



Namen: Sebastiaan, 9121164
Henri van den Esker, 1015990
Vak: Scientific computing and simulation
Vakcode: WI4 130
Datum: februari 2003
Opgave: 1c; C.W. Oosterlee
Email: sebastia@ch.twi.tudelft.nl
henri_vd_e@hotmail.com

Inhoudsopgave

| | |
|---|----|
| Inhoudsopgave | 3 |
| Inleiding | 5 |
| 1 Probleemstelling | 7 |
| 2 Discretisatie | 9 |
| 2.1 Keuze grid | 9 |
| 2.2 Discretisatie met behulp van EVM | 9 |
| 2.3 Nauwkeurigheid | 11 |
| 2.4 Voorbeeld | 12 |
| 3 Oplosmethode | 15 |
| 3.1 De A is een M-matrix | 15 |
| 3.2 gedempte Jacobi iteratie | 15 |
| 3.3 Convergentie factor van Jacobi | 16 |
| 3.4 Multigrid algoritmen | 18 |
| 3.4.1 Algoritme van de 2-grid methode | 19 |
| 3.4.2 Algoritme van de 3-grid methode | 19 |
| 3.4.3 Algoritme van de M-grid methode | 20 |
| 3.5 De matrices nader bekeken | 21 |
| 3.5.1 Van fijn naar grof | 21 |
| 3.5.2 Van grof naar fijn | 22 |
| 4 Resultaten | 25 |
| Bijlage A: Resultaten | 27 |
| A.1 Gedempte Jacobi | 27 |
| A.2 2-grid methode | 27 |
| A.3 3-grid methode | 27 |
| A.4 Smoothing | 28 |
| A.5 M-grid methode | 28 |
| Bijlage B: Listings | 29 |
| B.1 Generatie | 29 |
| genA.m | 29 |
| gen_u.m | 29 |
| genRL_f.m | 29 |
| genRL_u.m | 30 |
| genId_fijn2grof.m | 30 |
| genId_grof2fijn.m | 30 |
| B.2 Calculatie | 31 |
| A_row_product.m | 31 |
| A_product.m | 31 |
| I01_product | 32 |
| I10_product | 32 |
| B.3 Methoden | 33 |
| djacobi_stap.m | 33 |
| djacobi.m | 33 |
| a_Mgrid_cycle.m | 34 |
| a_Mgrid.m | 34 |
| Bijlage C: Opgave | 37 |

Inleiding

Voor het vak Scientific Computing and Simulation (WI4 130) hebben we een practicum opdracht moeten maken. In dit vak wordt gekeken naar het numeriek oplossen van differentiaalvergelijkingen, in ons geval de Helmholtz vergelijking, voor grootschalige problemen.

We hebben in dit practicum de Helmholtz vergelijking numeriek opgelost door middel van de gedempte Jacobi methode en daar op aansluitend de multigrid methode. De gedempte Jacobi methode heeft een trage convergentie wanneer het aantal roosterpunten stijgt. Om deze methode te blijven gebruiken voor fijne grids is dus niet aantrekkelijk.

Hiervoor wijken we uit naar een multigrid algoritme. Deze gebruikt ook de gedempte Jacobi, voor onder andere smoothing en het bepalen van de inverse. De inverse wordt bepaald op een grover grid dat het oorspronkelijke grid representeert. Aangezien dit grid een stuk grover is, is Jacobi weer aantrekkelijk.

We hebben ondervonden dat op fijne grids tot $N=8$ de gedempte Jacobi methode goed te gebruiken is. Voor N tussen 8 en 64 moet je uitwijken naar een 2-grid. Voor nog grotere problemen is een 3-grid of verder aan te raden. Voor ons probleem is gebleken dat de 3-grid methode voor $N=128$ meer dan voldoende is.

1 Probleemstelling

We bekijken de Helmholtz vergelijking met Dirichlet randvoorwaarden:

$$\begin{aligned} -\operatorname{div}\nabla u + 200u &= f & \text{op } \Omega = (0,1)\times(0,1) \\ u &= g & \text{op } \partial\Omega \end{aligned} \tag{1.1}$$

met $f = f(x, y)$, $u = u(x, y)$, $f(x, y) = -6(x^2 y(x^2 + 2y^2)) + 200x^4 y^3$ en $g(x, y) = x^4 y^3$

Het oplossen van deze differentiaalvergelijking geschied eerst direct met behulp van gedempte Jacobi, hetgeen een variant is op Jacobi. Daarna wordt gekeken naar multigrid methoden. We willen deze methoden vergelijken qua convergentie, het aantal benodigde stappen en de schaalbaarheid naar grote problemen.

2 Discretisatie

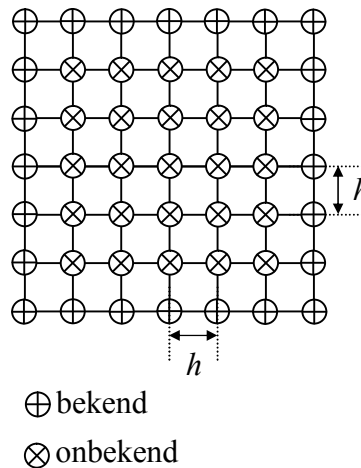
Voordat we met het oplossen van het probleem kunnen beginnen, moeten we eerst een geschikte discretisatie maken. Dit doen we door notaties in te voeren en een grid af spreken. De discretisatie geschied met behulp van de Eindige Volume Methode. Doel is om een discretisatie te krijgen van $O(h^2)$. We sluiten af met een voorbeeld.

2.1 Keuze grid

We discretiseren

$$\begin{aligned} -\operatorname{div}\nabla u + 200u &= f & \text{op } \Omega = (0,1)\times(0,1) \\ u &= g & \text{op } \partial\Omega \end{aligned} \quad (1.1)$$

op een equidistant vierkant grid. Als parameter voor het grid wordt N gekozen.



Gekozen is voor een rechthoekig rooster met equidistante afstanden:

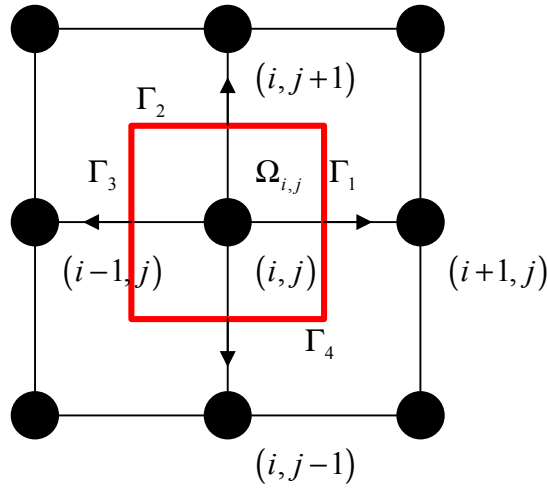
$$G = \{(x_i, x_j) \mid x_i = ih, y_i = jh, h = 1/N\}$$

2.2 Discretisatie met behulp van EVM

introduceer $u_{i,j} = u(ih, jh)$

De onbekenden $u_{i,j}$ worden gekozen op de cell vertices. We discretiseren met behulp van de Eindige Volume Methode.

Gebruik het volgende figuur (een element) als hulpmiddel:



| | |
|----------------|--|
| $\Omega_{i,j}$ | het volume rond de onbekende $u_{i,j}$ |
| Γ_1 | rechterrاند van $\Omega_{i,j}$ |
| Γ_2 | bovenrand van $\Omega_{i,j}$ |
| Γ_3 | linkerrاند van $\Omega_{i,j}$ |
| Γ_4 | onderrاند van $\Omega_{i,j}$ |

Herinner:

$$-\operatorname{div}\nabla u + 200u = f \quad (2.1)$$

Integreer over $\Omega_{i,j}$:

$$\int_{\Omega_{i,j}} -\operatorname{div}\nabla u + 200ud\Omega = \int_{\Omega_{i,j}} fd\Omega \quad (2.2)$$

Pas Gauss op het linkerlid van (2.2) toe:

$$\begin{aligned} \int_{\Omega_{i,j}} -\operatorname{div}\nabla u + 200ud\Omega &= \int_{\Gamma} -\nabla u \cdot dn + 200 \int_{\Omega_{i,j}} ud\Omega \\ &= -\int_{\Gamma} \frac{\partial u}{\partial n} d\Gamma + 200 \int_{\Omega_{i,j}} ud\Omega \\ \int_{\Omega_{i,j}} -\operatorname{div}\nabla u + 200ud\Omega &= -\sum_{i=1}^4 \int_{\Gamma_i} \frac{\partial u}{\partial n} d\Gamma + 200 \int_{\Omega_{i,j}} ud\Omega \end{aligned}$$

