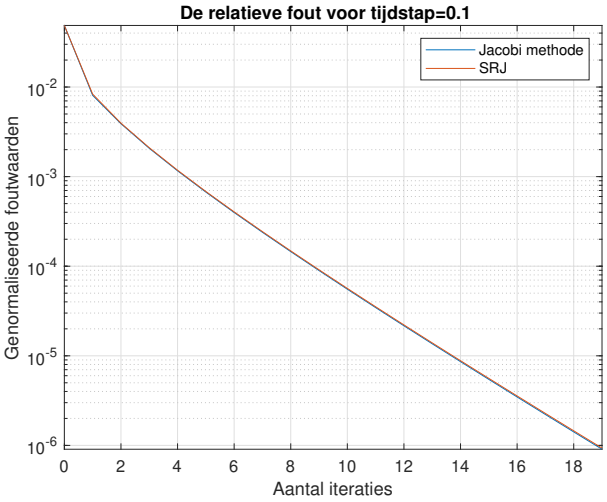
Methode	Iteraties	Genormaliseerde Fout
Jacobi	6	2.0374e-07
SRJ	8	2.9091e-07

Methode	Iteraties	Genormaliseerde Fout
Jacobi	8	4.6575e-07
SRJ	10	3.9539e-07



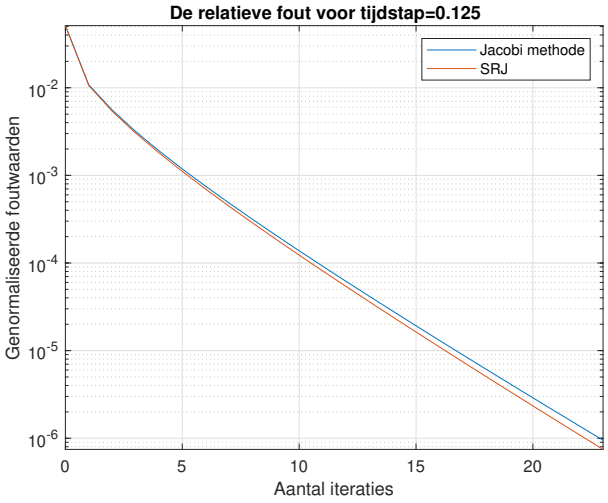
(a) De relatieve fout per iteratie van de te benaderende oplossing met een gridgrootte $N = 512$ en een tijdstap $\Delta t = 0,05$ met Dirichlet randvoorwaarden.

Methode	Iteraties	Genormaliseerde Fout
Jacobi	12	7.2723e-07
SRJ	13	8.8555e-07

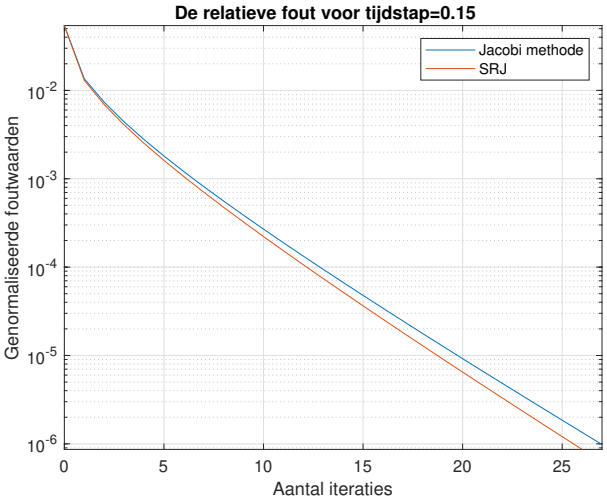


(b) De relatieve fout per iteratie van de te benaderende oplossing met een gridgrootte $N = 512$ en een tijdstap $\Delta t = 0,1$ met Dirichlet randvoorwaarden.

Methode	Iteraties	Genormaliseerde Fout
Jacobi	20	9.0613e-07
SRJ	20	9.3333e-07



(a) De relatieve fout per iteratie van de te benaderende oplossing met een gridgrootte $N = 512$ en een tijdstap $\Delta t = 0,125$ met Dirichlet randvoorwaarden.



