Universiteit Utrecht

Bachelorscriptie

Optimaliseren van iteratieve oplosmethoden voor de Helmholtz-vergelijking

P.J. Wolters

Begeleider
Dr. T. van Leeuwen

9 augustus 2017

Inhoudsopgave

Hoofdstuk 1

Inleiding

Een belangrijke vergelijking in veel wiskundige toepassingen is de Helmholtzvergelijking. Deze partiële differentiaalvergelijking vindt zijn oorsprong in de 19e eeuw en heeft een sterke relatie met golfvergelijking en bijbehorende natuurkundige vraagstukken. Gedurende de afgelopen eeuwen is er uitvoerig onderzoek naar gedaan, wat geresulteerd heeft in een goed analytisch begrip van de vergelijking. Zo zijn er voor de meeste eenvoudige geometrieën exacte oplossingen bekend. Door de opkomst van krachtige computers is het mogelijk geworden om oplossingen van onder andere partiële differentiaalvergelijkingen numeriek te benaderen. Voor de Helmholtzvergelijking zijn de gebruikelijke methoden niet toereikend. Om dit probleem te overkomen zijn verscheidene methoden verbeterd en ontwikkeld, maar er is zeker nog ruimte voor verbetering. In deze scriptie zullen we een van deze methoden onderzoeken.

Ter introductie zullen we de verwantschap met de golfvergelijking verklaren en aantal analytische oplossingen behandelen. Vervolgens zullen we een introductie geven voor twee numerieke methoden; SSOR en CG. Deze twee methoden vormen de basis voor een derde methode; CGMN. Deze methode zullen we verder analyseren. Met deze analyse zullen we een aantal verbeteringen onderzoeken. Als laatste zullen we nog een aantal numerieke experimenten uitvoeren om onze conclusies te testen.

Hoofdstuk 2

De Helmholtz vergelijking

Het centrale onderwerp van deze scriptie zal het oplossen van de Helmholtz-vergelijking zijn. Dit is de volgende partiële differentiaalvergelijking

$$(\nabla^2 + k^2)u = -g, \quad x \in \mathbb{R}^n \tag{2.1}$$

We kunnen deze vergelijking als volgt interpreteren; laat c de voortplantingssnelheid van een golf zijn, bijvoorbeeld de geluidssnelheid. Deze kan verschillen over het domein. De frequentie van de golven wordt gegeven door f in Hertz. De golflengte λ wordt dus gegeven door c/f. We definiëren het golfgetal $k = 2\pi f/c$, waaruit volgt dat $\lambda = 2\pi/k$. Tenslotte is de g de functie die de golf voortplant.

Om tot een unieke oplossing te komen moeten we randvoorwaarden stellen. We onderscheiden de volgende twee gevallen.

1. Het domein is begrensd, bijvoorbeeld de eenheidscirkel. In dit geval kunnen we kiezen tussen Neumann of Dirichlet randvoorwaarden. Deze worden respectievelijk gegeven door $u'(x) = f(x)$, $u(x) = f(x)$ voor x op de rand van het domein. Hierbij is f een continue functie. Als $f \equiv 0$ spreken we van homogene randvoorwaarden.

2. Het domein is onbegrensd, bijvoorbeeld $\mathbb{R}^2$. We stellen in dit geval de eis dat de energie uit een zekere bron komt, en wegstroomt richting oneindig. Hiertoe eisen we dat $u(x)$ voldoet aan de Sommerfeld radiation condition: $\partial u/\partial \vec{n} - iku = 0$. Waarbij $\vec{n}$ de uit de bron wijzende eenheidsvector is.

2.1 Connectie met de golfvergelijking

De vergelijking kan gezien worden als een tijdonafhankelijke variant van de golfvergelijking, wat we nu zullen laten zien. De golfvergelijking is de volgende vergelijking

$$\frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u. \tag{2.2}$$

hierin is ∇^2 de ruimtelijke, n-dimensionale Laplaciaan en interpreteren we c op dezelfde manier als in (2.1). In het algemeen is de functie u afhankelijk van een tijds variabele t, en een aantal ruimtelijke variabelen $x_1, ..., x_n$. Om de analyse hiervan te vereenvoudigen kan men scheiding van variabelen toepassen. We nemen hiervoor aan dat $u(x,t) = X(x)T(t)$. Als we dit substitueren in (2.2) verkrijgen we de volgende vergelijking:

$$\frac{d^2T}{dt^2}X = c^2T\nabla^2X$$

Wat herschreven kan worden tot:

$$\frac{1}{Tc^2}\frac{d^2T}{dt^2} = \frac{\nabla^2X}{X}, \tag{2.3}$$

We zien dat de linkerzijde slechts afhankelijk is van t en de rechterzijde slechts van x. Hieruit concluderen we dat zowel het linker- als het rechterlid gelijk zijn aan een constante. Deze constante kunnen we vinden door gebruik te maken van de randvoorwaarden. Als deze scheidingsconstante gelijk is aan $-k^2$ voor een zekere $k > 0$, verkrijgen we uit (2.3) de volgende vergelijkingen

$$(\nabla^2 + k^2)X = 0 \tag{2.4}$$

$$\left(\frac{d^2}{dt^2} + k^2\right)T = 0 \tag{2.5}$$

We herkennen hier twee Helmholtz vergelijkingen in; (2.4) n-dimensionaal en (2.5) 1-dimensionaal. De oplossing voor de golfvergelijking wordt in dit geval dus gegeven door een combinatie van oplossingen van deze twee vergelijkingen.

2.2 Analytische oplossingen

2.2.1 Oplossing op een interval

Voordat we ons verder richten op de numerieke aanpak, zullen we een aantal situaties behandelen waarin analytische oplossingen te vinden zijn. Dit zal ons een indicatie geven van hoe de oplossingen zich gedragen en de noodzaak aantonen van het gebruik van numerieke methoden voor complexere situaties. Als eerste beschouwen we het volgende 1-dimensionale geval van de Helmholtz-vergelijking.

$$\frac{d^2}{dx^2}u(x) + k^2u(x) = f(x), x \in (0,a) \tag{2.6}$$

Met als rand voorwaarden $u(0) = u(a) = 0$ en k constant over het hele domein. De bijbehorende eigenwaarden en eigenfuncties van dit probleem zijn

$$\lambda_n = \left(\frac{n\pi}{a}\right)^2, \quad g_n(x) = \sin\left(\frac{n\pi x}{a}\right), \quad n \in \mathbb{N} \tag{2.7}$$

We drukken f en u uit in eigenfuncties;

$$f(x) = \sum_{n=1}^{\infty} f_n g_n(x), \quad u(x) = \sum_{n=1}^{\infty} u_n g_n(x) \tag{2.8}$$

Als we onze nieuwe uitdrukking voor u combineren met (2.6) verkrijgen we de volgende gelijkheid:

$$\frac{d^2}{dx^2}u(x) + k^2 u(x) = \sum_{n=1}^{\infty} \left(-\lambda_n + k^2\right) u_n g_n(x) \tag{2.9}$$

We kunnen de factoren f_n van de expansie van f in eigenfuncties vinden. Als we dit vervolgens combineren met (2.9) vinden we de volgende uitdrukking voor de factoren u_n.

$$u_n = \frac{f_n}{k^2 - \lambda_n}$$

We kunnen hieruit het volgende concluderen over de oplossing $u(x)$

1. De oplossing is uniek als voor alle $n \in \mathbb{N}$, $k^2 \neq \lambda_n$.
2. De oplossing bestaat niet als $k^2 = \lambda_n$ voor een zekere n, met $f_n \neq 0$. Een voorbeeld van deze situatie ontstaat uit de volgende condities: $a = \pi$, $k = 2$ en $f(x) = \sin(2x)$. Hieruit volgt dat $\lambda_2 = k^2 = 4$ terwijl $f_2 = 1$.
3. De oplossing is niet uniek als $k^2 = \lambda_{n'}$ en $f_{n'} = 0$ voor een zekere $n' \in \mathbb{N}$. Voor een voorbeeld hiervoor kiezen we dezelfde condities als in de vorige situatie, met als verschil dat $f \equiv 0$. Wederom geldt dat $\lambda_2 = k^2$, maar deze keer is $f_2 = 0$. Hieruit volgt dat $u(x) = c\sin(2x)$ een oplossing is voor alle $c \in \mathbb{R}$.

2.2.2 Oplossingen op een vierkant domein

In het vorige voorbeeld was het mogelijk de oplossing als een som van eigenfuncties uit te drukken. Dit is niet altijd mogelijk. De volgende twee voorbeelden dienen om de complexiteit van de analytische oplossingen aan te tonen.