Bepaling van de randintegralen:

$$\begin{aligned} \int_{\Gamma_1} \frac{\partial u}{\partial n} d\Gamma &\approx h \left. \frac{\partial u}{\partial x} \right|_{i+\frac{1}{2},j} \approx u_{i+1,j} - u_{i,j} \\ \int_{\Gamma_2} \frac{\partial u}{\partial n} d\Gamma &\approx h \left. \frac{\partial u}{\partial y} \right|_{i,j+\frac{1}{2}} \approx u_{i,j+1} - u_{i,j} \\ \int_{\Gamma_3} \frac{\partial u}{\partial n} d\Gamma &\approx -h \left. \frac{\partial u}{\partial x} \right|_{i-\frac{1}{2},j} \approx u_{i-1,j} - u_{i,j} \\ \int_{\Gamma_4} \frac{\partial u}{\partial n} d\Gamma &\approx -h \left. \frac{\partial u}{\partial y} \right|_{i,j-\frac{1}{2}} \approx u_{i,j-1} - u_{i,j} \end{aligned} \quad (2.3)$$

Vul de termen van (2.3) in $\int_{\Omega_{i,j}} -\operatorname{div}\nabla ud\Omega$:

$$\int_{\Omega_{i,j}} -\text{div}\nabla u d\Omega \approx -u_{i,j+1} - u_{i+1,j} + 4u_{i,j} - u_{i-1,j} - u_{i,j-1} \quad (2.4)$$

De rest van de integralen benaderen we op een soortgelijke manier:

$$200 \int_{\Omega_{i,j}} u d\Omega \approx 200 |\Omega_{i,j}| u_{i,j} = 200h^2 u_{i,j} \quad (2.5)$$

$$\int_{\Omega_{i,j}} f d\Omega \approx |\Omega_{i,j}| f_{i,j} = h^2 f_{i,j} \quad (2.6)$$

Als we (2.4), (2.5) en (2.6) combineren met $\int_{\Omega_{i,j}} -\text{div}\nabla u + 200u d\Omega = \int_{\Omega_{i,j}} f d\Omega$ samenvoegen dan verkrijgen wij:

$$\frac{-u_{i,j+1} - u_{i+1,j} + (4 + 200h^2)u_{i,j} - u_{i-1,j} - u_{i,j-1}}{h^2} = f_{i,j} \quad (2.7)$$

Dit kan men ook noteren in stencil notatie:

$$A_h^{\text{intern}} = \begin{pmatrix} & -1 & \\ -1 & 4 + 200h^2 & -1 \\ & -1 & \end{pmatrix}_h$$

$$A_h^{\text{rand}} = \begin{pmatrix} & -1 & \\ -1 & 4 + 200h^2 & \\ & -1 & \end{pmatrix}_h$$

$$A_h^{\text{hoek}} = \begin{pmatrix} 4 + 200h^2 & -1 \\ -1 & \end{pmatrix}_h$$

2.3 Nauwkeurigheid

De volgende stap is het aantonen dat de discretisatie van $O(h^2)$ is. We bekijken de vergelijking $Au = f$. Noem \tilde{u} de werkelijke oplossing van de vergelijking $-\text{div}\nabla u + 200u = f$ (2.1) onder de gegeven randvoorwaarden. Dit gaan we per molecuul bekijken. We vullen de echte oplossing in vergelijking (2.7) in:

$$\frac{-\tilde{u}_{i,j+1} - \tilde{u}_{i+1,j} + (4 + 200h^2)\tilde{u}_{i,j} - \tilde{u}_{i-1,j} - \tilde{u}_{i,j-1}}{h^2} = f_{i,j}$$

We willen het volgende aantonen:

$$-\text{div}\nabla \tilde{u}_{i,j} - \frac{-\tilde{u}_{i,j+1} - \tilde{u}_{i+1,j} + (4 + 200h^2)\tilde{u}_{i,j} - \tilde{u}_{i-1,j} - \tilde{u}_{i,j-1}}{h^2} = O(h^2) \quad (2.8)$$

Dit betekent dat de lokale fout van orde h^2 is.

Pas Taylor toe op enkele elementen van (2.8):

$$\begin{aligned}
-\tilde{u}_{i,j+1} &= -\tilde{u}(ih, jh) - h \frac{\partial \tilde{u}}{\partial y}(ih, jh) - \frac{h^2}{2!} \frac{\partial^2 \tilde{u}}{\partial y^2}(ih, jh) - \frac{h^3}{3!} \frac{\partial^3 \tilde{u}}{\partial y^3}(ih, jh) + O(h^4) \\
-\tilde{u}_{i+1,j} &= -\tilde{u}(ih, jh) - h \frac{\partial \tilde{u}}{\partial x}(ih, jh) - \frac{h^2}{2!} \frac{\partial^2 \tilde{u}}{\partial x^2}(ih, jh) - \frac{h^3}{3!} \frac{\partial^3 \tilde{u}}{\partial x^3}(ih, jh) + O(h^4) \\
4\tilde{u}_{i,j} &= 4\tilde{u}(ih, jh) \\
-\tilde{u}_{i-1,j} &= -\tilde{u}(ih, jh) + h \frac{\partial \tilde{u}}{\partial x}(ih, jh) - \frac{h^2}{2!} \frac{\partial^2 \tilde{u}}{\partial x^2}(ih, jh) + \frac{h^3}{3!} \frac{\partial^3 \tilde{u}}{\partial x^3}(ih, jh) + O(h^4) \\
-\tilde{u}_{i,j-1} &= -\tilde{u}(ih, jh) + h \frac{\partial \tilde{u}}{\partial y}(ih, jh) - \frac{h^2}{2!} \frac{\partial^2 \tilde{u}}{\partial y^2}(ih, jh) + \frac{h^3}{3!} \frac{\partial^3 \tilde{u}}{\partial y^3}(ih, jh) + O(h^4)
\end{aligned}$$

Optellen links en rechts geeft:

$$-\tilde{u}_{i,j+1} - \tilde{u}_{i+1,j} + 4\tilde{u}_{i,j} - \tilde{u}_{i-1,j} - \tilde{u}_{i,j-1} = -h^2 \frac{\partial^2 \tilde{u}}{\partial x^2}(ih, jh) - h^2 \frac{\partial^2 \tilde{u}}{\partial y^2}(ih, jh) + O(h^4)$$

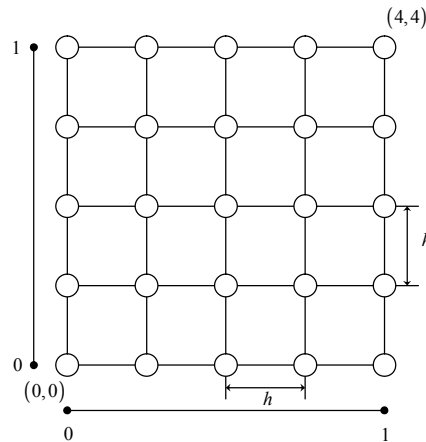
Vul dit in (2.8):

$$\begin{aligned}
&-\frac{\partial^2 \tilde{u}}{\partial x^2}(ih, jh) - \frac{\partial^2 \tilde{u}}{\partial y^2}(ih, jh) - \frac{-h^2 \frac{\partial^2 \tilde{u}}{\partial x^2}(ih, jh) - h^2 \frac{\partial^2 \tilde{u}}{\partial y^2}(ih, jh) + O(h^4)}{h^2} = \\
&-\frac{\partial^2 \tilde{u}}{\partial x^2}(ih, jh) - \frac{\partial^2 \tilde{u}}{\partial y^2}(ih, jh) + \left\{ \frac{\partial^2 \tilde{u}}{\partial x^2}(ih, jh) + \frac{\partial^2 \tilde{u}}{\partial y^2}(ih, jh) + O(h^2) \right\} = O(h^2)
\end{aligned}$$

$$\text{Dus } -\text{div} \nabla \tilde{u}_{i,j} - \frac{-\tilde{u}_{i,j+1} - \tilde{u}_{i+1,j} + (4 + 200h^2)\tilde{u}_{i,j} - \tilde{u}_{i-1,j} - \tilde{u}_{i,j-1}}{h^2} = O(h^2)$$

2.4 Voorbeeld

We geven een voorbeeld voor het geval $N = 4$.



introduceer $f_{i,j} = f(ih, jh)$
 $g_{i,j} = g(ih, jh)$

3 Oplosmethode

We willen de gedempte Jacobi methode gebruiken om het discrete stelsel op te lossen. Als de matrix A een M-matrix is dan convergeert de Jacobi methode. Als Jacobi convergeert dan convergeert ook de gedempte Jacobi methode. Daarna wordt gekeken naar multigrid methoden.