(b) De relatieve fout per iteratie van de te benaderende oplossing met een gridgrootte $N = 512$ en een tijdstap $\Delta t = 0,15$ met Dirichlet randvoorwaarden.

Methode	Iteraties	Genormaliseerde Fout
Jacobi	24	9.4979e-07
SRJ	24	7.4604e-07

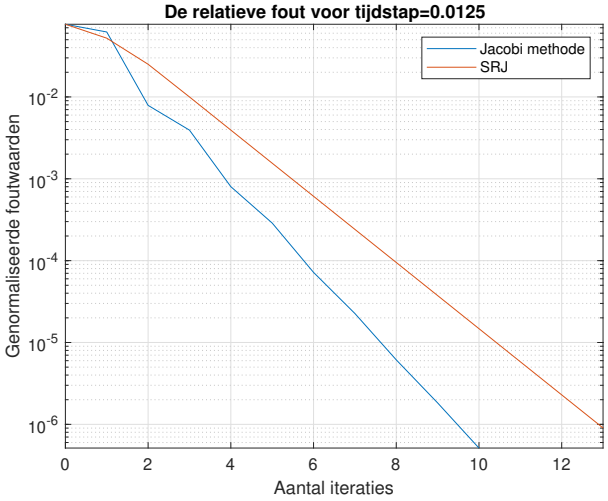
Methode	Iteraties	Genormaliseerde Fout
Jacobi	28	9.8269e-07
SRJ	27	8.6529e-07

De resultaten voldoen aan de verwachtingen. Er is te zien dat de Jacobi methode inderdaad sneller is dan de Scheduled Relaxation Jacobi methode voor kleinere tijdstappen en dat de convergentie van de Jacobi methode meer in snelheid afneemt bij vergroting van de tijdstap dan de SRJ methode waardoor de SRJ methode sneller convergeert dan de Jacobi methode bij een grotere tijdstap. De resultaten zijn erg vergelijkbaar met de resultaten van de eendimensionel warmtevergelijking met Dirichlet randvoorwaarden. De verwachting was dat beiden methoden iets langzamer zouden convergeren aangezien het tweedimensionele stelsel $(N+1)^2 = 513^2$ keer zo groot is. Aan de resultaten is echter te zien dat beiden methoden wel langzamer convergeren bij een

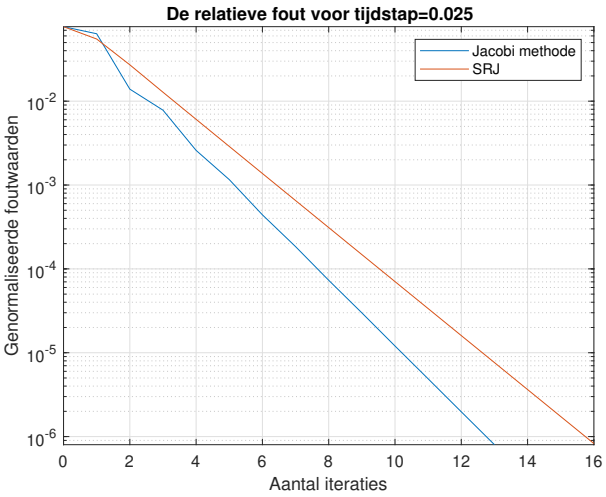
grotere tijdstappen, maar amper langzamer convergeren bij een kleinere tijdstap.

4.2.2 Neumann randvoorwaarden

Het tweedimensionele geval is ook bekeken met Neumann randvoorwaarden $\frac{\partial u(x,y)}{\partial n} = 30$. De verwachting is vergelijkbare resultaten met het eendimensionele geval met langzamere convergentie voor beiden methoden bij grotere tijdstappen. Bij de eendimensionele warmtevergelijking met Neumann randvoorwaarden, is te zien dat de oplossing van de SRJ methode na enkele iteraties steeds langzamer convergeert bij een te grote tijdstap. De verwachting is dat dat in het tweedimensionele geval ook zal gebeuren en dat hierdoor de Jacobi methode zelfs bij een grotere tijdstap sneller zal convergeren dan de SRJ methode.