We beschouwen het volgende 2-dimensionale probleem:

$$\left(\nabla^2 + k^2\right) u(x, y) = f(x, y) \tag{2.10}$$

Met domein $[0, a] \times [0, a]$, k constant over het gehele domein en homogene dirichlet randvoorwaarden; $u(0, y) = u(a, y) = u(x, 0) = u(a, y) = 0$. De eigenwaarden en eigenfuncties van (2.10) met $f \equiv 0$ zijn:

$$\lambda_{mn} = \pi^2 \left(\frac{m^2 + n^2}{a^2}\right) \quad m, n \in \mathbf{N} \tag{2.11}$$

$$g_{mn}(x, y) = \sin\left(\frac{m\pi x}{a}\right) \sin\left(\frac{n\pi x}{a}\right) \tag{2.12}$$

Om dezelfde redenen als in het vorige voorbeeld, vinden we geen unieke oplossingen als $k^2 = \lambda_{mn}$ voor zekere m en n. De oplossing van dit probleem is bekend. Voor de uitwerking ervan verwijzen we naar [6] we laten deze achterwege.

$$u(x,y) = \iint_{[0,a]^2} f(x,y)G(x,y,\xi,\eta)\,d\eta\,d\xi \tag{2.13}$$

$$G(x,y,\xi,\eta) = \frac{2}{a}\sum_{n=1}^{\infty} \frac{\sin(p_n x)\sin(p_n \xi)}{\beta_n \sinh(\beta_n a)} H_n(y,\eta) \tag{2.14}$$

$$p_n = \frac{\pi n}{a}, \quad \beta_n = \sqrt{p_n^2 - k^2} \tag{2.15}$$

$$H_n(y,\eta) = \begin{cases} \sinh(\beta_n \eta)\sinh[\beta_n(a-y)] & \text{voor} a \geq y > \eta \geq 0 \\ \sinh(\beta_n \eta)\sinh[\beta_n(a-\eta)] & \text{voor} a \geq \eta > y \geq 0 \end{cases} \tag{2.16}$$

Bovenstaande vergelijkingen geven de oplossing van (2.10). De gekozen Greense functie $G(x)$ is een van de mogelijke representaties. Voor de expliciete oplossing moeten we (2.13) oplossen. Dit is in het algemeen zeker niet mogelijk. Daarnaast hebben we in ons voorbeeld voor eenvoudige randvoorwaarden gekozen en een constant golfgetal, wat de analyse al vereenvoudigd. In de meeste gevallen zijn we dus aangewezen op numerieke methode. We merken als laatste nog op dat de oplossingen een periodiek karakter hebben met golfgetal k.

2.2.3 Onbegrensd domein

Tot slot van dit hoofdstuk zullen we nog de volgende variant van de Helmholtz vergelijking behandelen.

$$\nabla^2 U(x) + k^2 U(X) = -f(x), \quad x \in \mathbb{R}^n \tag{2.17}$$

Waarbij f een functies is met compacte drager. Voor een unieke oplossingen stellen we de eerder geïntroduceerde Sommerfeld radiation condition als randvoorwaarde. We kunnen de oplossing van de vergelijking nu vinder door de volgende convolutie

$$U(x) = (G * f)(x) = \int_{\mathbf{R}^n} G(x-y)f(y)\,dy = \int_{\operatorname{supp} f} G(x-y)f(y)\,dy \tag{2.18}$$

Waarbij G wederom de Greense functie is, die we in dit geval kunnen vinden door (2.17) op te lossen met $f(x) = \delta(x)$, de dirac delta functie. De Greense functies voor $n \leq 3$ worden gegeven door [4]:

$$G(x) = \begin{cases} \dfrac{ie^{ik|x|}}{2k} & n = 1 \\ \dfrac{i}{4}H_0^{(1)}(k|x|) & n = 2, \text{ waarbij } H \text{ een Hankel functie is} \\ \dfrac{e^{ik|x|}}{4\pi|x|} & n = 3 \end{cases} \tag{2.19}$$

Net als in het vorige voorbeeld is het wederom niet mogelijk een expliciete oplossing te geven. Dus ook in deze situaties kunnen numerieke methoden uitkomst bieden. Verder hebben alle oplossingen wederom een periodiek karakter met golfgetal k.

Hoofdstuk 3

Discretisatie van de Helmholtz vergelijking

3.1 Deelinleiding

In dit gedeelte zullen we eerst een introductie geven in de discretisatie. Vervolgens zullen we de methoden SOR en CG (Conjugate Gradient) introduceren. Deze methoden combineren we vervolgens om tot een nieuwe methode te komen, CGMN. Deze laatst genoemde methode gaan verder analyseren en optimaliseren.

3.2 Disrectisatie

Voor dit gedeelte is gebruik gemaakt van [1].

Voor het discretiseren van de Helmholtzvergelijking zullen we gebruik maken van de eindige-differentie-methode (EDM). We zullen deze methode introduceren door het volgende voorbeeld. We beschouwen de 2-Dimensionale Helmholtzvergelijking op het eenheidsvierkant

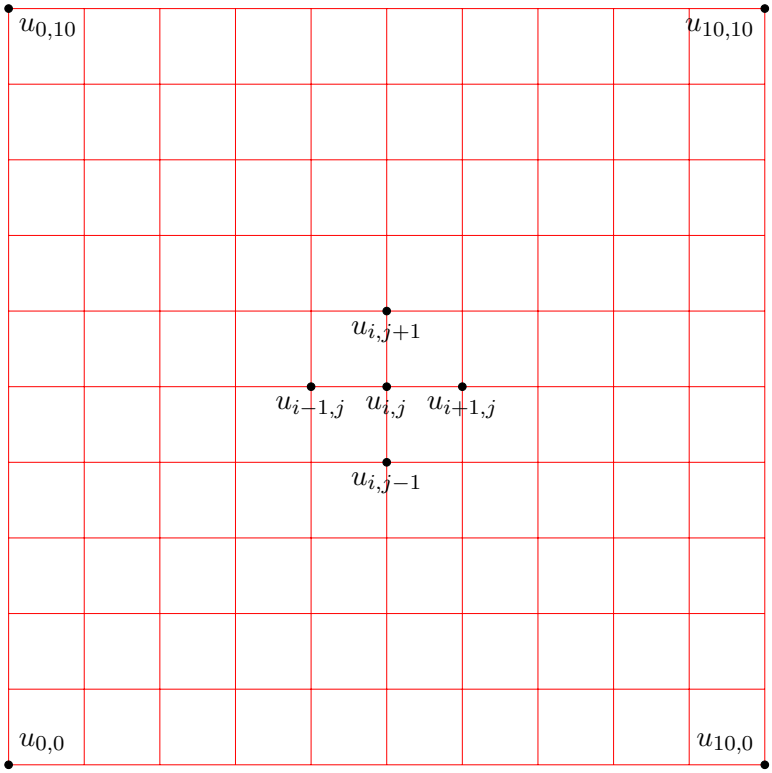
$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + k(x,y)^2\right) u(x,y) = -f(x,y) \tag{3.1}$$

We delen het vierkant op in een uniform rooster met $n+1$ cellen in elke coördinaat-richting. We hanteren de volgende notatie:

$$u_{ij} = u(x_i, y_j), \quad (x_i, y_j) = (ih, jh), \quad h = \frac{1}{n+1}, \quad i,j \in \{1,\ldots,n\} \tag{3.2}$$

Oftewel, $u_{i,j}$ is gelijk aan de waarde van u op het hoekpunt (i,j). We gebruiken dezelfde notatie voor k en f. Zie figuur 3.2 voor een visuele weergave. We benaderen de laplaciaan als volgt:

$$\nabla^2(u_{i,j}) \approx \frac{u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{i,j}}{h^2} \tag{3.3}$$



Figuur 3.1: Discretisatie van het eenheidsvierkant met $n = 9$, geïnspireerd op fig 7.2 in [1]

Waar we gebruik hebben gemaakt van de volgende benadering voor beide partiële afgeleiden

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + O(h^4) \tag{3.4}$$

Als we deze benadering terug stoppen in (3.1) krijgen we voor de inwendige punten de volgende lineaire vergelijkingen:

$$\frac{u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{i,j}}{h^2} + (k_{i,j})^2 u_{i,j} = -f_{i,j} \tag{3.5}$$

$$i, j \in \{1, \ldots, n\} \tag{3.6}$$

Voor de punten op de rand van het domein gebruiken we vanzelfsprekend de randvoorwaarden. Als we bijvoorbeeld Dirichlet randvoorwaarden gebruiken, $u(x, y) = g(x, y)$ voor $(x, y) \in \partial\Omega$, dan krijgen we $u_{0,j} = g_{0,jh}$, $u_{i,0} = g_{ih,0}$ enz. We kunnen de uit vergelijkingen (3.5) in een lineair stelsel uitdrukken; $A\mathbf{u} = \mathbf{f}$. Voor de vectoren $\mathbf{u}$ kiezen we de volgende ordening: $\mathbf{u}^T = (u_{1,1}, u_{1,2}, \ldots, u_{1,n}, u_{2,1}, \ldots, u_{n,n})$. Voor $\mathbf{f}$ en $\mathbf{k}$ gebruiken we uiteraard dezelfde ordening. De matrix A is dus een $N \times N$ matrix, met $N = n^2$

We kunnen de matrix A als volgt opsplitsen, $A = A' + \mathrm{diag}(\mathbf{k})$, waarbij A' de differentie matrix is van de Laplaciaan. Als voorbeeld geven we hier de matrix A' met $N = 3$:

$$A' = \begin{pmatrix} -4 & 1 & & 1 & & & & & \\ 1 & -4 & 1 & & 1 & & & & \\ & 1 & -4 & 0 & & 1 & & & \\ 1 & & 0 & -4 & 1 & & 1 & & \\ & 1 & & 1 & -4 & 1 & & 1 & \\ & & 1 & & 1 & -4 & 0 & & 1 \\ & & & 1 & & 0 & -4 & 1 & \\ & & & & 1 & & 1 & -4 & 1 \\ & & & & & 1 & & 1 & -4 \end{pmatrix} \tag{3.7}$$