3.1 De A is een M-matrix

De matrix A is een M-matrix als

1. $a_{ij} \leq 0 \forall i \neq j$, $a_{ii} \geq \sum_{j=1, j \neq i}^n |a_{ij}|$ en $\exists j \neq i$ zodanig dat $a_{ii} > a_{ij}$ wordt voldaan:
 $|a_{ij}| \leq \frac{4}{h^2} \leq a_{ii}$
2. $a_{ii} > 0$; Hieraan wordt voldaan: $a_{ii} = \frac{4}{h^2} + 200 > 0$
3. A is irreducibel. Hieraan wordt voldaan. In principe moeten we nagaan dat de matrix niet in onafhankelijke submatrices kan gesplitst worden. Echter elk $u_{i,j}$ is verbonden met de directe burens: $\{u_{i,j+1}, u_{i+1,j}, u_{i-1,j}, u_{i,j-1}\}$. Dit geldt voor elk punt dus de matrix kan niet splitsbaar zijn.

Dit betekent dat als Jacobi wordt toegepast op het systeem $Au = f$ dan convergeert deze.

3.2 gedempte Jacobi iteratie

Gebruikt wordt de gedempte Jacobi methode:

$$u_i^* = \frac{1}{a_{ii}} \left(b_i - \sum_{\substack{k=1 \\ k \neq i}}^{N^2} a_{ik} u_i^n \right) \quad (3.1)$$

$$u_i^{n+1} = \omega u_i^* + (1 - \omega) u_i^n$$

Dit lijkt veel op de Jacobi methode. De benadering u_i^* wordt gevonden door middel van de Jacobi methode. Deze wordt gecombineerd met de vorige benadering u_i^n met behulp van een gewogen gemiddelde, de parameter ω .

Herschrijf (3.1) in vector notatie:

$$u^* := D^{-1} (b + (D - A)u^n) \quad (3.2)$$

$$u^{n+1} := \omega u^* + (1 - \omega)u^n$$

Hetgeen te vereenvoudigen is tot:

$$\begin{aligned} u^* &:= D^{-1}b + (I - D^{-1}A)u^n \\ u^{n+1} &:= \mathbf{w}u^* + (1 - \mathbf{w})u^n \end{aligned} \quad (3.3)$$

Vul u^* in de onderste vergelijking van (3.3):

$$u^{n+1} = \mathbf{w}D^{-1}b + (I - \mathbf{w}D^{-1}A)u^n \quad (3.4)$$

Indien $\mathbf{w}=1$ dan zien we in (3.3) inderdaad de Jacobi iteratie weer terug:

$$u^{n+1} := D^{-1}b + (I - D^{-1}A)u^n$$

Dus de iteratiematrix voor de gedempte Jacobi is:

$$Q_{dJac} = I - \mathbf{w}D^{-1}A \quad (3.5)$$

3.3 Convergentie factor van Jacobi

We bekijken het homogene geval van de Helmholtz vergelijking:

$$\begin{aligned} -div \nabla u + 200u &= 0 \quad \text{op } \Omega = (0,1) \times (0,1) \\ u &= 0 \quad \text{op } \partial\Omega \end{aligned} \quad (3.6)$$

We willen

$$\mathbf{r}(Q_{JAC}) = \max \{ |\mathbf{l}| \mid \mathbf{l} \text{ is een eigenwaarde van } Q_{JAC} \}$$

bepalen.

We zoeken de eigenwaarden van de discrete Helmholtz operator. We kunnen de volgende vorm van eigenwaarden aannemen:

$$\mathbf{j}(\mathbf{x}, x, y) = e^{i\mathbf{x}_1 x/h} e^{i\mathbf{x}_2 y/h} \quad (3.7)^1$$

De Helmholtz vergelijking discreet geschreven geeft:

$$-\nabla_h u_{i,j} + 200u_{i,j} = \mathbf{l}u_{i,j} \quad (3.8)$$

Uitschrijven van (3.8) en na deling van $u_{i,j}$ geeft:

$$\mathbf{l} = \frac{1}{h^2} \frac{-u_{i,j+1} - u_{i+1,j} + 4u_{i,j} - u_{i-1,j} - u_{i,j-1}}{u_{i,j}} + 200 \quad (3.9)$$

Vul in (3.9) $u_{i,j} = \mathbf{j}(\mathbf{x}, ih, jh)$ dan krijgen we:

¹ Let op de schrijfwijze van de i : $i^2 = -1$ en i is een index.

$$\begin{aligned}
\mathbf{I} &= \frac{1}{h^2} \{-e^{ix_2} - e^{ix_1} + 4 - e^{-ix_1} - e^{-ix_2}\} + 200 \\
&= \frac{2}{h^2} \{2 - \cos \mathbf{x}_1 - \cos \mathbf{x}_2\} + 200 \\
\mathbf{I} &= \frac{2}{h^2} \{\sin^2 \frac{1}{2} \mathbf{x}_1 + \sin^2 \frac{1}{2} \mathbf{x}_2\} + 200
\end{aligned} \tag{3.10}$$

Pas nu \mathbf{x}_1 en \mathbf{x}_2 aan op rooster: $\mathbf{x}_1 = k\mathbf{p}h$ $\mathbf{x}_2 = l\mathbf{p}h$ met $l, k \in \mathbb{N}$. Dan krijgen we

$$\mathbf{I}_{k,l} = \frac{2}{h^2} \{\sin^2 \frac{1}{2} k\mathbf{p}h + \sin^2 \frac{1}{2} l\mathbf{p}h\} + 200$$

We weten de eigenwaarden $\mathbf{I}_{k,l}$ van A . Dan zijn de eigenwaarden $\mathbf{m}_{k,l}$ van $Q_{JAC} = I - D^{-1}A$ gelijk aan:²

$$\mathbf{m}_{k,l} = 1 - d^{-1} \mathbf{I}_{k,l} \text{ met } d = \frac{2}{h^2} + 200 \Rightarrow d^{-1} = \frac{h^2}{2 + 200h^2}$$

invullen geeft:

$$\begin{aligned}
\mathbf{m}_{k,l} &= 1 - d^{-1} \mathbf{I}_{k,l} \\
&= 1 - \frac{h^2}{2 + 200h^2} \{\sin^2 (\frac{1}{2} k\mathbf{p}h) + \sin^2 (\frac{1}{2} l\mathbf{p}h)\} - 200 \frac{h^2}{2 + 200h^2} \\
&= 1 - \frac{1}{2} \frac{h^2}{2 + 200h^2} \{2 - \cos(k\mathbf{p}h) - \cos(l\mathbf{p}h)\} - \frac{100h^2}{1 + 100h^2} \\
\mathbf{m}_{k,l} &= 1 + \frac{1}{2} \frac{h^2}{2 + 200h^2} \{\cos(k\mathbf{p}h) + \cos(l\mathbf{p}h) - 2\} - \frac{100h^2}{1 + 100h^2}
\end{aligned}$$

Het maximum treed op als $k = l = 1$:

$$\mathbf{r}(Q_{JAC}) = 1 + \frac{1}{2} \frac{h^2}{2 + 200h^2} \{\cos(\mathbf{p}h) + \cos(\mathbf{p}h) - 2\} - \frac{100h^2}{1 + 100h^2}$$

² Herinner: A heeft eigenwaarden \mathbf{I}_i . De eigenwaarden van de matrix $B = I - cA$ zijn dan: $\mathbf{I}_i = c^{-1}(1 - \mathbf{m}_i)$.
Bekijk het volgende:

$$\begin{aligned}
B - \mathbf{m}I &= 0 \\
(I - cA) - \mathbf{m}I &= 0 \\
cA - (1 - \mathbf{m})I &= 0 \\
A - c^{-1}(1 - \mathbf{m})I &= 0
\end{aligned}$$

De eigenwaarden van A zijn bekend dus gebruik $c^{-1}(1 - \mathbf{m})I = I$. Dus voor de eigenwaarden \mathbf{m} van B geldt:
 $\mathbf{I}_i = c^{-1}(1 - \mathbf{m}_i)$.

$$= 1 + \frac{h^2}{2 + 200h^2} \{ \cos(\mathbf{p}h) - 1 \} - \frac{100h^2}{1 + 100h^2}$$

$$\mathbf{r}(Q_{JAC}) = 1 + \frac{h^2}{2 + 200h^2} \left\{ -\frac{1}{2}(\mathbf{p}h)^2 + O(h^4) \right\} - \frac{100h^2}{1 + 100h^2}$$