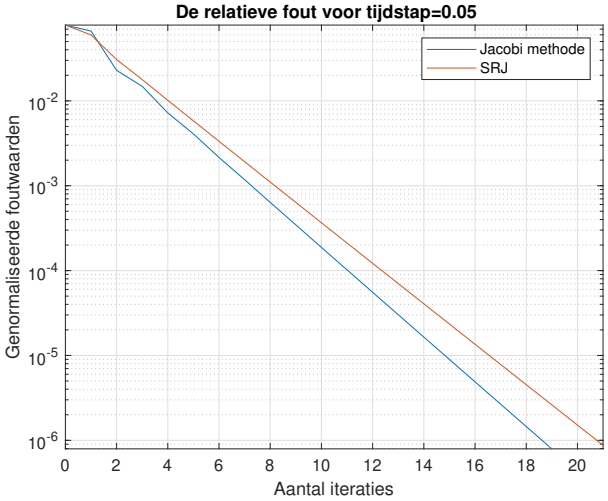
(a) De relatieve fout per iteratie van de te benaderende oplossing met een gridgrootte $N = 512$ en een tijdstap $\Delta t = 0,0125$ met Neumann randvoorwaarden.



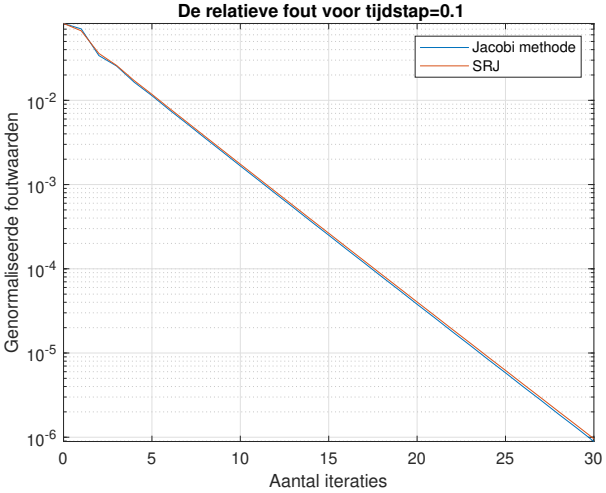
(b) De relatieve fout per iteratie van de te benaderende oplossing met een gridgrootte $N = 512$ en een tijdstap $\Delta t = 0,025$ met Neumann randvoorwaarden.

Methode	Iteraties	Genormaliseerde Fout
Jacobi	11	5.1355e-07
SRJ	14	9.0356e-07

Methode	Iteraties	Genormaliseerde Fout
Jacobi	14	8.0045e-07
SRJ	17	8.2645e-07



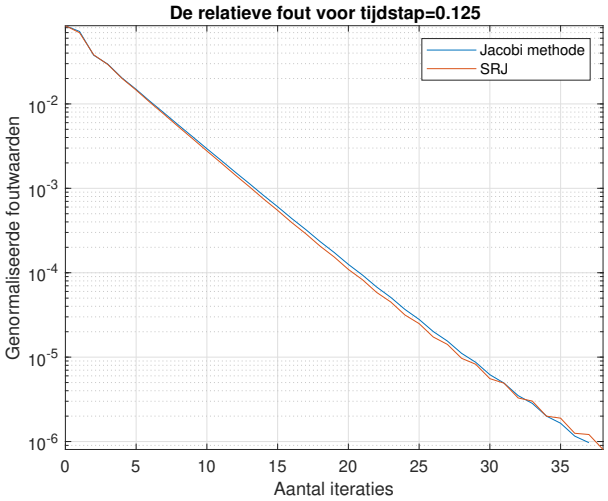
(a) De relatieve fout per iteratie van de te benaderende oplossing met een gridgrootte $N = 512$ en een tijdstap $\Delta t = 0,05$ met Neumann randvoorwaarden.



