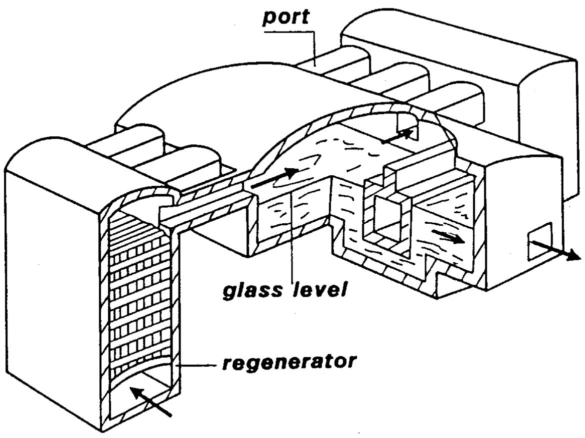
Inhoudsopgave

Hoofdstuk 1

Inleiding

TNO-TPD (Technisch Physische Dienst), gevestigd te Delft, is één van de 15 instituten van de TNO (de Nederlandse organisatie voor toegepast natuurwetenschappelijk onderzoek). Het instutuut heeft naast de uitvoering van de projecten uit het bedrijfsleven ook samenwerking met de TU Delft. De Procesfysica afdeling is gericht op het onderzoek aan processen, waarin stroming, warmteoverdracht en transport van stoffen een rol spellen, vooral met behulp van geavanceerde rekenmodellen. Hierbij zijn de projecten uit de de glas- en keramiekindusrie één van de belangrijkste onderzoeksgebieden. Daarnaast houdt men bezig met het ontwikkelen van thermofysische intrumentatie zoals warmtefluxmeters. Hieronder is een voorbeeld van een glasoven te zien, waarnaar veel onderzoek gedaan wordt binnen de afdeling.

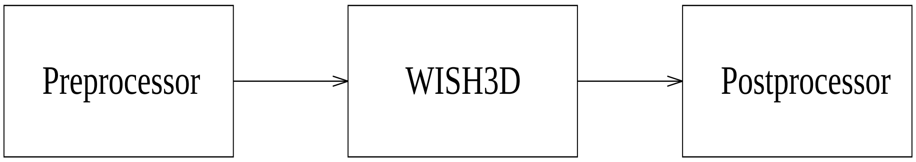


Ten behoeve van het simuleren van de fysische processen, is bij TNO-TPD het CFD (Computational Fluid Dynamics) pakket WISH3D ontwikkeld (daarnaast wordt ook wel het CFD pakket FLUENT gebruikt). WISH3D heeft twee hoofdonderdelen:

1. Het mathematische model voor het beschrijven van fysische problemen. Hierbij kunnen een aantal turbulentie- en verbrandingsmodellen gebruikt worden.

2. Het numerieke model voor het oplossen van de gediscretiseerde vergelijkingen, die verkregen zijn door het discretiseren van de gekoppelde niet-lineaire partiële differentiaalvergelijkingen (PDV) met behulp van de eindige volume methode, met de iteratieve methode SIMPLE (Semi Implicit Method for Pressure-Linked Equations) of met de variant daarop SIMPLER (SIMPLE Revised). Binnen een SIMPLE/SIMPLER iteratie worden een aantal lineaire stelsels opgelost met de WISH3D solvers zoals Line-TDMA, Plane-TDMA en Space-TDMA.

WISH3D is verbonden met een pre- en een postprocessor zoals hieronder geschetst is.



1. De preprocessor bestaat uit de input-files,waarin problemen gedefinieerd kunnen worden. Hiermee wordt WISH3D bestuurd.

2. Met de postprocessor worden problemen gevisualiseerd. Hierbij worden de data verkregen van WISH3D gebruikt.

Het doel van het onderzoek is om uit te zoeken welke van de genoemde WISH3D solvers de convergentie van SIMPLER kunnen versnellen met het oog op de rekentijd. Een andere mogelijkheid is om uit te zoeken in hoeverre de de Krylov solvers een beter resultaat kunnen opleveren.

In het verslag zijn een aantal onderwerpen behandeld.
Ten eerste is een mathematische beschrijving gegeven voor de incompressibele turbulente stromingen met een laag Mach-getal, warmteoverdracht en transport van chemische stoffen. Hierbij is het standaard $(k-\epsilon)$turbulentiemodel en het mixed=burnt verbrandingsmodel kort beschreven. Daarnaast is enige aandacht besteden aan het discretiseren van de vergelijkingen met behulp van de eindige volume methode.
Ten tweede zijn de WISH3D solvers Line-TDMA, Plane-TDMA en Space-TDMA en een aantal Krylov methoden behandeld.
Ten derde zijn de WISH3D solvers en de Krylov solvers getest op een pijpstroming en een glassmeltoven.
Als laatste zijn conclusies en aanbevelingen gegeven voor verder onderzoek.