3.2.1 Eigenwaarden

Voor de analyse van de numerieke methoden die we gaan gebruiken hebben we de eigenwaarden van A nodig. Deze vinden we op de volgende manier. We kunnen voor elke richting waarin we de gradiënt benaderen een afzonderlijke matrix A'' creëren;

$$A'' = h^{-2} \begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{pmatrix} \tag{3.8}$$

De eigenwaarden van matrix A'' zijn eenvoudiger te vinden dan die van A'. De matrix komt overeen met de 3-punts benadering van de tweede afgeleide:

$$\frac{x_{i-1} - 2x_i + x_{i+1}}{h^2} = \lambda x_i \tag{3.9}$$

$$x_0 = x_{n+1} = 0 \tag{3.10}$$

Dit kunnen we herschrijven naar:

$$x_{i+1} = (2 + h^2\lambda)x_i - x_{i-1} \tag{3.11}$$

Als we vervolgens aannemen dat $x_1 \neq 0$ kunnen we ons stelsel vergelijkingen herschalen zodat $x_1 = 1$. Dit geeft de volgende recurrente betrekking:

$$x_0 = 0, \quad x_1 = 1, \quad x_{i+1} = (2 + h^2\lambda)x_i - x_{i-1} \tag{3.12}$$

Deze relatie komt overeen met de Chebyshev polynoom van de tweede soort; $x_{i+1} = U_i(y)$ met $2y = 2 + h^2\lambda$. Gebruikmakend van de eis $x_{n+1} = 0$ concluderen we dat $U_n(y) = 0$. We kunnen de eigenwaarden van ons stelsel

nu vinden door de nulpunten van $U_n(y)$ te berekenen. De nulpunten zijn bekend, $y_k = \cos\left(\frac{k\pi}{n+1}\right)$, en we leiden hieruit de eigenwaarden af:

$$2y_k = 2 + h^2\lambda_k \tag{3.13}$$

$$\lambda_k = \frac{2}{h^2}\left(1 - \cos\left(\frac{k\pi}{n+1}\right)\right) \tag{3.14}$$

$$\lambda_k = -\frac{4}{h^2}\sin^2\left(\frac{k\pi}{2(n+1)}\right) \tag{3.15}$$

Terug naar onze originele matrix A. We kunnen deze matrix op de volgende wijze schrijven.

$$A = \left(A'' \oplus A''\right) + \mathrm{diag}(k) \tag{3.16}$$

Hierin is $\oplus$ de Kronecker som, gedefinieerd als $A \oplus B = A \otimes I + I \otimes B$.
$\otimes$ is het Kronecker product, gedefinieerd als:

$$A \otimes B = \begin{pmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{n1}B & \dots & a_{nn}B \end{pmatrix} \tag{3.17}$$

De Kronecker som heeft als handige eigenschap dat de eigenwaarden van de som uitgedrukt kunnen worden in een lineaire combinatie van de originele eigenwaarden. We verwijzen hiervoor naar de volgende stelling:

Theorem 3.2.1 (Eigenwaarden van $\oplus$[3])**.** *Laat $A \in \mathbb{R}^{n\times n}$ met eigenwaarden λ_i, $1 \le i \le n$, en $A \in \mathbb{R}^{m\times m}$ met eigenwaarden μ_i, $1 \le i \le m$. Dan heeft $A \oplus B$ mn eigenwaarden, gegeven door:*

$$\lambda_1 + \mu_1, \dots, \lambda_1 + \mu_m, \dots, \lambda_n + \mu_m \tag{3.18}$$

Bewijs. Het bewijs is makkelijk af te leiden uit de volgende gelijkheid: Laat a en b eigenvectoren zijn van respectievelijk A en B. Dan geldt dat

$$(I_n \otimes A + B \otimes I_m)(b \otimes a) = (b \otimes Aa) + (Bb \otimes a) \tag{3.19}$$

$$= (b \otimes \lambda a) + (\mu b \otimes a) \tag{3.20}$$

$$= (\lambda + \mu)(a \otimes b) \tag{3.21}$$

□

Dit kunnen we combineren met (3.15) om de eigenwaarden van A' te vinden,

$$\lambda_{km}(A') = -\frac{4}{h^2}\left(\sin^2\left(\frac{m\pi}{2(n+1)}\right) + \sin^2\left(\frac{k\pi}{2(n+1)}\right)\right) \tag{3.22}$$

Ten slotte moeten we alleen nog de invloed van diag($\mathbf{k}$) op de eigenwaarden onderzoeken. Voor een homogeen medium, dat wil zeggen k is constant, worden de eigenwaarden gegeven door er simpelweg k bij op te tellen. Als het medium heterogeen is, k is variabel, kunnen we aan de hand van de volgende stelling boven en ondergrenzen vinden:

Theorem 3.2.2 (Weyl's ongelijkheid, Matrix versie[10])**.** *Laat $M = N + K$ met M, N, K hermitische $n \times n$ matrices met respectievelijk de volgende geordende eigenwaarden $\mu_1 \leq \ldots \leq \mu_n$, $\nu_1 \leq \ldots \leq \nu_n$ en $\kappa_1 \leq \ldots \leq \kappa_n$. Dan geldt voor alle $1 \leq i \leq n$ dat $\nu_i + \kappa_1 \leq \mu_i \leq \nu_i + \kappa_n$*

We laten nu $A = M$, $A' = N$ en diag($\mathbf{k}$) $= K$ als in de stelling. Voor de eigenwaarden μ_i van A geldt dus

$$\nu_i + \kappa_1 \leq \mu_i \leq \nu_i + \kappa_n \tag{3.23}$$
$$\nu_1 + \kappa_1 \leq \mu_i \leq \nu_1 + \kappa_n \tag{3.24}$$

aangezien $\nu_i = \lambda_{km}(A') < -\frac{8}{h^2}$ voor alle km kunnen we de volgende grenzen stellen voor de eigenwaarden μ_{km} van A.

$$-\frac{8}{h^2} + k_{min} \leq \mu_{km} \leq -\frac{8}{h^2} + k_{max} \tag{3.25}$$

In het volgende hoofdstuk zullen we iteratieve methoden behandelen voor het oplossen van het stelsel $Ax = b$. We laten directe methoden buiten beschouwing om de volgende reden: In het tweede hoofdstuk zagen we dat de oplossingen periodiek gedrag vertoonden. Bovendien zagen we een relatie tussen k en het aantal golven op het domein. Om een goede benadering te vinden door middel van onze discretisatie, is het dus noodzakelijk een zeker aantal hoekpunten per golflengte, n_{hp}, te nemen. Op een domein met dimensie d resulteert dit in een matrix van grote N^d, met $N = k \cdot n_{hp}$. De waarde van k kan in de praktijk oplopen tot 500. De resulterende matrix is dan van dusdanige grootte dat directe methoden te traag worden of moeite hebben met de opslag van de matrix.

Hoofdstuk 4

Iteratieve Methoden

Voor het oplossen van grote lineaire stelsels, zoals in ons geval, zijn we dus aangewezen op iteratieve methoden. Onze aanpak zal gebruik maken van de volgende twee methoden; de CG-methode (Conjugate Gradient) en SOR (Successive Over-Relaxation). In de huidige situatie zullen ze beiden echter nog niet gereed zijn voor het oplossen van ons stelsel, convergentie is alleen gegarandeerd bij symmetrische positief-definiete matrices. Daarnaast is de convergentiesnelheid van beide methoden niet toereikend voor het oplossen van ons stelsel. We zullen dit oplossen door een derde methode te introduceren, een combinatie van SSOR en CG.

Voor het numeriek oplossen van stelsels vergelijkingen zijn er natuurlijk veel meer methoden mogelijk. Veel van deze methoden zijn echter niet geschikt voor het oplossen van de Helmholtzvergelijking [2].

4.1 (S)SOR

Voortzettende waar we in het vorige gedeelte gebleven waren, hebben we het linare stelsel $Ax = b$. Om het stelsel geschikt te maken voor iteratieve methoden zullen we het om schrijven naar $f(x_k) = x_{k+1}$. Hiertoe splitsen we de matrix A op, $A = M - N$, zodat we het stelsel als volgt kunnen schrijven; $Mx = Nx + b$. Gebruik makend van $N = M - A$ en vermenigvuldigen met M^{-1} geeft ons de volgende vergelijking: $x_{k+1} = x_k + M^{-1}r_k$, met residu $r_k = b - Ax_k$. We zullen dit residu later gebruiken bij het onderzoeken van de CG methode. Er zijn meerdere mogelijke keuzes voor M^{-1}, maar we zullen ons beperken tot de keuze behorende tot de Gauss-Seidel methode; $A = L + D + U$, met $M = L + D$ en $N = -U$. Hierin is L de benedendriehoeksmatrix is, D de hoofddiagonaal en U de bovendriehoeksmatrix. Dit geeft ons:

$$(D + L)x_{k+1} = b - Ux_k, \quad \text{Gauss-Seidel} \tag{4.1}$$

wat we herschrijven naar:

$$x_{k+1} = D^{-1}\left(b - Lx_{k+1} - Ux_k\right) \tag{4.2}$$

Met een relatief eenvoudige aanpassing kunnen we deze methode verbeteren. Als we aannemen dat de methode convergeert, dan benaderen we de oplossing x per iteratie met $|x_k - x_{k+1}|$ in de richting $x_{k+1} - x_k$. Het is mogelijk dat een andere afstand, in dezelfde richting, de oplossing beter benadert. We introduceren hiervoor een relaxatie parameter $0 < \omega < 2$. En we itereren als volgt: $x_{k+1} = x_k + \omega(x_{k+1} - x_k)_{GS}$. Waarbij

$$(x_{k+1} - x_k)_{GS} = D^{-1}\left(b - Lx_{k+1} - Ux_k - Dx_k\right) \tag{4.3}$$