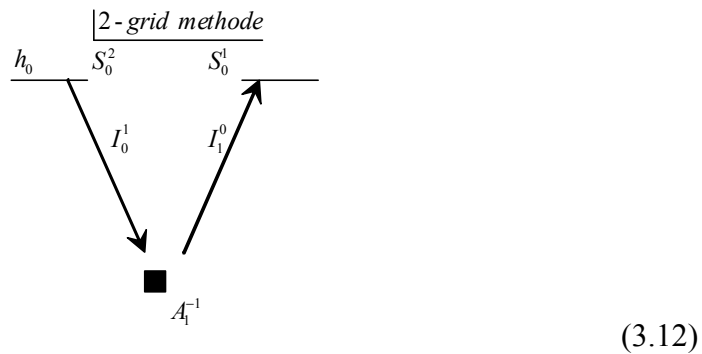
3.4 Multigrid algoritmen

Het doel is om het volgende stelsel op te lossen:

$$Au = f \quad (3.11)$$

Als A inverteerbaar is dan krijgen we: $u = A^{-1}f$. u is een vector bestaande uit $(N+1)^2$ elementen. Dit komt overeen met het aantal punten van het gebruikte grid. De matrix A bestaat dan uit $(N+1)^4$ elementen. Numeriek gezien is het niet aantrekkelijk om zo'n matrix te invertieren voor grote N . Om dit probleem te ontwijken worden multigrid methoden gebruikt.

Voor de 2-grid methode kan het volgende schema worden gebruikt:



We beginnen op een fijn grid G_0 met $h = h_0$. We passen 2 maal pre-smoothing op de vergelijking $Au = f$ toe, met startvector $u = 0$. Pre-smoothing is niets anders dan het bepalen van een benadering \hat{u} voor het stelsel $Au = f$. Dit wordt gedaan met behulp van de gedempte Jacobi methode. Noteer het defect als $d = f - A\hat{u}$. De volgende stap is om de vergelijking $Ae = d$ op te lossen. Hiermee kunnen we onze benadering corrigeren.

$$\left. \begin{array}{l} d = f - A\hat{u} \\ Ae = d \end{array} \right\} Ae = f - A\hat{u} \Rightarrow A(\hat{u} + e) = f$$

Dan is $\hat{u} + e$ de echte oplossing. Wij gaan e benaderen. Om de vergelijking $Ae = d$ op te lossen gaan we over op een grover grid G_1 met $h = h_1 = 2h_0$. We lossen $Ae = d$ (aangepast op het grid G_1) op door middel van de inverse. Numeriek kan dit ook gedaan worden door bijvoorbeeld 20 slagen Jacobi. De gevonden e wordt door interpolatie teruggebracht op het oorspronkelijke grid G_0 we hebben dan de benadering \hat{e} . Deze voegen we toe aan \hat{u} : $\hat{u} := \hat{u} + \hat{e}$. De laatste stap is post-smoothing. Dit is weer niets anders dan het verbeteren van onze benadering \hat{u} . Dit proces wordt herhaald tot de gewenste nauwkeurigheid behaald is. De keuze van de volgende startvector is dan \hat{u} .

Alvorens introduceren we wat notatie:

| | |
|------------------------|--|
| G_i | grid i met G_0 het fijnste grid en G_2 het grofste grid |
| A_i | de matrix aangepast op G_i , merk op dat $A_0 \equiv A$ |
| f_i | het rechterlid op G_i , merk op dat $f_0 = f$ |
| d_i | het defect op grid G_i |
| $h_i := \frac{2^i}{N}$ | de stapgrootte aangepast op G_i |
| I_i | De identiteit aangepast op G_i |
| I_i^{i+1} | De identiteit die zorgt voor de transformatie van G_i naar G_{i+1} |
| I_i^{i-1} | De identiteit die zorgt voor de transformatie van G_i naar G_{i-1} |
| S_i^n | n maal smoothing op G_i |

3.4.1 Algoritme van de 2-grid methode

Het schema (3.12) is als volgt als algoritme te beschrijven:

| | | |
|--|----|---------------------------------------|
| $\hat{u}_0 \leftarrow S_0^{v_1}(u_0, f_0)$ | s1 | pre-smoothing, v_1 stappen |
| $d_0 \leftarrow f_0 - A_0 \hat{u}_0$ | s2 | berekening van de residuen op G_0 |
| $d_1 \leftarrow I_0^1 d_0$ | s3 | restrictie van de residuen op G_1 |
| $A_1 e_1 = d_1$ | s4 | bepaal $e_1 = A_1^{-1} d_1$ |
| $e_0 \leftarrow I_1^0 \hat{e}_1$ | s5 | transformeer de correctie naar G_0 |
| $\hat{u}_0 \leftarrow \hat{u}_0 + e_0$ | s6 | voeg de correctie toe aan \hat{u}_0 |
| $u_0 \leftarrow S_0^{v_2}(\hat{u}_0, b_0)$ | s7 | post-smoothing, v_2 stappen |

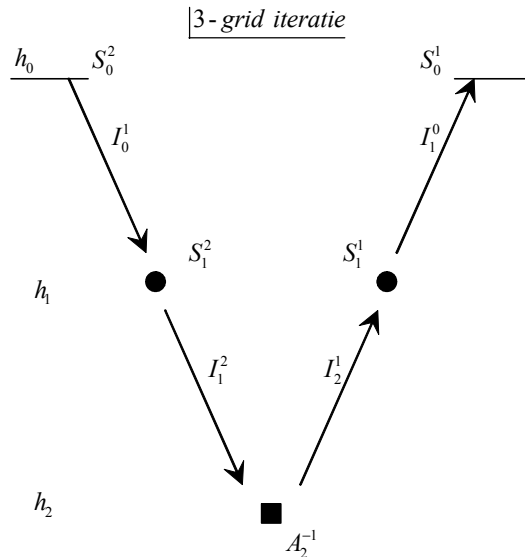
3.4.2 Algoritme van de 3-grid methode

Het algoritme van de 2-grid iteratie kunnen we uitbreiden naar een 3-grid door bij stap 4 $A_1 e_1 = d_1$ te vervangen door de 2-grid methode toegepast op de vergelijking $A_1 e_1 = d_1$.

| | | |
|---|------|---------------------------------------|
| $\hat{u}_0 \leftarrow S_0^{v_1}(u_0, b_0)$ | s1 | pre-smoothing, v_1 stappen |
| $d_0 \leftarrow b_0 - A_0 \hat{u}_0$ | s2 | berekening van de residuen op G_0 |
| $d_1 \leftarrow I_0^1 d_0$ | s3 | restrictie van de residuen op G_1 |
| <i>Het volgende niveau heeft als doel $A_1 e_1 = d_1$ te bepalen</i> | | |
| $\hat{e}_1 \leftarrow S_1^{v_1}(e_1, d_1)$ | s4.1 | v_1 pre-smoothing steps |
| $D_1 \leftarrow d_1 - A_1 \hat{e}_1$ | s4.2 | berekening van de residuen op G_1 |
| $D_2 \leftarrow I_1^2 D_1$ | s4.3 | restrictie van de residuen op G_2 |
| $A_2 e_2 = D_2$ | s4.4 | bepaal $e_2 = A_2^{-1} D_2$ |
| $e_1 \leftarrow I_2^1 e_2$ | s4.5 | transformeer de correctie naar G_1 |
| $\hat{e}_1 \leftarrow \hat{e}_1 + e_1$ | s4.6 | voeg de correctie toe aan \hat{e}_1 |
| $\hat{e}_1 \leftarrow S_1^{v_2}(\hat{e}_1, d_1)$ | s4.7 | post-smoothing, v_2 stappen |

| | | |
|--|----|---------------------------------------|
| $e_0 \leftarrow I_1^0 \hat{e}_1$ | s5 | transformeer de correctie naar G_0 |
| $\hat{u}_0 \leftarrow \hat{u}_0 + e_0$ | s6 | voeg de correctie toe aan \hat{u}_0 |
| $u_0 \leftarrow S_0^{v_2}(\hat{u}_0, b_0)$ | s7 | post-smoothing, v_2 stappen |

Hierbij hoort het volgende schema



Gebruik het schema om de iteratie matrices voor de 2-grid en 3-grid methoden te bepalen:

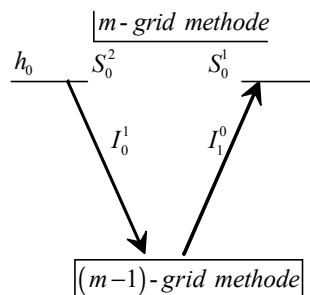
$$Q_{2-grid} = S_0^1 K_0^1 S_0^2 \quad \text{met} \quad K_0^1 = I_0 - I_1^0 A_1^{-1} I_0^1 A_0$$

$$Q_{3-grid} = S_0^1 (I_0 - I_1^0 K_1^2 I_0^1 A_0) S_0^2 \quad \text{met} \quad K_1^2 = I_1 - I_2^1 A_2^{-1} I_1^2 A_1$$

Met $S_i^n = (Q_{Jac,i})^v$

3.4.3 Algoritme van de M-grid methode

Dit proces kunnen herhalen. Hieruit verkrijgen we het zogenaamde M-grid methode.