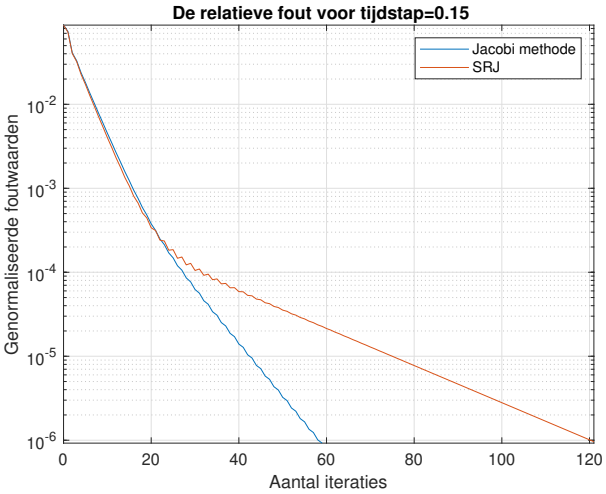
(b) De relatieve fout per iteratie van de te benaderende oplossing met een gridgrootte $N = 512$ en een tijdstap $\Delta t = 0,1$ met Neumann randvoorwaarden.

Methode	Iteraties	Genormaliseerde Fout
Jacobi	20	7.9313e-07
SRJ	22	8.751e-07

Methode	Iteraties	Genormaliseerde Fout
Jacobi	31	8.8955e-07
SRJ	31	9.5867e-07



(a) De relatieve fout per iteratie van de te benaderende oplossing met een gridgrootte $N = 512$ en een tijdstap $\Delta t = 0,125$ met Neumann randvoorwaarden.



(b) De relatieve fout per iteratie van de te benaderende oplossing met een gridgrootte $N = 512$ en een tijdstap $\Delta t = 0,15$ met Neumann randvoorwaarden.

Methode	Iteraties	Genormaliseerde Fout
Jacobi	38	9.6996e-07
SRJ	39	8.0508e-07

Methode	Iteraties	Genormaliseerde Fout
Jacobi	60	9.2344e-07
SRJ	122	9.6129e-07

Aan de bovenstaande resultaten is te zien dat beiden methode iets langzamer convergeren voor de tweedimensionele warmtevergelijking dan voor de eendimensionele warmtevergelijking. Beiden methode tonen echter een vergelijkbare verhouding in convergentiesnelheid. Dit is te verwachten aangezien de stelselmatrix met eenzelfde verhouding vergroot is.

5 Conclusie

De scheduled Relaxation Jacobi methode is een iteratieve methode die bestaat uit meerdere cycli van iteraties van de gerelaxeerde Jacobi methode. Deze methode is ontworpen voor elliptische vergelijking, om de convergentie van de Jacobi methode te versnellen. In dit verslag is deze methode toegepast op een parabolische vergelijking. Deze methode is vergeleken met de standaard Jacobi methode om te onderzoeken of deze methode ook tijdsafhankelijke problemen sneller kan oplossen. Ter illustratie is gekeken naar de warmtevergelijking. Na discretitatie met zowel Dirichlet als Neumann randvoorwaarden, zijn de oplossingen van de iteratieve methode geanalyseerd. Deze analyses zijn vervolgens getest met behulp van MATLAB.
Aan de resultaten is te zien dat de methoden in zowel het eendimensionele als in het tweedimensionale geval en met zowel Dirichlet als Neumann randvoorwaarden, voldoet aan de verwachtingen naar aanleiding van het analyseren van de amplificatiefactoren van de Jacobi methode en de SRJ methode. De SRJ methode is ontworpen om de standaard Jacobi sneller te laten convergeren voor elliptische vergelijkingen. Voor tijdsafhankelijke vergelijkingen convergeert de standaard Jacobi methode een stuk sneller dan voor tijdsonafhankelijke vergelijkingen vanwege sterkere diagonale dominantie in de stelselmatrices. Aan de analyses en de resultaten is te zien dat de standaard Jacobi methode niet tot amper geoptimaliseerd kan worden met behulp van relaxatiegewichten voor tijdsafhankelijke problemen. Bij een zeer grote tijdstap in verhouding tot de plaatsstap kan de SRJ methode wel sneller convergeren dan de Jacobi methode, dit is echter te verklaren doordat de tijdsafhankelijkheid in dat geval nog een zeer kleine rol speelt en de vergelijking dus de vorm krijgt van een elliptische vergelijking.