En verkrijgen de SOR-methode als volgt:

$$x_{k+1} = x_k + \omega D^{-1}\left(b - Lx_{k+1} - Ux_k - Dx_k\right) \tag{4.4}$$
$$Dx_{k+1} = Dx_k + \omega\left(b - Lx_{k+1} - Ux_k - Dx_k\right) \tag{4.5}$$
$$(D + \omega L)x_{k+1} = \left((1-\omega)D - \omega U\right)x_k + \omega b \tag{4.6}$$
$$x_{k+1} = (D + \omega L)^{-1}\left([(1-\omega)D - \omega U]x_k + \omega b\right) \tag{4.7}$$

We kunnen x_{k+1}, gebruik makend van (4.6), berekenen door middel van voorwaartse substitutie

$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \frac{\omega}{a_{ii}}\left(b_i - \sum_{j<i} a_{ij}x_j^{(k+1)} - \sum_{j>i}^{n} a_{ij}x_j^{(k)}\right) \tag{4.8}$$

De zojuist beschreven methode is een gewogen variant van voorwaartse Gauss-Seidel. De achterwaartse variant wordt geven door de methode als volgt aan te passen:

$$x_{k+1} = (D + \omega U)^{-1}\left([(1-\omega)D - \omega L]x_k + \omega b\right) \tag{4.9}$$

oftewel (4.7) met substitutie $U = L$. Met achterwaartse substitutie geeft dit:

$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \frac{\omega}{a_{ii}}\left(b_i - \sum_{j>i} a_{ij}x_j^{(k+1)} - \sum_{j<i}^{n} a_{ij}x_j^{(k)}\right) \tag{4.10}$$

Als de matrix A symmetrisch is, $U = L^T$, kunnen we (4.7) en (4.9) na elkaar uitvoeren;

$$x_{k+1} = (D + \omega U)^{-1}\left([(1-\omega)D - \omega L]x_{k'} + \omega b\right) \tag{4.11}$$
$$x_{k'} = (D + \omega L)^{-1}\left([(1-\omega)D - \omega U]x_k + \omega b\right) \tag{4.12}$$

We kunnen dit als één stap schrijven; $x_{k+1} = A_\omega x_k + B_\omega$

$$A_\omega = (D + \omega L)^{-1}\left[(1-\omega)D - \omega U\right](D + \omega U)^{-1}\left[(1-\omega)D - \omega L\right] \tag{4.13}$$
$$B_\omega = \omega\left(D + \omega U\right)^{-1}\left(I + \left[(1-\omega)D - \omega L\right](D + \omega L)^{-1}\right)b \tag{4.14}$$
$$= \omega\left(2 - \omega\right)(D + \omega U)^{-1}D(D + \omega L)^{-1}b \tag{4.15}$$

Hierbij hebben we in de laatste stap gebruik gemaakt van:

$$[(1-\omega)D - \omega L]\,(D+\omega L)^{-1} = [(2-\omega)D - (D+\omega L)]\,(D+\omega L)^{-1} \quad (4.16)$$

De zojuist beschreven methode is Symmetrische SOR (SSOR). Elke iteratie kost ongeveer twee keer zoveel werk, maar doordat we geen voorkeur geven aan richting waarin we vegen kan het aantal iteraties aanzienlijk verminderen.

Voor dit en het volgende deelhoofdstuk is gebruik gemaakt van [7].

4.1.1 Convergentie van (S)SOR

Een essentiële vraag voor elke methode is of de methode convergeert, en hoe snel. Voor (S)SOR hangt de convergentie onder andere af van de parameter ω. Om de convergentie te bestuderen nemen we aan dat er een oplossing bestaat voor het stelsel $Ax = b$; we noteren deze oplossing als x^*. We kunnen beide methoden, SOR en SSOR, schrijven in de vorm $x^{(k+1)} = Bx^{(k)} + b'$. We hebben dus $x^* = Bx^* + b'$. Voor de fout bij de k-de iteratie noteren we $e^{(k)} = x^{(k)} - x^*$. We kunnen de fout nu als volgt uitdrukken:

$$e^{(k+1)} = x^{(k+1)} - x^* = B(x^{(k)} - x^*) = \ldots = B^{k+1}(x^{(0)} - x^*) \quad (4.17)$$

Het is meteen duidelijk de methode convergeert als $\|B\| < 1$. We kunnen echter een betere conditie voor convergentie geven. Laat v_j de eigenvectoren van de matrix B zijn, met bijbehorende eigenwaarden λ_i. We drukken onze benadering uit in eigenvectoren $x_j = c_j v_j$, $c_j \in \mathbb{R}$ afhankelijk van x. We leiden hieruit af dat

$$e^{(k+1)} = B^{k+1} \sum_{j=1}^{n} c_j v_j = \sum_{j=1}^{n} c_j \lambda_j^{k+1} \quad (4.18)$$

Waarbij c_j zo gekozen zijn dat $(x^{(0)} - x^*) = \sum_j v_j c_j$. Uit (4.18) kunnen we afleiden dat de methode convergeert d.e.s.d.a voor alle eigenwaarden geldt dat $|\lambda_j| < 1$. Met andere worden; de spectrale radius $\rho(A_\omega$ is kleiner dan 1. De spectrale radius van SOR toegepast op de discretisatie matrix is $\rho(A_\omega) > 1$. (S)SOR kan dus niet ons stelsel vergelijkingen oplossen. We kunnen dit oplossen door ons stelsel te vermenigvuldigen met A^T. Hierdoor wordt het stelsel positief-definiet en symmetrisch. We vermelden de volgende stelling:

Theorem 4.1.1 (Ostrowski [5])**.** *Laat A een positief-definiete, symmetrische matrix zijn. Dan is $\rho(A_\omega) < 1$ voor alle $0 < \omega < 2$.*

Hieruit kunnen we afleiden dat SOR wel convergeert voor het aangepaste stelsel $A^T Ax = A^T b$. We zullen in een later hoofdstuk echter zien dat $\rho(AA^T_{omega}) \approx 1$. De convergentiesnelheid zal derhalve laag zijn.

4.2 Conjugate Gradient

De andere methode die we zullen gebruiken in onze aanpak is de conjugate gradient (CG) methode. CG is een niet-stationaire methode, wat inhoud dat de matrix waarop we onze vector projecteren per iteratie veranderd. We zullen de methode introduceren door middel van de kwadratische vorm en de steilste-afdaling methode.

4.2.1 Kwadratische vorm en steilste afdaling

In plaats van het stelsel $Ax = b$ zullen we ons hier richten op de volgende functie:

$$f(x) = x^T Ax - b^T x + c \tag{4.19}$$

Deze functie heeft de mooie eigenschap dat de gradiënt gelijk is aan ons originele stelsel. Oftewel $f'(x) = Ax - b$. De extreme waarden van $f(x)$ vallen dus samen met $f'(x) = 0$. Als de matrix A symmetrisch is en x^* een extremum, dan geldt de volgende gelijkheid:

$$f(x^* + y) = f(x^*) + \frac{1}{2} y^T Ay \tag{4.20}$$

Als de matrix daarnaast ook nog positief-definiet is, dat wil zeggen $x^T Ax \geq 0$, is het extremum een globaal minimum. We kunnen $Ax = b$ dus ook oplossen door het minimum van $f(x)$ te vinden. Bovendien heeft de functie f in dit geval een paraboloïde vorm. Als we dus starten op het punt x_0, hoeven we alleen maar af te dalen om uiteindelijk bij x^* te arriveren. Een manier om dit te doen is door in de richting van het residu, $r_0 = b - Ax_0$, af te dalen. Immers, als het residu kleiner wordt, dan ook de fout. Dan rest ons nog een stapgrootte a te bepalen. We doen dit door a zo te kiezen dat $f(x_{i+1})$ geminimaliseerd wordt. We kunnen het minimum hiervan vinden met behulp van de richtingsafgeleide; $\frac{d}{da_i} f(x_{i+1}) = f'(x_{i+1})^T r_i = 0$. We merken nu op dat $f'(x_{i+1}) = r_{i+1}$, waaruit we afleiden dat dat a_i zo gekozen moet worden dat r_i en r_{i+1} orthogonaal zijn. Dit is het geval voor $a_i = \frac{r_i^T r_i}{r_i^T A r_i}$. Samengevat levert dit de volgende methode op:

$$x_{i+1} = x_i + a_i r_i \tag{4.21}$$

die bekend staat als steilste afdaling. Deze methode convergeert echter niet snel. Een uitgebreide analyse laten we achterwege, maar we melden echter nog het volgende. Een van de oorzaken van de trage convergentie is dat de methode alleen rekening houdt met de vorige iteratie. De informatie uit eerdere iteraties wordt niet gebruikt, waardoor het gebeurt dat de methode meerdere keren in dezelfde richting zoekt. In het volgende gedeelte zullen we dit probleem oplossen.