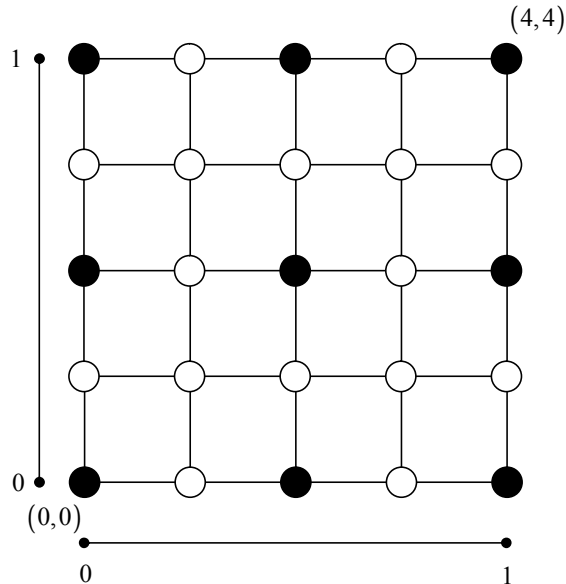


Waarin de 0-grid methode de inverse is.

3.5 De matrices nader bekeken

3.5.1 Van fijn naar grof

Al reeds zijn de identiteiten: I_i^{i+1} en I_i^{i-1} gebruikt. Echter ze zijn nog niet gedefinieerd. De identiteit die hoort bij de overgang naar een grover rooster is voor het geval dat $N=4$ als volgt:



De puntenverzameling van het fijne grid is $\{(i, j) \mid i, j \in \{0, 1, \dots, N-1, N\}\}$ na toepassen van de identiteit blijft er $\{(2i, 2j) \mid i, j \in \{0, 1, \dots, N/2\}\}$ over.

Definieer de matrix $(n+1) \times (N+1)$ C_n als volgt $C_n := \begin{pmatrix} 1 & 0 & 0 & 0 & & \\ 0 & 0 & 1 & 0 & \ddots & \\ 0 & 0 & 0 & \ddots & & \\ & \ddots & & & 1 & 0 & 0 \\ & & & & 0 & 0 & 1 \end{pmatrix}$

Definieer de $(n+1)^{n+1} \times (N+1)^{N+1}$ matrix $I_0^1 = \begin{pmatrix} C_n & 0_n & & & \\ & 0_n & C_n & & \\ & & & \ddots & \\ & & & & C_n & 0_n & \\ & & & & & 0_n & C_n \end{pmatrix}$

4 Resultaten

ω We starten door de gedempte Jacobi methode toe te passen op ons probleem voor verschillende grids. We kiezen $N \in \{4, 8, 16, 32\}$. We krijgen dan de volgende resultaten

| ω=0,8 | gedempte Jacobi | | | | | | | |
|--------------------|-----------------|----------|----------|----------|----------|----------|----------|----------|
| | N=4 | | N=8 | | N=16 | | N=32 | |
| tolerantie | 1,00E-03 | 1,00E-06 | 1,00E-03 | 1,00E-06 | 1,00E-03 | 1,00E-06 | 1,00E-03 | 1,00E-06 |
| reductie i-4 | 0,2391 | 0,3415 | 0,5432 | 0,5993 | 0,8242 | 0,8476 | 0,9462 | 0,9559 |
| reductie i-3 | 0,2555 | 0,3398 | 0,5461 | 0,6001 | 0,8286 | 0,8477 | 0,9463 | 0,9559 |
| reductie i-2 | 0,2690 | 0,3386 | 0,5500 | 0,6009 | 0,8287 | 0,8483 | 0,9463 | 0,9560 |
| reductie i-1 | 0,2802 | 0,3380 | 0,5540 | 0,6017 | 0,8289 | 0,8500 | 0,9464 | 0,9560 |
| reductie i | 0,2985 | 0,3377 | 0,5580 | 0,6025 | 0,8292 | 0,8501 | 0,9464 | 0,9560 |
| iteraties | 6 | 12 | 10 | 23 | 25 | 65 | 69 | 213 |
| fout | 1,03E-02 | 2,31E-04 | 3,84E-05 | 7,57E-05 | 0,00E+00 | 1,92E-05 | 0,00E+00 | 3,90E-06 |
| O(h ²) | 6,25E-02 | 6,25E-02 | 1,56E-02 | 1,56E-02 | 3,91E-03 | 3,91E-03 | 9,77E-04 | 9,77E-04 |

De reductie per stap is laag, dat wil zeggen dat deze waarde naar 1 gaat, met als gevolg dat het aantal benodigde iteratiestappen stijgt. We bekijken nu de 2-grid methode:

| ω=0,8 | 2-grid methode | | | | | | | |
|--------------------|----------------|----------|----------|----------|----------|----------|----------|----------|
| | N=8 | | N=16 | | N=32 | | N=64 | |
| tolerantie | 1,00E-03 | 1,00E-06 | 1,00E-03 | 1,00E-06 | 1,00E-03 | 1,00E-06 | 1,00E-03 | 1,00E-06 |
| reductie i-4 | | 0,0324 | | 0,1052 | | 0,1674 | | 0,1811 |
| reductie i-3 | | 0,0457 | | 0,1236 | 0,0879 | 0,1737 | 0,0973 | 0,1893 |
| reductie i-2 | 0,0324 | 0,0493 | 0,0660 | 0,1269 | 0,1225 | 0,1728 | 0,1436 | 0,1861 |
| reductie i-1 | 0,0457 | 0,0522 | 0,0917 | 0,1338 | 0,1639 | 0,1755 | 0,1869 | 0,1912 |
| reductie i | 0,0493 | 0,0547 | 0,1052 | 0,1344 | 0,1674 | 0,1758 | 0,1811 | 0,1898 |
| iteraties | 3 | 5 | 3 | 7 | 4 | 8 | 4 | 8 |
| fout | 6,84E-05 | 7,59E-05 | 2,73E-04 | 1,99E-05 | 1,83E-04 | 5,06E-06 | 2,98E-04 | 1,27E-06 |
| O(h ²) | 1,56E-02 | 1,56E-02 | 3,91E-03 | 3,91E-03 | 9,77E-04 | 9,77E-04 | 2,44E-04 | 2,44E-04 |

We zien dat per iteratie de reductie hoog is. Er zijn dus aanzienlijk minder iteraties nodig om de gewenste nauwkeurigheid te bereiken. Voor fijnmazige grids is het numeriek oplossen met de 2-grid methode nog steeds ondoenlijk. Er bestaat dus behoefte om uit te wijken naar een 3-grid methode.

| ω=0,8 | 3-grid methode | | | | | | | |
|--------------------|----------------|----------|----------|----------|----------|----------|----------|----------|
| | N=16 | | N=32 | | N=64 | | N=128 | |
| tolerantie | 1,00E-03 | 1,00E-06 | 1,00E-03 | 1,00E-06 | 1,00E-03 | 1,00E-06 | 1,00E-03 | 1,00E-06 |
| reductie i-4 | | 0,1095 | | 0,1726 | | 0,1910 | | 0,2063 |
| reductie i-3 | | 0,1245 | 0,0911 | 0,1739 | 0,1015 | 0,1937 | 0,1059 | 0,2070 |
| reductie i-2 | 0,0673 | 0,1263 | 0,1373 | 0,1740 | 0,1662 | 0,1957 | 0,1850 | 0,2071 |
| reductie i-1 | 0,0944 | 0,1328 | 0,1669 | 0,1749 | 0,1914 | 0,1970 | 0,2033 | 0,2061 |
| reductie i | 0,1095 | 0,1338 | 0,1726 | 0,1756 | 0,1910 | 0,1985 | 0,2045 | 0,2068 |
| iteraties | 3 | 7 | 4 | 8 | 4 | 8 | 4 | 9 |
| fout | 1,56E-04 | 1,99E-05 | 5,10E-05 | 5,05E-06 | 8,82E-05 | 1,30E-06 | 1,07E-05 | 3,15E-07 |
| O(h ²) | 3,91E-03 | 3,91E-03 | 9,77E-04 | 9,77E-04 | 2,44E-04 | 2,44E-04 | 6,10E-05 | 6,10E-05 |

We zien dat de 2-grid methode beter is bij gelijke grids. Dit komt omdat de 3-grid methode naar een grover grid afdaalt dan de 2-grid methode. Op dit grovere grid wordt de oplossing bepaald door de inverse van A , die makkelijker numeriek te bepalen is dan op het fijne grid. Door interpolatie wordt dit naar het fijnere grid teruggebracht. De 2-grid methode daarentegen bepaald op dit grid de inverse en heeft op dat niveau geen last van fouten door interpolatie.