Referenties

[1] Nguyen van B. (2017). *Een snellere oplosmethode voor lineaire stelsels.* TU Delft.

[2] Recktenwald, G.W. (2011). *Finite-Difference Approximations to the Heat Equation* Portland State University

[3] Vuik C., & Lahaye, D.J.P. (2015). *Scientific Computing.* Delft Institute of Applied Mathematics.

[4] Vuik, C., Vermolen, F.J., Gijzen, van M.B., & Vuik, M.J. (2015). *Numerical Methods for Ordinary Differential Equations.* Delft Academic Press.

[5] Xiyang, I.A., & Mittal, R. (2014). *Acceleration of the Jacobi iterative method by factors exceeding 100 using scheduled relaxation.* Journal of Computational Physics.

Bijlage

MATLAB code voor de eendimensionele warmtevergelijking

```
clear;
close all;

%gridgrote bepalen
N=512;
h=1/N;
Ntijd=8;
delta = 1/(Ntijd);
Trand=30;

%beginwaarden
u0=20*ones((N+1),1);

%de discretitatiematrices definieren
I = ones(N+1,1);
p = 1.9 *delta*10^(-5)/h^2;
I1=diag(I);
A = spdiags([-p*I (1+2*p*I) -p*I],-1:1,N+1, N+1);
Ad1=A;
Ad1(1,1)=1; Ad1(N+1,N+1)=1;
Ad1(1,2)=0;Ad1(N+1,N)=0;
An1=Ad1;
An1(1,1)=p;An1(N+1,N+1)=p;
An1(2,1)=0;An1(N,N+1)=0;
%Ad1=[[5,2,0,1,0];[2,3,0,0,0];[0,1,6,0,3];[0,1,2,4,0];[1,1,1,1,5]] %test

u=u0;

%% de fout voor de dirichlet randvoorwaarden
% f=zeros((N+1),1);
% f(2)=p*Trand+f(2);
% f(N)=p*Trand+f(N);
%
% Mjac=diag(diag(Ad1));
% Njac=Mjac-Ad1;
% Bjac=I1-Mjac\Ad1;
```

```
% s=u+f; %=u^(m+1)+f^m=Au
% s(1)=Trand;
% s(N+1)=Trand;
%
%
% x=Ad1\s;
% i=20; %aantal iteraties
%
% Error1=[];
% Residu1=[];
% while 1
%       e=x-u;
%       error=norm(e);
%       r=s-Ad1*u;
%       residu=norm(r);
%       u1=Mjac\(Njac*u+s);
%       u=u1;
%       Error1=[Error1,error];
%       Residu1=[Residu1,residu];
%       if (residu/norm(s))<10^(-6)
%           break
%       end
%       Residu1=Residu1;
% end
%
% % w=1.033;
% % w=(1+2*p)/(1+4*p);
% % b=(2/(1+2*p)-6*(2+(1/(1+2*p)))^2)/((1/(1+2*p))+(2+(1/(1+2*p)))^2);
% % a=-b-sqrt(b^2-4);
% % w2=(a+1)/(2+(1/(1+2*p)));
% % w1=w2/a;
% % q1=1; q2=1;
% % w1=1972;
% % w2=0.99267;
% % q1=1; q2=564;
% % M=q1+q2;
% w1=59226;w2= 3900.56;w3= 187.53;w4= 9.1194;w5= 0.73905; q1=1;q2= 6;
% q3=40; q4=277; q5=1500; M=q1+q2+q3+q4+q5;
%
% Error2=[];
% Residu2=[];
% U=u0;
% while 1
%       e2=x-U;
%       error2=norm(e2);
%       r2=s-Ad1*U;
%       residu2=norm(r2);
% %       u2=U+w*(Mjac\r2);
% %       U=u2;
% %       u3=(U+w1*(Mjac\r2)).^(q1/M).*(U+w2*(Mjac\r2)).^(q2/M);
% %       U=u3;
%       u4=(U+w1*(Mjac\r2)).^(q1/M).*(U+w2*(Mjac\r2)).^(q2/M).*(U+w3*(Mjac\r2)).^(q3/M).*(U+w
%       U=u4;
%       Error2=[Error2,error2];