4.2.2 Afleiding van CG

Om te voorkomen dat we voor de oplossing herhaaldelijk in dezelfde richting zoeken, kunnen we voor onze zoekrichtingen een verzameling orthogonale vectoren gebruiken, $v_1, \ldots, v_n$. We kiezen per iteratie dus een van deze vectoren als zoekrichting, $x_{i+1} = x_i + a_i v_i$. Waarbij we a_i zo kiezen dat $v_i^T e_{i+1} = 0$, oftewel, de fout staat loodrecht op onze zoekrichting. Dit is echter alleen mogelijk als we de fout, en dus de oplossing al kennen.

We kunnen dit oplossen door gebruik te maken van de volgende eigenschap van A. Omdat A symmetrisch en positief definiet is, definieert A een inwendig product; $x^T A y = \langle x, y \rangle_A$. We eisen vervolgens dat onze vectoren v_i en de fout e_i orthogonaal zijn ten opzichte van dit inwendig product, met andere woorden, geconjugeerd. Hierdoor kunnen we de scalair a_i wel vinden:

$$v_i^T A e_{i+1} = 0 \tag{4.22}$$

$$v_i^T A(e_i + a_i v_i) = 0 \tag{4.23}$$

$$a_i = \frac{v_i^T A e_i}{v_i^T A v_i} \tag{4.24}$$

$$a_i = \frac{v_i^T r_i}{v_i^T A v_i} \tag{4.25}$$

Hieruit volgt ook meteen dat het i-de residu r_{i+1} loodrecht staat op de bijbehorende zoekrichting v_i aangezien $v_i^T r_{i+1} = -v_i^T A e_{i+1} = 0$. Daarnaast kunnen we het residu uitdrukken in een lineaire combinatie van het voorgaande residu en zoekrichting:

$$r_{i+1} = -A e_{i+1} \tag{4.26}$$

$$= -A(e_i + a_i v_i) \tag{4.27}$$

$$= r_i - a_i A v_i \tag{4.28}$$

De uitdaging is nu om een verzameling geconjugeerde vectoren te vinden. Door het eerder genoemde feit dat het residu loodrecht staat op de zoekrichtingen en eigenschap (4.28) komen we op het idee om het residu als zoekrichting te kiezen. Voor de eerste iteratie kiezen we $p_0 = r_0$ als zoekrichting, en kiezen a_0 als in (4.25), wat ons brengt bij x_1. Hieruit kunnen we het residu r_1 afleiden. Vervolgens willen we onze nieuwe zoekrichtingen een combinatie laten zijn van de vorige zoekrichting p_i en het nieuwe residu r_{i+1}; $p_{i+1} = r_{i+1} + \beta_{i+1} p_i$. Hierbij willen we β_1 zodanig kiezen dat $p_{i+1}^T A p_j = 0$ voor alle $j \leq i$. Als we gebruik maken van (4.28) kunnen we β_{i+1} afleiden:

$$p_{i+1}^T A p_j = (r_{i+1} + \beta_{i+1} p_j)^T A p_j \tag{4.29}$$

$$\beta_{i+1} = \frac{r_{i+1}^T A p_j}{p_j^T A p_j} \tag{4.30}$$

We kunnen deze uitdrukking verder vereenvoudigen. We doen dit door (4.28) te vermenigvuldigen met r_j:

$$r_j^T r_{i+1} = r_j^T r_i - a_i r_j^T A p_i \tag{4.31}$$

$$a_i r_j^T A p_i = r_j^T r_i - r_j^T r_{i+1} \tag{4.32}$$

$$r_j^T A p_i = \begin{cases} = a_i^{-1} r_i^T r_i, & i = j \\ = -a_{i-1}^{-1} r_i^T r_i, & i+1 = j \\ = 0, & \text{overige gevallen} \end{cases} \tag{4.33}$$

We kunnen (4.30) dus herschrijven naar $\beta_i = \dfrac{r_{i+1}^T r_{i+1}}{r_i^T r_i}$.

De enige taak die ons nog rest is aan te tonen dat de zoekrichtingen p_k onderling conjugeren. We doen dit door middel van inductie. Voor $k = 1$ is dit meteen duidelijk. Stel dat $p_k^T A p_j = 0$ voor alle $j < k$. Dan geldt

$$p_{k+1}^T A p_j = r_{k+1}^T A p_j + \beta_{k+1} p_k^T A p_j = 0 \tag{4.34}$$

als $j = i$ volgt dit uit de definitie van β_{i+1}. Voor $j < i$ volgt dit uit de inductie hypothese.

Als we het voorgaande combineren komen we uit bij de CG methode. Deze zullen we hier als algoritme weergeven:

Algorithm 1 Conjugate Gradient

procedure CG(A, x_0, b)
  $r_0 \leftarrow b - Ax_0$
  $p_0 \leftarrow r_0$
  while $r_i >$ convergentiegrens **do**
    $\alpha_k \leftarrow \dfrac{r_k^T r_k}{p_k^T A p_k}$
    $x_{k+1} \leftarrow x_k + \alpha_k p_k$
    $r_{k+1} \leftarrow r_k - \alpha_k A p_k$
    $\beta_{k+1} \leftarrow \dfrac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$
    $p_{k+1} \leftarrow r_{k+1} + \beta_k p_k$
    $k \leftarrow k + 1$
  end while
  return
end procedure

We merken nog het laatste op over de terminatie van het algoritme. Aangezien we de oplossing x^* nog niet weten, kunnen we niet de fout e_i gebruiken voor de convergentie grens. Het residu is echter wel bekend, en kan dus dienen als stopconditie. We laten het algoritme doorlopen tot dat er nog een fractie van het originele residu over is; $|r_i| \leq \epsilon |r_0|$. Gebruikelijk is om voor ϵ minimaal 10^{-6} te kiezen.

4.2.3 Analyse van CG

Voor de analyse van de zojuist beschreven methoden is het handig om de energie-Norm en de Krylov-deelruimte te introduceren. De Krylov-deelruimte $\mathcal{K}_k(A; y)$ is gedefinieerd als $\mathcal{K}_k(A; y) = \text{span}\{y, Ay, A^2y, \ldots, A^{k-1}y\}$. En de energie norm: $\|x\|_A := (x^T Ax)^{1/2}$.

Voor de CG methode komt $\text{span}\{p_0, \ldots, p_{k-1}\}$ overeen met $\mathcal{K}_k(A; r_0)$ aangezien alle zoekrichting geconjugeerd zijn te opzichte van A en lineaire combinaties zijn van de residuen. De fout e_i ligt dus in de ruimte opgespannen door e_0 en $\mathcal{K}_i(A; r_0)$. We kunnen de fout e_i op de volgende manier uitdrukken:

$$e_i = \left(I + \sum_{k=1}^{i} \gamma_k A^k \right) e_0 \tag{4.35}$$

waarin de coëfficiënten γ_k van α_k en β_k afhangen. We kunnen hier een polynoom $P_i(A)$ uit afleiden, en schrijven $e_i = P_i(A)e_0$.

We drukken de fout e_0 nu uit in A-orthogonale eigenvectoren v_i met lengte 1, $e_0 = \sum_{j=1}^{n} c_j v_j$. Voor elke eigenwaarden λ van A geldt dus dat $P_k(A)v_i = P_k(\lambda)v_i$. We kunnen hier het volgende uit afleiden:

$$e_i = P_i(A)e_0 \tag{4.36}$$

$$e_i = \sum_{j=1}^{n} c_j P_i(\lambda_j) v_j \tag{4.37}$$

$$Ae_i = \sum_{j=1}^{n} c_j P_i(\lambda_j) \lambda_j v_j \tag{4.38}$$

$$\|e_i\|_A^2 = \sum_{j=1}^{n} c_j^2 \left(P_i(\lambda_j)\right)^2 \lambda_j \tag{4.39}$$

$$\|e_i\|_A^2 \leq \min_{P_i} \max_{\lambda_j} P_i(\lambda_j) \sum_{j=1}^{n} c_j^2 \lambda_j \tag{4.40}$$

$$= \min_{P_i} \max_{\lambda_j} P_i(\lambda_j) \|e_0\|_A^2 \tag{4.41}$$

Zoals aangetoond kiest CG de coëfficiënten a_i en β_i zo dat $r_{i+1} = Ae_i$ geminimaliseerd wordt. Dus is $P_i(A)$ de polynoom met die (4.39) minimaliseert. Als de matrix A n verschillende eigenwaarden heeft en P_i de uitdrukking

minimaliseert hebben we de oplossing dus gevonden naar n-iteraties. Dit geldt uiteraard alleen in theorie, in de praktijk moeten we rekening houden met afrondingsfouten. Voor meer informatie over CG verwijzen we naar [8], wat tevens de bron is van dit deelhoofdstuk.