We zien dat als het grid erg fijn is, dus N groot, dan is het verstandig om een M-grid methode te kiezen waarbij M past bij N . Bij de M-grid methode wordt de inverse van de

matrix A_M (dus de A op het grofste grid) bepaald. Het is verstandig dat A_M niet te groot is. Ter demonstratie doen we de 3-, 4- en 5-grid methode met $N = 128$. Bij de 4-grid methode geldt voor het grofste grid dat $N_{grof} = \frac{N}{2^{M-1}} = 16$, dit is numeriek nog goed te doen.

| | $\omega=0.8$ M-gridmethode | | | | | | | |
|--------------|----------------------------|----------|----------|----------|----------|----------|----------|----------|
| | N=128 | | M=3 | | M=4 | | M=5 | |
| | 1,00E-03 | 1,00E-06 | 1,00E-03 | 1,00E-06 | 1,00E-03 | 1,00E-06 | 1,00E-03 | 1,00E-06 |
| reductie i-4 | | 0,2063 | | 0,2067 | | 0,2067 | | 0,2067 |
| reductie i-3 | 0,1059 | 0,2070 | 0,1059 | 0,2072 | 0,1059 | 0,2072 | 0,1059 | 0,2072 |
| reductie i-2 | 0,1850 | 0,2071 | 0,1852 | 0,2073 | 0,1852 | 0,2073 | 0,1852 | 0,2073 |
| reductie i-1 | 0,2033 | 0,2061 | 0,2036 | 0,2064 | 0,2036 | 0,2064 | 0,2036 | 0,2064 |
| reductie i | 0,2045 | 0,2068 | 0,2047 | 0,2073 | 0,2047 | 0,2073 | 0,2047 | 0,2073 |
| iteraties | 4 | 9 | 4 | 9 | 4 | 9 | 4 | 9 |
| fout | 1,07E-05 | 3,15E-07 | 5,67E-05 | 3,16E-07 | 5,67E-05 | 3,16E-07 | 5,67E-05 | 3,16E-07 |
| $O(h^2)$ | 6,10E-05 | 6,10E-05 | 6,10E-05 | 6,10E-05 | 6,10E-05 | 6,10E-05 | 6,10E-05 | 6,10E-05 |

We zien dat het voor het gegeven probleem de 3-grid methode voldoende nauwkeurigheid geeft.

Als laatste bekijken wat de invloed is van de gedempte Jacobi. Bekijk de volgende data:

| $\omega=1.0$ | 3-grid methode | |
|--------------|----------------|----------|
| | N=16 | N=32 |
| tolerantie | 1,00E-06 | 1,00E-06 |
| reductie 1 | 0,0224 | 0,0413 |
| reductie 2 | 0,0709 | 0,0762 |
| reductie 3 | 0,2405 | 0,2087 |
| reductie 4 | 0,3647 | 0,4307 |
| reductie 5 | 0,4258 | 0,6584 |
| reductie i-4 | 0,4803 | 0,8146 |
| reductie i-3 | 0,4824 | 0,8151 |
| reductie i-2 | 0,4862 | 0,8157 |
| reductie i-1 | 0,4905 | 0,8163 |
| reductie i | 0,5051 | 0,8170 |
| iteraties | 11 | 32 |
| fout | 2,04E-05 | 4,51E-06 |
| $O(h^2)$ | 3,91E-03 | 3,91E-03 |

| $\omega=0.8$ | 3-grid methode | |
|--------------|----------------|----------|
| | N=16 | N=32 |
| tolerantie | 1,00E-06 | 1,00E-06 |
| reductie i-4 | 0,1095 | 0,1726 |
| reductie i-3 | 0,1245 | 0,1739 |
| reductie i-2 | 0,1263 | 0,1740 |
| reductie i-1 | 0,1328 | 0,1749 |
| reductie i | 0,1338 | 0,1756 |
| iteraties | 7 | 8 |
| fout | 1,99E-05 | 5,05E-06 |
| $O(h^2)$ | 3,91E-03 | 9,77E-04 |

Hier is $w=1$ gekozen, hetgeen equivalent is met Jacobi. We zien dat de reductie per stap richting de 1 gaat, hetgeen zich uit in slechte convergentie. De oorzaak is dat de oplossing niet zo glad is in het geval van de gedempte Jacobi.

Bijlage A: Resultaten

In deze bijlage staan alle gevonden resultaten.

De tolerantie geeft het stopcriteria aan. Het proces stopt als

$$\frac{\|f - Au^i\|}{\|f - Au^0\|} \leq \text{tolerantie}$$

Met $\|\cdot\|$ de maximumnorm.

De reductie van stap i is de volgende factor:

$$\text{reductie } i : d^i = \frac{\|f - Au^i\|}{\|f - Au^{i-1}\|}$$

De fout is gelijk aan de norm tussen de gevonden oplossing en de echte oplossing. Verder is de orde van h^2 gegeven.

A.1 Gedempte Jacobi

| $\omega=0,8$ | gedempte Jacobi | | | | | | | |
|--------------|-----------------|----------|----------|----------|----------|----------|----------|----------|
| | N=4 | | N=8 | | N=16 | | N=32 | |
| tolerantie | 1,00E-03 | 1,00E-06 | 1,00E-03 | 1,00E-06 | 1,00E-03 | 1,00E-06 | 1,00E-03 | 1,00E-06 |
| reductie i-4 | 0,2391 | 0,3415 | 0,5432 | 0,5993 | 0,8242 | 0,8476 | 0,9462 | 0,9559 |
| reductie i-3 | 0,2555 | 0,3398 | 0,5461 | 0,6001 | 0,8286 | 0,8477 | 0,9463 | 0,9559 |
| reductie i-2 | 0,2690 | 0,3386 | 0,5500 | 0,6009 | 0,8287 | 0,8483 | 0,9463 | 0,9560 |
| reductie i-1 | 0,2802 | 0,3380 | 0,5540 | 0,6017 | 0,8289 | 0,8500 | 0,9464 | 0,9560 |
| reductie i | 0,2985 | 0,3377 | 0,5580 | 0,6025 | 0,8292 | 0,8501 | 0,9464 | 0,9560 |
| iteraties | 6 | 12 | 10 | 23 | 25 | 65 | 69 | 213 |
| fout | 1,03E-02 | 2,31E-04 | 3,84E-05 | 7,57E-05 | 0,00E+00 | 1,92E-05 | 0,00E+00 | 3,90E-06 |
| O(h^2) | 6,25E-02 | 6,25E-02 | 1,56E-02 | 1,56E-02 | 3,91E-03 | 3,91E-03 | 9,77E-04 | 9,77E-04 |

A.2 2-grid methode

| $\omega=0,8$ | 2-grid methode | | | | | | | |
|--------------|----------------|----------|----------|----------|----------|----------|----------|----------|
| | N=8 | | N=16 | | N=32 | | N=64 | |
| tolerantie | 1,00E-03 | 1,00E-06 | 1,00E-03 | 1,00E-06 | 1,00E-03 | 1,00E-06 | 1,00E-03 | 1,00E-06 |
| reductie i-4 | | 0,0324 | | 0,1052 | | 0,1674 | | 0,1811 |
| reductie i-3 | | 0,0457 | | 0,1236 | 0,0879 | 0,1737 | 0,0973 | 0,1893 |
| reductie i-2 | 0,0324 | 0,0493 | 0,0660 | 0,1269 | 0,1225 | 0,1728 | 0,1436 | 0,1861 |
| reductie i-1 | 0,0457 | 0,0522 | 0,0917 | 0,1338 | 0,1639 | 0,1755 | 0,1869 | 0,1912 |
| reductie i | 0,0493 | 0,0547 | 0,1052 | 0,1344 | 0,1674 | 0,1758 | 0,1811 | 0,1898 |
| iteraties | 3 | 5 | 3 | 7 | 4 | 8 | 4 | 8 |
| fout | 6,84E-05 | 7,59E-05 | 2,73E-04 | 1,99E-05 | 1,83E-04 | 5,06E-06 | 2,98E-04 | 1,27E-06 |
| O(h^2) | 1,56E-02 | 1,56E-02 | 3,91E-03 | 3,91E-03 | 9,77E-04 | 9,77E-04 | 2,44E-04 | 2,44E-04 |