```

```
%       Residu2=[Residu2,residu2];
%       if (residu2/norm(s))<10^(-6)
%           break
%       end
%       Residu2=Residu2;
% end
%
% L1=length(Residu1);
% L2=length(Residu2);
%
% Iteraties=[L1;L2];
% GenormaliseerdeFout = [residu/norm(s);residu2/norm(s)];
% Methode = {'Jacobi'; 'SRJ'};
% T=table(Iteraties,GenormaliseerdeFout,'RowNames', Methode)
% TT=["Methode","Iteraties","GenormaliseerdeFout";"Jacobi",L1,residu/norm(s);"SRJ",L2,resid
% latex_table=latex(sym(TT));
%% writetable(T,'Ddtable.txt');


% Ddirichlet6=figure;
% y1=0:1:L1-1;
% y2=0:1:L2-1;
% semilogy(y1,Residu1/norm(s))
% axis tight
% hold on
% semilogy(y2,Residu2/norm(s))
% grid on
% hold off
% title(['De relatieve fout voor tijdstap=', num2str(delta)])
% xlabel('Aantal iteraties')
% ylabel('Genormaliseerde foutwaarden')
% legend({'Jacobi methode', 'SRJ'})
% print Ddirichlet6 -depsc



% de fout voor de neumann randvoorwaarden
f=zeros((N+1),1);
f(1)=p*h*Trand+(1/2)*f(1);
f(N+1)=p*h*(-30)+(1/2)*f(N+1);

Mjac=diag(diag(An1));
Njac=Mjac-An1;
 %=u^(m+1)+f^m=Au


i=50; %aantal iteraties
s=u+f;
x=An1\s;
residu=norm(s);

Error1=[];
Residu1=[];
while (residu/norm(s))>10^(-6)
```

```
    e=x-u;
    error=norm(e);
    r=s-An1*u;
    residu=norm(r);
    u1=Mjac\(Njac*u+s);
    u=u1;
    Error1=[Error1,error];
    Residu1=[Residu1,residu];
end

w=1.033;
w=(1+2*p)/(1+4*p);
b=(2/(1+2*p)-6*(2+(1/(1+2*p)))^2)/((1/(1+2*p))+(2+(1/(1+2*p)))^2);
a=-b-sqrt(b^2-4);
w2=(a+1)/(2+(1/(1+2*p)));
w1=w2/a;
q1=1; q2=1;
% w1=1972;
% w2=0.99267;
% q1=1; q2=564;
M=q1+q2;

Error2=[];
Residu2=[];
U=u0;
while 1
    e2=x-U;
    error2=norm(e2);
    r2=s-An1*U;
    residu2=norm(r2);
%     u2=U+w*(Mjac\r2);
%     U=u2;
    u3=(U+w1*(Mjac\r2)).^(q1/M).*(U+w2*(Mjac\r2)).^(q2/M);
    U=u3;
    Error2=[Error2,error2];
    Residu2=[Residu2,residu2];
    if (residu2/norm(s))<10^(-6)
        break
    end
    Residu2=Residu2;
end

L1=length(Residu1);
L2=length(Residu2);

% Iteraties=[L1;L2];
% GenormaliseerdeFout = [residu/norm(s);residu2/norm(s)];
% Methode = {'Jacobi'; 'SRJ'};
% T=table(Iteraties,GenormaliseerdeFout,'RowNames', Methode)
% TT=["Methode","Iteraties","GenormaliseerdeFout";"Jacobi",L1,residu/norm(s);"SRJ",L2,resid


Dneuman10=figure;
y1=0:1:L1-1;
```

```
y2=0:1:L2-1;
semilogy(y1,Residu1/norm(s))
axis tight
hold on
semilogy(y2,Residu2/norm(s))
grid on
hold off
title(['De relatieve fout voor tijdstap=', num2str(delta)])
xlabel('Aantal iteraties')
ylabel('Genormaliseerde foutwaarden')
legend({'Jacobi methode', 'SRJ'})
print Dneuman10 -depsc
```