4.2.4 Chebyshev polynomen

We kunnen onze analyse nog verder voortzetten. In plaats van(4.41) op een aantal punten minimaliseren, kunnen we deze uitdrukking ook minimaliseren over het interval $[\lambda_{min}, \lambda_{max}]$. Onze minimalisatie over de eigenwaarden is dus minstens zo goed als deze benadering. Voor deze intervalsminimalisatie wenden we ons tot de Chebyshev polynomen:

$$T_i(\xi) = \frac{1}{2}\left((\xi + (\sqrt{\xi^2+1})^i + (\xi - (\sqrt{\xi^2-1})^i\right) \tag{4.42}$$

Deze polynomen hebben de eigenschap dat $|T_i(\xi)| \leq 1$ voor $\xi \in [-1, 1]$ en dat $|T_i(\xi)| > 1$ voor $\xi \notin [-1, 1]$. Alle nulpunten van T_i vallen dus in het interval $[-1, 1]$. We breiden deze polynoom uit naar het interval $[\lambda_{min}, \lambda_{max}]$.

$$P_i^*(\lambda) = \frac{T_i\left(\frac{\lambda_{max}+\lambda_{min}-2\lambda}{\lambda_{max}-\lambda_{min}}\right)}{T_i\left(\frac{\lambda_{max}+\lambda_{min}}{\lambda_{max}-\lambda_{min}}\right)} \tag{4.43}$$

Deze polynoom oscilleert tussen $\pm T_i\left(\frac{\lambda_{max}+\lambda_{min}}{\lambda_{max}-\lambda_{min}}\right)$ voor $\lambda_{min} \leq \xi \leq \lambda_{max}$. Bovendien is dit de minimale polynoom met $P_i(0) = 1$ op dit interval ten opzichte van het absolute maximum. Het bewijs is vrij eenvoudig:

Stel dat er een andere i-de graads polynoom Q_i is die kleiner is ten opzichte van het maximum op het interval $[\lambda_{min}, \lambda_{max}]$, en die aan dezelfde voorwaarden voldoet als P_i. Dan geldt dat $|Q_i| < |T_i\left(\frac{\lambda_{max}+\lambda_{min}}{\lambda_{max}-\lambda_{min}}\right)^{-1}|$ op het interval. Uit de middelwaardestelling volgt nu dat de polynoom $P_i - Q_i$ i nulpunten op het interval heeft naast het nulpunt $\xi = 0$. Dus $P_i - Q_i$ heeft $i+1$ nulpunten, terwijl het een i-de graads polynoom is. Dit is niet mogelijk, dus bestaat er geen polynoom Q_i die minimaler is op $[\lambda_{min}, \lambda_{max}]$.

4.2.5 Convergentiesnelheid van CG

We kunnen T_i^* dus gebruiken om (4.41) te benaderen. We merken nogmaals op dat $P_i^* \leq T_i \left(\frac{\lambda_{max}+\lambda_{min}}{\lambda_{max}-\lambda_{min}}\right)$ en leiden hier de volgende benadering uit af:

$$\|e_i\|_A \leq T_i \left(\frac{\lambda_{max} + \lambda_{min}}{\lambda_{max} - \lambda_{min}}\right)^{-1} \|e_0\|_A \tag{4.44}$$

$$= T_i \left(\frac{\kappa + 1}{\kappa - 1}\right)^{-1} \|e_0\|_A \tag{4.45}$$

$$= 2 \left[\left(\frac{\sqrt{\kappa} + 1}{\sqrt{\kappa} - 1}\right)^{i} + \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^{i}\right]^{-1} \|e_0\|_A \tag{4.46}$$

$$\leq \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^{i} \|e_0\|_A \tag{4.47}$$

In de tweede stap hebben we gebruik gemaakt van (4.42). De laatste schatting maken we aangezien de tweede term naar nul convergeert voor grote i. κ is het conditiegetal wat voor normale, en dus symmetrische, matrices gedefinieerd is als $\frac{|\lambda_{max}|}{|\lambda_{min}|}$. Het conditiegetal geeft de verhouding tussen kleine wijzigingen in de invoer ten opzichte van het resultaat. Voor een klein conditiegetal hebben wijzigingen in de invoer dus weinig invloed op het resultaat. Matrices met een klein conditiegetal geven dus een snellere convergentie.

4.3 Voorconditionering

We willen nu CG toepassen op onze discretisatie van de Helmholtzvergelijking. Echter, onze discretisatiematrix A is niet positief definiet, aangezien de matrix negatieve eigenwaarden heeft. Daarnaast zijn de eigenwaarden niet dicht bij elkaar gegroepeerd, waardoor de convergentiesnelheid tegen kan vallen. Om dit op te lossen vermenigvuldigen we onze matrix met een ander matrix, met als resultaat een symmetrische, positief definiete matrix met indien mogelijk een lager conditiegetal. We zouden met de getransponeerde matrix A^T kunnen vermenigvuldigen, maar dit willen we voorkomen om de volgende redenen. Als eerste kunnen kleine verstoringen in de vermenigvuldiging groot effect hebben aangezien het conditiegetal groot is. Als tweede is de resulterende matrix AA^T minder schaars, wat het aantal vermenigvuldigingen per iteratie vergroot. Bovenal zijn er andere mogelijkheden die betere werken.

4.3.1 Kaczmarz methode

Voor de voor conditionering maken we gebruik van de Kaczmarz methode. Deze methode lost een lineair stelsel $Ax = b$ op door x op de de rijen van

de matrix A te projecteren:

$$x = x + \frac{b_i - a_i^T x}{\|a_i\|^2} a_i \tag{4.48}$$

Voor elke iteratie doorlopen we alle rijen a_i van de matrix en componenten b_i van de vector b. Er bestaat een relatie tussen deze methode en SOR. We herschrijven (4.8) naar

$$y_i^{(k+1)} = y_i^{(k)} + \frac{b_i - a_i^T A^T y_i}{\|a_i\|^2} \tag{4.49}$$

Door dezelfde paramater $0 < \omega < 2$ in te voeren kunnen we de Kaczmarz methode zien als een SOR iteratie op het stelsel $Ax = b^*$ met $x = A^T y$ en $b^* = A^T b$. De resulterende methode kunnen we op de volgende manier schrijven:

$$x = Q_i x + \frac{\omega a_i}{\|a_i\|^2} b_i \tag{4.50}$$

waarbij $Q_i = (I - \frac{\omega a_i a_i^T}{\|a_i\|^2})$, en $x_{n+1}^{(k)} = x^{(k+1)}$ en b_i de i-de component van de vector b is.

Als conditionering voeren we twee iteraties uit op onze discretisatiematrix. Voor de eerste iteraties vegen we voorwaarts door de rijen, van de eerste tot de n-de kolom. Bij de tweede iteratie vegen we achterwaarste, dus van rij n naar rij 1. We kunnen dit samenvoegen tot één iteratie:

$$x^{(k+1)} = Qx^{(k)} + Rb, \tag{4.51}$$

waarbij $Q = Q_1 Q_2 \dots Q_n Q_n Q_{n-1} \dots Q_1$. De matrix R bevat alle factoren van b_i. De matrices Q_i zijn allemaal symmetrisch, aangezien I en $A_i := \frac{a_i a_i^T}{\|a_i\|^2}$ dit ook zijn. Daarnaast heeft de matrix A_i rang 1, aangezien de rijen worden opgespannen door dezelfde vector, a_i. De enige eigenwaarde λ ongelijk aan nul vinden we uit de eigenvector a_i, dus $\lambda = \omega$. De eigenwaarden van alle Q_i zijn dus $1 - \omega$. We concluderen dat de matrix Q symmetrisch is en dat de eigenwaarden van Q op het interval $[-1, 1]$ liggen.

4.4 CGMN

Samengevat passen we een iteratie van (4.51) toe op onze discretisatie matrix. Vervolgens passen we CG toe op het resulterende stelsel: $(I-Q)x = Rb$. Dit stelsel is wel geschikt voor CG omdat $I - Q$ symmetrisch is. Daarnaast geldt voor alle eigenwaarden λ_i van $I-Q$ dat $\lambda_i = 1+z > 0$ met $z \in (-1, 1)$. Hieruit volgt dat $I - Q$ positief definiet is. Bovendien is de bovengrens voor $|\lambda_{max}|< 2$ een stuk lager dan voor onze discretisatiematrix. De zojuist beschreven methode staat bekend als CGMN

Hoofdstuk 5

Optimalisering van de parameter ω

Tijdens de conditionering van de discretisatiematrix, (4.50), moeten we een keuze maken voor de parameter ω. In dit hoofdstuk zullen met behulp van Fourieranalyse de optimale waarde van ω proberen te vinden.

5.1 Relatie tot SSOR

Zoals opgemerkt in het vorige gedeelte is een Kaczmarz-iteratie op het stelsel $Ax = b$ gelijk aan een SOR iteratie op de normaalvergelijkingen $AA^T y = b$ met $x = A^T y$. De voorconditionering die door ons gekozen is komt dus overeen met een voorwaartse SOR iteratie gevolgd door een achterwaartse SOR iteratie op de normaalvergelijkingen AA^T. Aangezien AA^T symmetrisch is komt dit vervolgens weer overeen met een SSOR iteratie. We splitsen AA^T op in een diagonale, beneden- en bovendriehoeksmatrix, $AA^T = D+L+L^T$. We herhalen hier (4.14) en (4.15):

$$N = (D + \omega L)^{-1} \left[(1-\omega)D - \omega U\right] (D + \omega U)^{-1} \left[(1-\omega)D - \omega L\right] \tag{5.1}$$

$$M = \omega\,(2-\omega)\,(D + \omega U)^{-1} D (D + \omega L)^{-1} \tag{5.2}$$

Een SSOR iteratie wordt zodoende gegeven door $y = Ny + Mb$. Het stelsel waarop we CG gaan toepassen wordt gegeven door $(I - Q)x = Rb$. We kunnen hier het volgende uit afleiden voor de matrices M, N, Q en R.

$$QA^T = A^T N \tag{5.3}$$

$$R = A^T M \tag{5.4}$$

Uit (5.1) en (5.2) kunnen we afleiden dat $N = I - MAA^T$. Hieruit kunnen we een nieuwe uitdrukking voor Q afleiden:

$$Q = I - A^T M A \tag{5.5}$$

Voor de analyse zullen we bovenstaande vergelijking gebruiken.