A.3 3-grid methode

| $\omega=0,8$ | 3-grid methode | | | | | | | |
|--------------|----------------|----------|----------|----------|----------|----------|----------|----------|
| | N=16 | | N=32 | | N=64 | | N=128 | |
| tolerantie | 1,00E-03 | 1,00E-06 | 1,00E-03 | 1,00E-06 | 1,00E-03 | 1,00E-06 | 1,00E-03 | 1,00E-06 |
| reductie i-4 | | 0,1095 | | 0,1726 | | 0,1910 | | 0,2063 |
| reductie i-3 | | 0,1245 | 0,0911 | 0,1739 | 0,1015 | 0,1937 | 0,1059 | 0,2070 |
| reductie i-2 | 0,0673 | 0,1263 | 0,1373 | 0,1740 | 0,1662 | 0,1957 | 0,1850 | 0,2071 |
| reductie i-1 | 0,0944 | 0,1328 | 0,1669 | 0,1749 | 0,1914 | 0,1970 | 0,2033 | 0,2061 |
| reductie i | 0,1095 | 0,1338 | 0,1726 | 0,1756 | 0,1910 | 0,1985 | 0,2045 | 0,2068 |
| iteraties | 3 | 7 | 4 | 8 | 4 | 8 | 4 | 9 |
| fout | 1,56E-04 | 1,99E-05 | 5,10E-05 | 5,05E-06 | 8,82E-05 | 1,30E-06 | 1,07E-05 | 3,15E-07 |
| O(h^2) | 3,91E-03 | 3,91E-03 | 9,77E-04 | 9,77E-04 | 2,44E-04 | 2,44E-04 | 6,10E-05 | 6,10E-05 |

A.4 Smoothing

| $\omega=1,0$ | 3-grid methode | |
|--------------------|----------------|----------|
| | N=16 | N=32 |
| tolerantie | 1,00E-06 | 1,00E-06 |
| reductie 1 | 0,0224 | 0,0413 |
| reductie 2 | 0,0709 | 0,0762 |
| reductie 3 | 0,2405 | 0,2087 |
| reductie 4 | 0,3647 | 0,4307 |
| reductie 5 | 0,4258 | 0,6584 |
| reductie i-4 | 0,4803 | 0,8146 |
| reductie i-3 | 0,4824 | 0,8151 |
| reductie i-2 | 0,4862 | 0,8157 |
| reductie i-1 | 0,4905 | 0,8163 |
| reductie i | 0,5051 | 0,8170 |
| iteraties | 11 | 32 |
| fout | 2,04E-05 | 4,51E-06 |
| O(h ²) | 3,91E-03 | 3,91E-03 |

A.5 M-grid methode

| $\omega=0,8$ | M-gridmethode | | | | | |
|--------------------|---------------|----------|----------|----------|----------|----------|
| | N=128 | | M=3 | | M=5 | |
| | 1,00E-03 | 1,00E-06 | 1,00E-03 | 1,00E-06 | 1,00E-03 | 1,00E-06 |
| reductie i-4 | | 0,2063 | | 0,2067 | | 0,2067 |
| reductie i-3 | 0,1059 | 0,2070 | 0,1059 | 0,2072 | 0,1059 | 0,2072 |
| reductie i-2 | 0,1850 | 0,2071 | 0,1852 | 0,2073 | 0,1852 | 0,2073 |
| reductie i-1 | 0,2033 | 0,2061 | 0,2036 | 0,2064 | 0,2036 | 0,2064 |
| reductie i | 0,2045 | 0,2068 | 0,2047 | 0,2073 | 0,2047 | 0,2073 |
| iteraties | | 4 | | 9 | | 9 |
| fout | 1,07E-05 | 3,15E-07 | 5,67E-05 | 3,16E-07 | 5,67E-05 | 3,16E-07 |
| O(h ²) | 6,10E-05 | 6,10E-05 | 6,10E-05 | 6,10E-05 | 6,10E-05 | 6,10E-05 |

Bijlage B: Listings

B.1 Generatie

genA.m

```
% genA(N)
%
% genA genereert de matrix A voor N+1 punten
%

function A=genA(N)

h=1/N;
A=sparse((4+200*h^2)*eye((N+1)^2));

A=A-diag(ones((N+1)^2-1,1), -1);
A=A-diag(ones((N+1)^2-1,1), 1);

t=N+1;
A=A-diag(ones((N+1)^2-t,1), t);
A=A-diag(ones((N+1)^2-t,1), -t);

% verwijder rand elementen
for i=1:N+1
    % verwijder steeds 4 rijen
    % onderste rand
    A(i,:)=0;    A(i,i)=1;

    % bovenste rand
    p=(N+1)*N + i;    A(p,:)=0;    A(p, p)=1;

    % linker rand
    p=1 + (i-1)*(N+1);    A(p,:)=0;    A(p,p)=1;

    % rechter rand
    p=i*(N+1);    A(p,:)=0;    A(p,p)=1;
end

A=1/h^2*A;
```

gen_u.m

```
% gen_u(N)
%
% gen_u geeft een vector terug met de exacte oplossing
% van de Helmholtz vergelijking op een grid met gridmaat N
%

function u=gen_u(N)
h=1/N;

u=zeros((N+1)^1, 1);

for i=1:N+1
    for j=1:N+1
        u(i + (j-1)*(N+1))=(h*(i-1))^4*(h*(j-1))^3;    % x^4*y^3;
    end
end

end
```

genRL_f.m

```
% genRL_f(N)
%
% genRL_f berekend het rechterlid ten gevolge van de
% waarde van de functie f in die punten. Uitvoer is een
% vector behorende bij een grid met gridmaat N
%

function f=genRL_f(N)
h=1/N;
```

```

f=-inf*ones((N+1)^2, 1);

for i=2:N
    for j=2:N
        x=h*(i-1);
        y=h*(j-1);
        f(i + (j-1)*(N+1))=-6*(x^2*y*(x^2+2*y^2))+200*x^4*y^3;
    end
end
end

```

genRL_u.m

```

% genRL_u(N)
%
% genRL_u berekend het rechterlid u ten gevolge van de
% bekende waarde van u op de rand. Uitvoer is een
% vector behorende bij een grid met gridmaat N
%
function u=genRL_u(N)

u=-inf*ones((N+1)^2, 1);
h=1/N;

% verwijder rand elementen
for i=1:N+1
    % maak de vier rijen
    % onderste rand
    u(i)=((i-1)*h)^4*(0)^3; % x^4*y^3

    % bovenste rand
    p=(N+1)*N + i;          u(p)=((i-1)*h)^4*(1)^3; % x^4*y^3

    % linker rand
    p=1 + (i-1)*(N+1);      u(p)=(0)^4*((i-1)*h)^3; % x^4*y^3

    % rechter rand
    p=i*(N+1);              u(p)=(1)^4*((i-1)*h)^3; % x^4*y^3
end

u=1/h^2*u;

```

genId_fijn2grof.m

```

% genId_fijn2grof(N)
%
% genId_fijn2grof genereert de transformatiematrix van N
% punten naar N/2=n punten door tussenliggende punten weg
% te laten
%
% merk op dat n=2^(k+1) met k een natuurlijk getal
%
function I=genId_fijn2grof(N)
n=N/2;
C=sparse(zeros(n+1, 2*n+1));
for i=1:n+1
    C(i,2*i - 1)=1;
end

I=sparse(zeros((n+1) * (n+1), (N+1) * (N+1)));

for i=1:n+1
    I(1+(i-1)*(n+1) : i*(n+1) , 1+2*((i-1)*(N+1)) : 2*((i-1)*(N+1)) + 2*n + 1 )=C;
end

```

genId_grof2fijn.m

```

% genId_grof2fijn(n)
%
% genId_grof2fijn genereert de transformatiematrix van n
% punten naar 2*n=N punten door de nieuwe tussenliggende
% punten te interpoleren
%

```

```

% merk op dat n=2^k met k een natuurlijk getal
%
function I=genId grof2fijn(n)
N=2*n;

D=sparse(N+1,n+1);

for i=1:n+1
    D(2 * i - 1, i)=1;
end

for i=1:n
    D(2 * i, i)=1/2;
    D(2 * i, i + 1)=1/2;
end

E=sparse(.5*D);

I=sparse((N+1)^2, (n+1)^2);

for i=1:n+1
    I(2*(i-1)*(N+1) + 1: 1 + 2*(i-1)*(N+1) + N , 1+(i-1)*(n+1) : (i-1)*(n+1) + (n+1))=D;
end

for i=1:n
    I(2*(i-1)*(N+1) + 1+N+1: 1 + 2*(i-1)*(N+1) + N +N+1, 1+(i-1)*(n+1) : (i-1)*(n+1) + (n+1)
+ (n+1))=[E,E];
end

I=sparse(I);