MATLAB code voor de tweedimensionele warmtevergelijking

```
clear;
close all;


N=512;
Ntijd=20;
delta = 3/Ntijd;
h=1/N;
p = 1.9 *10^(-5)*delta/h^2;
Trand = 30;

u0=20*ones((N+1)^2,1);

I = ones(N+1,1);
I1 = diag([0 ones([1 N-1]) 0]);
T = spdiags([-p*I 0.5+2*p*I -p*I],-1:1,N+1, N+1);
T1 = T;
T1(1,1)=1;T1(1,2)=0;T1(2,1)=0;T1(end,end-1)=0;T1(end-1,end)=0;T1(end,end)=1;
T1_full=full(T1);

Ad2 = kron(T1,I1)+kron(I1,T1);
An2 = kron(T1,I1)+kron(I1,T1);
%j= (N+1)^2-N;
%A(1,1)=h^2;A(N+1,N+1)=h^2;A((N+1)^2,(N+1)^2)=h^2;A(j,j)=h^2;
for i=1:N+1
    Ad2(i,i)=1;
    Ad2((N+1)^2-i+1,(N+1)^2-i+1)=1;
end
for i=1:N+1
    An2(i,i)=p;
    An2(i,N+1+i)=-p;
    An2(N+1+i,i)=-p;
    An2((N+1)^2-i+1,(N+1)^2-i+1)=p;
    An2((N+1)^2-N-i,(N+1)^2-i+1)=-p;
    An2((N+1)^2-i+1,(N+1)^2-N-i)=-p;
end

% %dirichlet
% d=diag(Ad2);
```

```
% Mjac=diag(d);
% Njac=Mjac-Ad2;
%
% f=zeros((N+1)^2,1);
% for i=0:N
%     f(i*(N+1)+2)=p*Trand+f(i*(N+1)+2);
%     f(i*(N+1)+N)=p*Trand+f(i*(N+1)+N);
% end
% for j=1:N+1
%     f((N+1)+j)=p*Trand+f(2*(N+1)+j);
%     f((N-1)*(N+1)+j)=p*Trand+f(N*(N+1)+j);
% end
%
% u=u0;
%
% s=u+f; %=u^(m+1)+f^m=Au
% for i=1:N+1
%     s(i)=Trand;
%     s(N*(N+1)+i)=Trand;
%     s((i-1)*(N+1)+1)=Trand;
%     s((i-1)*(N+1)+(N+1))=Trand;
% end
%
% x=Ad2\s;
%  %aantal iteraties
%
% Error1=[];
% Residu1=[];
% while 1
%     e=x-u;
%     error=norm(e);
%     r=s-Ad2*u;
%     residu=norm(r);
%     u1=Mjac\(Njac*u+s);
%     u=u1;
%     Error1=[Error1,error];
%     Residu1=[Residu1,residu];
%     if (residu/norm(s))<10^(-6)
%         break
%     end
%     Residu1=Residu1;
% end
%
% w=1.05;
% b=(2/(1+2*p)-6*(2+(1/(1+2*p)))^2)/((1/(1+2*p))+(2+(1/(1+2*p)))^2);
% a=-b-sqrt(b^2-4);
% w2=(a+1)/(2+(1/(1+2*p)));
% w1=w2/a;
% q1=1;q2=1;
% % w1=1972;
% % w2=0.99267;
% % q1=1; q2=564;
% M=q1+q2;
% Error2=[];
```

```
% Residu2=[];
% U=u0;
% while 1
%       e2=x-U;
%       error2=norm(e2);
%       r2=s-Ad2*U;
%       residu2=norm(r2);
% %       u2=U+w*(Mjac\r2);
% %       U=u2;
%       u3=(U+w1*(Mjac\r2)).^(q1/M).*(U+w2*(Mjac\r2)).^(q2/M);
%       U=u3;
%       Error2=[Error2,error2];
%       Residu2=[Residu2,residu2];
%       if (residu2/norm(s))<10^(-6)
%           break
%       end
%       Residu2=Residu2;
% end
%
% L1=length(Residu1);
% L2=length(Residu2);
%
% % Iteraties=[L1;L2];
% % GenormaliseerdeFout = [residu/norm(s);residu2/norm(s)];
% % Methode = {'Jacobi'; 'SRJ'};
% % T=table(Iteraties,GenormaliseerdeFout,'RowNames', Methode)