5.2 Fourieranalyse

We bestuderen onze voorconditionering aan de hand van de matrix die de voortplanting van de fout geeft. Deze matrix kunnen we vinden door een Richardson iteratie uit te voeren op ons stelsel $x^{(k+1)} = x^{(k)}Q + Rb$. De fout wordt hierbij gegeven door $e^{(k+1)} = Qe^{(k)}$ en het residu door $r^{(k+1)} = AQe^{(k)} = G^T r^{(k)}$. Voor de analyse van de fout bekijken we hoe de fout zich voortplant over de matrix. Dit is te vergelijken met het bestuderen van het effect van de voorconditionering op de waarden van de hoekpunten van onze discretisatierooster. We splitsen de fout hiervoor op in Fouriercomponenten, $e_j^k(\theta, \phi) = e^{i(m\theta+n\phi)}$. We kiezen m en n hier zodanig dat het hoekpunt (m, n) overeenkomt met dezelfde hoekpunten als we kozen bij de discretisatie. In andere woorden, als de j-de component overeenkomt met hoekpunt (k, l) kiezen we $m = k$ en $n = l$. We zullen er in het vervolg van uitgaan dat de hoekpunten geordend zijn zoals in deel 3.2 is behandeld.

We willen nu amplitude vinden; $a(\theta, \phi)$, met θ en ϕ zo gekozen zijn dat $a(\theta, \phi)e_j(\theta, \phi) = Qe_j(\theta, \phi)$. We kunnen nu (5.5) en (5.2) gebruiken om $a(\theta, \phi)$ uit te drukken:

$$a = 1 - \omega(2 - \omega)\frac{A_1 A_2}{A_3 A_4} \tag{5.6}$$

Hierin is $A_1 = A^2$, $A_2 = D$, $A_3 = D + \omega L$ en $A_4 = D + \omega U$. We kunnen nu a uitdrukken in termen van $e^{i(\theta+\phi)}$.

5.3 Uitwerking van $a(\theta, \phi)$

5.3.1 Aanpak

Voor de analyse gebruiken we de in hoofdstuk 3 behandelde discretisatie op de 2-dimensionale Helmholtzvergelijking in het eenheidsvierkant:

$$(\nabla^2 + k^2)u = -g, \quad \text{op } [0, 1] \times [0, 1] \tag{5.7}$$

We zullen voor elke matrix A_i alle elementen $a_{ij} \neq 0$ vinden. Hieruit kunnen we voor alle inwendige punten van onze discretisatie een uitdrukking afleiden. Door herhaaldelijk gebruik te maken van de identiteit $e^{ix} + e^{-ix} = 2\cos(x)$ kunnen we de voor elk van deze vectoren een amplitude $a(\theta, \phi)$ vinden zodat $a(\theta, \phi)e^{(k)}(\theta, \phi) = Qe^{(k)}(\theta, \phi)$. Met andere woorden, e^k is een eigenvector met $a_{\theta,\phi}$ als eigenwaarde. We zullen als eerste de matrix A_1 uitwerken, waarbij we onze handelswijze toelichten. Voor de overige matrices zullen we dezelfde aanpak gebruiken.

5.3.2 Uitwerking van A_1, A_2, A_3, A_4

Om de niet-nul elementen van de matrix $A_1 = A^2$ te vinden hebben we de niet-nul elementen van A nodig. We merken nogmaals op dat dit een $n^2 \times n^2$

matrix is met n^2 gelijk aan het aantal roosterpunten. De niet-nul elementen a_{ij} in de i-de rij worden gegeven door:

$$a_{ii} = k^2 - 4h^{-2} \tag{5.8}$$
$$a_{i,i\pm1} = h^{-2} \tag{5.9}$$
$$a_{i,i\pm n} = h^{-2} \tag{5.10}$$

waarbij $h = 1/(n+1)$, de afstand tussen de hoekpunten. We kunnen nu de elementen vinden van A_1 op de i-de rij:

$$a_{ii} = \alpha^2 + 4h^{-4} \tag{5.11}$$
$$a_{i,i\pm1} = 2\alpha h^{-2} \tag{5.12}$$
$$a_{i,i\pm2} = h^{-4} \tag{5.13}$$
$$a_{i,i\pm n} = 2\alpha h^{-2} \tag{5.14}$$
$$a_{i,i\pm(n\pm1)} = 2h^{-4} \tag{5.15}$$
$$a_{i,i\pm2n} = h^{-4} \tag{5.16}$$

hierin is $\alpha = k^2 - 4h^{-2}$. We hebben nu alle informatie die nodig is om A_1 uit te werken:

$$\begin{aligned} A_1 = \alpha^2 + 4h^{-4} + 4\alpha h^{-2}\left[\cos(\theta) + \cos(\phi)\right] \\ + 2h^{-4}\left[\cos(2\theta) + \cos(2\phi) + 4\cos(\theta)\cos(\phi)\right] \end{aligned} \tag{5.17}$$

waar we gebruik hebben gemaakt van het feit dat $e_{j+1} = e_j e^{i\theta}$ en dat $e_{j+n} = e^j e^{i\phi}$. We kunnen dit verder vereenvoudigen:

$$A_1 = [\alpha + 2h^{-2}(\cos\theta + \cos\phi)]^2 \tag{5.18}$$

We kunnen de uitdrukkingen voor de matrices A_2, A_3 en A_4 ook vinden door gebruik te maken van de eerder gevonden a_{ii}'s. Immers, als A^T symmetrisch is geldt: $AA^T = A^2$. We leiden hier de volgende uitdrukkingen uit af:

$$A_2 = \alpha^2 + 4h^{-4} \tag{5.19}$$
$$\begin{aligned} A_3 = \alpha^2 + 4h^{-4} \\ + \omega h^{-2}[2\alpha(e^{-i\theta} + e^{-i\phi}) + 2h^{-2}e^{-i\phi}\cos(\theta) + h^{-2}(e^{-2i\theta} + e^{-2i\phi})] \end{aligned} \tag{5.20}$$
$$\begin{aligned} A_4 = \alpha^2 + 4h^{-4} \\ + \omega h^{-2}[2\alpha(e^{i\theta} + e^{i\phi}) + 2h^{-2}e^{i\phi}\cos(\theta) + h^{-2}(e^{2i\theta} + e^{2i\phi})] \end{aligned} \tag{5.21}$$

De gevonden uitdrukkingen voor A_i substitueren we in (5.6). Dit leidt tot de volgende uitdrukking:

$$a(\theta, \phi, \omega, h) = 1 - \frac{\beta^2[\alpha + 2h^{-2}(\cos\theta + \cos\phi)]^2}{\beta^2 + 2\omega h^{-2}\gamma_1 + 2\omega^2 h^{-4}\gamma_2} \tag{5.22}$$

waarbij $\beta = \alpha + 4h^{-4}$ en

$$\begin{aligned}\gamma_1 &= \beta h^{-2}(\cos 2\theta + \cos 2\phi) + 2\alpha^3(\cos\theta + \cos\phi) \\ &\quad + 4\alpha^2 h^{-2}\cos\theta\cos\phi + 4\alpha h^{-4}(\cos\theta + \cos\phi) + 16h^{-2}\cos\theta\cos\phi \quad (5.23)\\ \gamma_2 &= 2\alpha^2(h^{-2}(\cos\theta + \cos\phi) + 1 + 2\cos(\theta - \phi)) \\ &\quad + 2\alpha h^{-2}(5\cos\theta + \cos\phi + 4\cos(\phi - \theta)\cos\theta + \cos(2\theta - \phi) + \cos(2\phi - \theta)) \\ &\quad + h^{-4}(1 + 8\cos^2\theta + 4\cos(\phi - 2\theta)\cos\theta + \cos(2(\theta - \phi)) + 4\cos\theta\cos\phi) \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (5.24)\end{aligned}$$

Zoals gezegd kunnen we de $a(\theta, \phi, \omega, h)$'s zien als een eigenwaarden van de matrix Q. Het stelsel $(I - Q)$ heeft dus als eigenwaarden $1 - a(\theta, \phi, \omega, h)$. De convergentie van CG hangt af van het conditiegetal; $\kappa = \dfrac{|\lambda_{max}|}{|\lambda_{min}|}$. We kunnen κ benaderen met de volgende vergelijking:

$$\kappa = \frac{\max_{\theta,\phi}(1 - a(\theta, \phi))}{\min_{\theta,\phi}(1 - a(\theta, \phi))} \qquad (5.25)$$

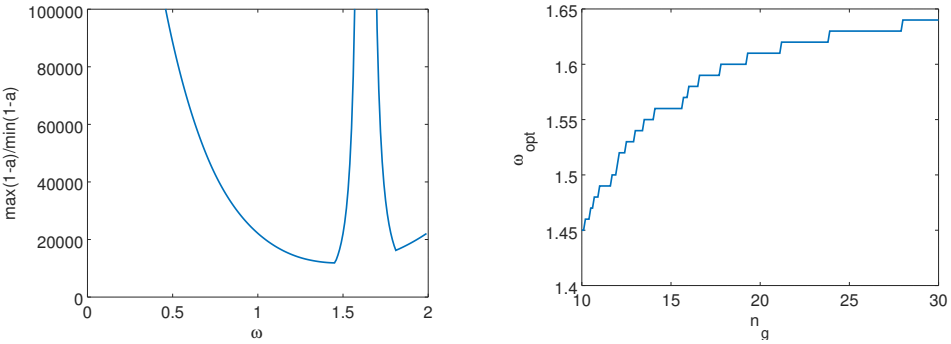
Om κ te minimaliseren, nemen we het minimum van (5.25) over $0 < \omega < 2$. De waarden van h en k zijn al bij de discretisatie bepaald. We hoeven deze dus niet meer mee te nemen in de uitdrukking van a. Bij de discretisatie hebben we een aantal hoekpunten n_{hp} per golflengte genomen. De relatie tussen h, k en n_{hp} wordt gegeven door $hk = \frac{2\pi}{n_{hp}}$. Voor $n_{hp} = 10$ vinden we als optimale waarde $\omega \approx 1.5$. In het volgende deel zullen we een aantal experimenten uitvoeren om deze waarden voor andere n_{hp} te berekenen.