```

B.2 Calculatie

A_row_product.m

```

% A_row_product(N, i, f)
%
% A_row_product rekent het product uit van de i-de rij
% van A met de vector f. Er wordt uitgegaan van
% een matrix A die bij de gridmaat N hoort (voor
% ons probleem).
%
function r=A_row_product(N, i, f)

% bepaal de positie
x=mod(i-1, N+1);
y=floor((i-1)/(N+1));

h=1/N;
if ((x == 0) | (x == N) | (y == 0) | (y == N))
    % een rand
    r=f(i);
else
    % een intern punt...
    %[-1, , -1 ,4, -1 , -1]
    %[i-N-1, i-1, i, i+1, i+N+1]
    r=-f(i-N-1)-f(i-1)+(4+200*h^2)*f(i)-f(i+1)-f(i+N+1);
end
r=1/h^2*r;

```

A_product.m

```

% A_product(N, f)
%
% A_product geeft het product terug van A*f
% door middel van rij producten. Dit omdat
% dan niet de matrix A opgeslagen hoeft
% te worden in het geheugen.
%
function r=A_product(N, f)
for i=1:(N+1)^2
    r(i)=A_row_product(N, i, f);
end

```

```
end
r=r';
```

I01_product

Merk op: Deze functie is achteraf geïmplementeerd om te voorkomen dat Matlab in geheugen problemen heeft voor bijvoorbeeld het geval $N=128$. Dus de functie is enigszins ad-hoc geïmplementeerd.

```
% I01_product (N, f)
%
%
function v=I01_product(N, f)
global ones1
global ones2
p=1;
v(p:p+N+1)=ones1(1:1+N+1);
p=p+N+1;
p=p+N+1;

for i=1:N/2
    v(p:p+N)=ones1(1:N+1);
    p=p+N+1; % skip
    p=p+N+1; % skip
end

f=v' .* f;
v=f(v==1);
```

I10_product

Merk op: Deze functie is achteraf geïmplementeerd om te voorkomen dat Matlab in geheugen problemen heeft voor bijvoorbeeld het geval $N=128$. Dus de functie is enigszins ad-hoc geïmplementeerd.

```
% I10_product (n, f)
%
%
function v=I10_product(n, f)
global ones1
global ones2

N=2*n;

for i=1:n
    m0=ones1(1:N+1);
    f0=f(1+(i-1)*(n+1):1+(i-1)*(n+1)+n);
    R1(m0==1)=f0;

    m1=ones1(1:N+1);
    f1=f(1+(i)*(n+1):1+(i)*(n+1)+n);
    R2(m1==1)=f1;
    R2=.5*(R1+R2);

    m0=.5*f0(1:n)'+.5*f0(2:n+1)';
    m1=1-ones1(1:N+1);
    T1=0*R1;
    T1(m1==1)=m0;

    m0=.25*(f1(1:n)+f0(1:n))'+.25*(f1(2:n+1)+f0(2:n+1))';
    m1=1-ones1(1:N+1);
    T2=0*R1;
    T2(m1==1)=m0;

    v(1+(i-1)*2*(N+1):(i-1)*2*(N+1)+2*(N+1))=[T1+R1, T2+R2];
end

i=n+1;
m0=mod(1:N+1,2);
f0=f(1+(i-1)*(n+1):1+(i-1)*(n+1)+n);
R1(m0==1)=f0;
```

```

m0=.5*f0(1:n)'+.5*f0(2:n+1)';
m1=1-mod(1:N+1,2);
T1=0*R1;
T1(m1==1)=m0;
v(1+(i-1)*2*(N+1):(i-1)*2*(N+1)+1*(N+1))=[T1+R1];
v=v';

```

B.3 Methoden

djacobi_stap.m

```

% djacobi_stap(N, f, omega, u0)
%
% djacobi_stap doet 1 gedempte Jacobi stap voor een
% grid met gridmaat N, vector f, dempingsparameter
% omega en startvector u0
%
function u1=djacobi_stap(N, f, omega, u0)
n=size(u0);
n=n(1);

for i=1:n
    vs=u0;
    vs(i)=0;

    Aii=A row product(N, i, [0*[1:i-1],1,0*[i+1:n]]');
    u ster i=1/Aii*(f(i)-A row product(N, i, vs));
    u1(i)=omega*u ster i + (1-omega) * u0(i);
end

u1=u1';

```

djacobi.m

```

% djacobi(N, f, omega, u0, TOL)
%
% djacobi lost iteratief Au=f op voor een matrix A die
% bij de gridmaat N hoort, de bijbehorende vector f en
% de gewenste dempingsparameter omega.
% u0 is de startvector en TOL is de gewenste tolerantie
%
function m=djacobi(N, f, omega, u0, TOL)
u_ster=u0;

defect_0 = f - A_product(N, u_ster);
defect_nieuw = defect_0;

% iteratief de stappen doen totdat de gewenste nauwkeurigheid
% bereikt is
stap=1;
abs_defect_nieuw=2* TOL * max(abs(defect_0));
while (abs_defect_nieuw/max(abs(defect_0)) > TOL)
% de stap zelf besteden we uit
    u ster = djacobi stap (N, f, omega, u ster);

    defect_oud=defect_nieuw;
    defect_nieuw=f - A_product(N, u_ster);

    abs_defect_nieuw=max(abs(defect_nieuw));
    red(stap)=abs_defect_nieuw/max(abs(defect_oud));
    stap=stap+1;
    abs_defect_nieuw;
end

l=length(red);
if (l > 5)
    red=red(l-4:l);
end

m.stappen=stap-1;
m.red=red';
m.u_ster=u_ster;
m.defect=abs_defect_nieuw;

```

a_Mgrid_cycle.m

```
% a_Mgrid_cycle(M, N, f0, u0, omega)
%
% a_Mgrid_cycle voert het multigrid algoritme uit voor
% M grids voor een grid met gridmaat N en f0 op het
% fijnste grid, u0 als startvector en omega als
% dempingsparameter voor Jacobi.
%
% omega is optioneel. Indien deze niet opgegeven wordt
% wordt omega=0.8 gebruikt
%

function u=a_Mgrid_cycle(M, N, f0, u0, omega)

if ( nargin < 5 ) omega=0.8; end

% s1
u0_dak = d_jacobi_stap (N, f0, omega, u0);
u0_dak = d_jacobi_stap (N, f0, omega, u0_dak);

% s2
d0 = f0 - A_product(N, u0_dak);

% s3
d1 = I01_product(N,d0);

% recursieve aanroep van het multigrid algoritme
if (M > 1)
    % s4.1-4.7
    e1=a_Mgrid_cycle(M-1, N/2, d1, 0*d1, omega);
else
    % s4 We hebben de keuze om de echte inverse te gebruiken of 20 slagen Jacobi
    e1=ijacobi(N/2, d1, omega, 0*d1);

    % e1=genA(N/2)^-1 * d1;
end

% s5
e0 = I10_product(N/2, e1);

% s6
u0_dak = u0_dak + e0;

% s7
u = d_jacobi_stap_(N, f0, omega, u0_dak);
```

a_Mgrid.m

```
% a_Mgrid(M, N, TOL, omega)
%
% a_Mgrid voert de M grid methode uit op een grid
% met gridmaat N en tolerantie TOL. Verder wordt
% de gedempte Jacobi iteratie gebruikt met
% dempingsparameter omega.
%

function r=a_Mgrid(M, N, TOL, omega)
f0=max(genRL_u(N), genRL_f(N));

global ones1
global ones2

ones1=mod(1:(N+1)^2,2);
ones2=1-ones1;

%u = zeros((N+1)^2,1);
u=1/N^2*genRL_u(N);
u(u==--inf)=0;

defect 0 = f0 - A_product(N, u);
defect nieuw = defect 0;
```

```

stap=1;
abs_defect_nieuw=2* TOL * max(abs(defect_0));
while (abs_defect_nieuw/max(abs(defect_0)) > TOL)
    u=a Mgrid cycle(M, N, f0, u, omega);

    defect_oud=defect_nieuw;
    defect_nieuw=f0 - A_product(N, u);

    abs_defect_nieuw=max(abs(defect_nieuw));
    red(stap)=abs_defect_nieuw/max(abs(defect_oud));
    stap=stap+1;

end

l=length(red);
if (l > 5)
    red=red(l-4:l);
end

r.stappen=stap-1;
r.red=red';
r.u ster=u;
r.defect=abs_defect_nieuw;

```


Bijlage C: Opgave