% % TT=["Methode","Iteraties","GenormaliseerdeFout";"Jacobi",L1,residu/norm(s);"SRJ",L2,res
% %
%
% DDdirichlet=figure;
% y1=0:1:L1-1;
% y2=0:1:L2-1;
% semilogy(y1,Residu1/norm(s))
% axis tight
% hold on
% semilogy(y2,Residu2/norm(s))
% grid on
% hold off
% title(['De relatieve fout voor tijdstap=', num2str(delta)])
% xlabel('Aantal iteraties')
% ylabel('Genormaliseerde foutwaarden')
% legend({'Jacobi methode', 'SRJ'})
% print DDdirichlet -depsc

%neumann
d=diag(An2);
Mjac=diag(d);
Njac=Mjac-An2;

f=ones((N+1)^2,1);
for i=1:N-1
    f(i+1)=p*h*Trand+(1/2)*f(i+1);
    f(i+N*(N+1)+1)=p*h*Trand+(1/2)*f(i+(N+1)*N+1);
end
```

```
for  j=0:N-2
     f(j*(N+1)+N+2)=p*h*Trand+(1/2)*f(j*(N+1)+N+2);
     f(j*(N+1)+2*(N+1))=p*h*Trand+(1/2)*f(j*(N+1)+2*(N+1));
end
f(1)=(1/4)*f(1)+h*p*Trand;
f(N+1)=(1/4)*f(N+1)+h*p*Trand;
f((N+1)^2-N)=(1/4)*f((N+1)^2-N)+h*p*Trand;
f((N+1)^2)=(1/4)*f((N+1)^2)+h*p*Trand;

u=u0;
s=u+f; %=u^(m+1)+f^m=Au

x=An2\s;
 %aantal iteraties

Error1=[];
Residu1=[];
while 1
     e=x-u;
     error=norm(e);
     r=s-An2*u;
     residu=norm(r);
     u1=Mjac\(Njac*u+s);
     u=u1;
     Error1=[Error1,error];
     Residu1=[Residu1,residu];
     if (residu/norm(s))<10^(-6)
          break
     end
     Residu1=Residu1;
end

w=1.05;
b=(2/(1+2*p)-6*(2+(1/(1+2*p)))^2)/((1/(1+2*p))+(2+(1/(1+2*p)))^2);
a=-b-sqrt(b^2-4);
w2=(a+1)/(2+(1/(1+2*p)));
w1=w2/a;
q1=1;q2=1;
% w1=1972;
% w2=0.99267;
% q1=1; q2=564;
M=q1+q2;
Error2=[];
Residu2=[];
U=u0;
while 1
     e2=x-U;
     error2=norm(e2);
     r2=s-An2*U;
     residu2=norm(r2);
%       u2=U+w*(Mjac\r2);
%       U=u2;
     u3=(U+w1*(Mjac\r2)).^(q1/M).*(U+w2*(Mjac\r2)).^(q2/M);
     U=u3;
```

```
    Error2=[Error2,error2];
    Residu2=[Residu2,residu2];
    if (residu2/norm(s))<10^(-6)
        break
    end
    Residu2=Residu2;
end

L1=length(Residu1);
L2=length(Residu2);

Iteraties=[L1;L2];
GenormaliseerdeFout = [residu/norm(s);residu2/norm(s)];
Methode = {'Jacobi'; 'SRJ'};
T=table(Iteraties,GenormaliseerdeFout,'RowNames', Methode)
TT=["Methode","Iteraties","GenormaliseerdeFout";"Jacobi",L1,residu/norm(s);"SRJ",L2,residu2


% DDneuman6=figure;
% y1=0:1:L1-1;
% y2=0:1:L2-1;
% semilogy(y1,Residu1/norm(s))
% axis tight
% hold on
% semilogy(y2,Residu2/norm(s))
% grid on
% hold off
% title(['De relatieve fout voor tijdstap=', num2str(delta)])
% xlabel('Aantal iteraties')
% ylabel('Genormaliseerde foutwaarden')
% legend({'Jacobi methode', 'SRJ'})
% print DDneuman6 -depsc
```