5.4 Numerieke Experimenten

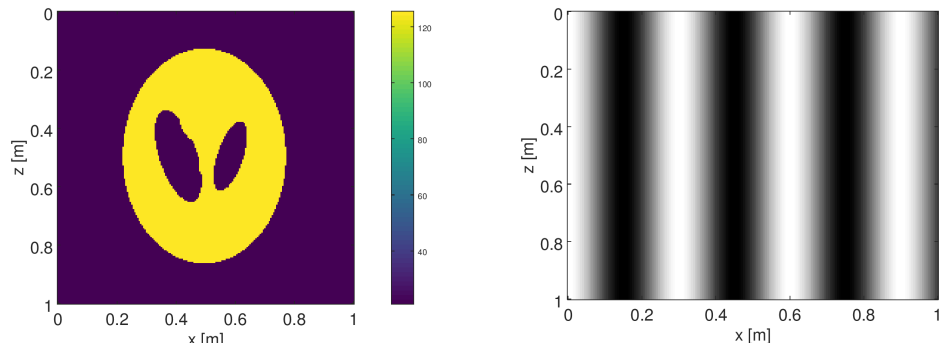
We voeren twee experimenten uit. Als eerste onderzoeken we de optimale waarde van ω_{opt}. Voor het tweede experiment passen we de CGMN methode toe op een model met verschillende waarden voor ω.

5.4.1 Experiment 1

Als eerste plotten we (5.25) met $n_{hp} = 10$ voor verschillende ω. We zien dat de optimale waarde net onder $\omega = 1.5$ ligt. Vervolgens plotten we de optimale waarde van ω voor verschillende n_{hp}. Voor grotere n_{hp} zijn we dus beter af door een grotere ω te kiezen. De grafieken zijn te vinden in Figuur 5.1.



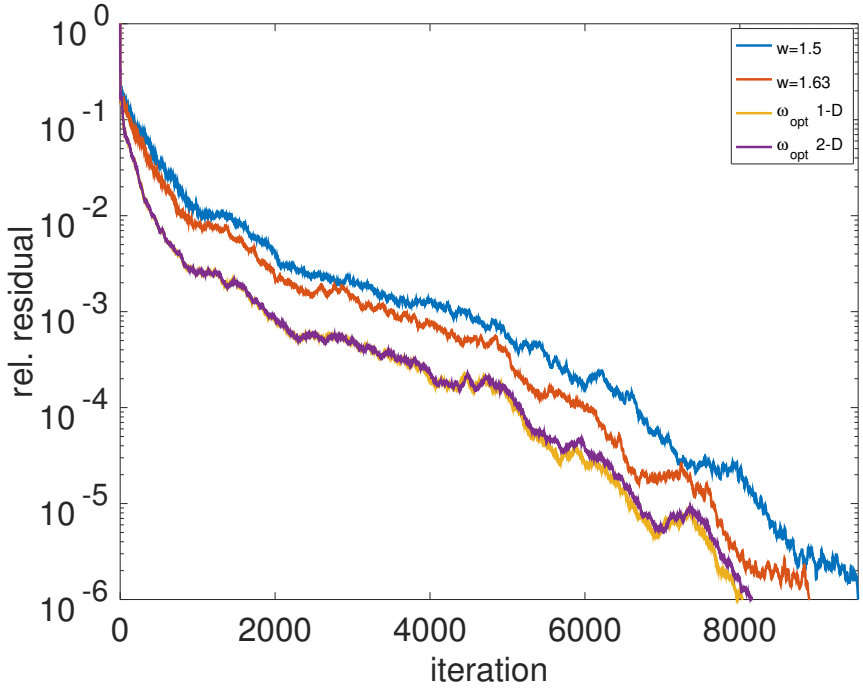
Figuur 5.1: In de linker grafiek plotten we het benaderde conditiegetal voor verschillende ω met $n_{hp} = 10$. In de rechter grafiek plotten we de optimale ω voor verschillende n_{hp}.



Figuur 5.2: Links: Domein van het model. Rechts: Golf die zich over het domein voortplant

5.4.2 Experiment 2

We voeren onze methode uit op hetzelfde model als gebruikt is in [9]. Het domein is heterogeen, en heeft in het midden een hoog golfgetal; zie Figuur 5.2. Vervolgens passen we de CGMN methode werken op dit model voor verschillende waarden van ω; $\omega = 1.5$, $\omega = 1.63$, ω_{opt}2-D uit Figuur 5.1 en ω_{opt}1-D uit [9]. Voor elk van deze waarden hebben we convergentie na minder dan 10000 iteraties. De matrix behorende tot dit systeem is van grootte $O(n^4)$ met $n = 200$. We hebben echter slechts een fractie hiervan nodig om convergentie te behalen. Wat betreft de keuze voor ω zien we dat de variabele keuzes een verbetering zijn ten opzichte van de constante keuzes. Tussen de twee variabele ω's is er nagenoeg geen verschil. Dit doet ons vermoeden dat een 3-D analyse niet tot verdere verbetering leidt.



Figuur 5.3: Aantal iteraties voordat de convergentie van 10^{-6} behaald is voor verschillende ω. ω_{opt}1-D en ω_{opt}2-D zijn nagenoeg even snel.

Hoofdstuk 6

Conclusie

Voor het oplossen van de Helmholtzvergelijking zijn we in de meeste gevallen aangewezen op numerieke methoden. Door bestaande methoden zoals CG te verbeteren is het mogelijk om de vergelijking numeriek op te lossen. Hiertoe discretiseren we het probleem eerst om een stelsel vergelijkingen af te leiden. Dit stelsel kunnen we vervolgens met numerieke methoden oplossen. We hebben hiervoor gekeken naar (S)SOR en CG. Beiden methoden zijn niet in staat om het stelsel op te lossen, of zijn te traag. Door deze methoden slim te combineren kunnen we een nieuwe methode CGMN afleiden, die wel in staat is het stelsel op te lossen. Deze methode kunnen we verder optimaliseren door de parameter ω. Experimenten suggereren dat het beste resultaat wordt behaald door ω af te laten hangen van het lokale golfgetal. Het resultaat is een methode die in staat is het stelsel vergelijkingen op te lossen, en dus de oplossing van de Helmholtzvergelijking te vinden.

Bibliografie

[1] Ascher, U. M., and Greif, C. Chapter 7: Linear Systems: Iterative Methods. In *A First Course in Numerical Methods*. Society for Industrial and Applied Mathematics, Philadelphia, PA, jun 2011, pp. 167–218.

[2] Ernst, O. G., and Gander, M. J. Why it is Difficult to Solve Helmholtz Problems with Classical Iterative Methods. `https://www.unige.ch/{~}gander/Preprints/HelmholtzReview.pdf` op 9-8-2017.

[3] Laub, A. J. *Matrix analysis for scientists and engineers*. Society for Industrial and Applied Mathematics, 2005.

[4] Orn, B., and Zhao, H. Approximate separability of green's function for high frequency helmholtz equations. `https://www.math.uci.edu/{~}zhao/homepage/Publication{\_}files/rank.pdf` op 9-8-2017.

[5] Ostrowski, A. On the linear iteration procedures for symmetric matrices. *Rend. Mat. Appl. 14* (1954), 140–163.

[6] Polyanin, A. D. *Handbook of linear partial differential equations for engineers and scientists*. CRC, 2002.

[7] Saad, Y. *Iterative methods for sparse linear systems*. SIAM, 2003.

[8] Shewchuk, J. R. An Introduction to the Conjugate Gradient Method Without the Agonizing Pain. *Science 49*, CS-94-125 (1994).

[9] Van Leeuwen, T. Fourier analysis of the CGMN method for solving the Helmholtz equation. `https://arxiv.org/pdf/1210.2644v2.pdf` op 9-8-2017.

[10] Weyl, H. Das asymptotische Verteilungsgesetz der Eigenwerte linearer partieller Differentialgleichungen (mit einer Anwendung auf die Theorie der Hohlraumstrahlung). *Mathematische Annalen 71*, 4 (dec 1912), 441–479.