Hoofdstuk 2

Mathematische beschrijving van turbulente stromingen en de discretisatie van de convectie-diffusie vergelijking

2.1 Inleiding

Dit hoofdstuk bestaat uit de mathematische beschrijving van turbulente stromingen, een korte beschrijving van het zogenaamde mixed=burnt verbrandingsmodel en het discretiseren van de diffusie-convectie vergelijking.

2.2 Mathematische beschrijving van turbulente stromingen

De beschrijving van stromingsproblemen, die met behulp van WISH3D uitgerekend worden, zoals het modelleren van industriële glassmeltovens is gebaseerd op de wetten van behoud van massa, impuls, energie en chemische stoffen, aangevuld met de toestandsvergelijking voor een ideaal gas. De stroming is bij dit soort problemen turbulent. Aangezien dit een wanordelijk en fluctuerend verschijnsel is, is men over het algemeen geïnteresseerd in het gemiddelde van een variabele ϕ over een tijdsinterval [0,T]. Hiertoe is een decompositie methode nodig, die de instantane waarde van ϕ in een gemiddelde plus een fluctuatie splitst. Hier zullen we de Favre decompositie gebruiken. Een andere decompositie is de Reynolds decompositie.
Deze twee decomposities zijn als volgt gedefinieerd:

Reynolds decompositie

Definitie van de Reynolds decompositie luidt:

$$\phi(\underline{x}, t) \;=\; \overline{\phi}(\underline{x}) \;+\; \phi'(\underline{x}, t) \quad , \tag{2.1}$$

$$\overline{\phi}(\underline{x}) \;=\; \lim_{T\to\infty} \frac{1}{T} \int_0^T \phi(\underline{x}, s) ds \quad . \tag{2.2}$$

Hieruit volgt:

$$\overline{\phi'}(\underline{x}, t) = 0 \quad , \tag{2.3}$$

waarbij $\overline{\phi}$ respectievelijk ϕ' het Reynolds gemiddelde en de Reynolds fluctuatie genoemd worden.

Favre decompositie

De Favre decompositie is als volgt gedefinieerd:

$$\phi(\underline{x}, t) \;=\; \tilde{\phi}(\underline{x}) \;+\; \phi''(\underline{x}, t) \quad , \tag{2.4}$$

waarbij het Favre gemiddelde $\tilde{\phi}$ gegeven wordt door de volgende relatie:

$$\tilde{\phi} \;=\; \frac{\overline{\rho\phi}}{\overline{\rho}} \quad , \tag{2.5}$$

hierbij zijn $\overline{\rho\phi}$ en $\overline{\rho}$ Reynolds gemiddelden, waarbij ρ de dichtheid is. Verder voldoet de Favre fluctuatie $\tilde{\phi''}$ aan:

$$\tilde{\phi''}(\underline{x}, t) \;=\; 0 \quad , \tag{2.6}$$

$$\overline{\phi''} \;=\; -\frac{\overline{\rho'\phi'}}{\overline{\rho}} \quad , \tag{2.7}$$

met ρ' de Reynolds fluctuatie van ρ.

2.2.1 Turbulente behoudswetten gebaseerd op de Favre decompositie

In het vervolg van dit verslag zullen we de volgende sommatieconventie gebruiken:
zij $\underline{u} \;=\; (u_1, u_2, u_3)^T$ en $\underline{x} \;=\; (x_1, x_2, x_3)^T$ dan geldt:
$u_i u_i \;=\; \sum_{i=1}^{3} u_i u_i$; $\frac{\partial u_j}{\partial x_j} \;=\; \sum_{j=1}^{3} \frac{\partial u_j}{\partial x_j}$.

Het toepassen van Favre decompositie (2.4) op partiële differentiaalvergelijkingen (PDV), gebaseerd op de behoudswetten, resulteert in de volgende vergelijkingen voor een turbulente stroming (zie [9]):

Behoud van massa

Behoud van massa wordt beschreven door de continuïteitsvergelijking:

$$\frac{\partial \overline{\rho}}{\partial t} + \frac{\partial \overline{\rho}\tilde{u}_j}{\partial x_j} = 0 \quad , \tag{2.8}$$

waarbij u_j de snelheidscomponent in de x_j-richting.

Behoud van impuls

De beschrijving van behoud van impuls in de x_i-richting wordt gegeven door de volgende vergelijking:

$$\frac{\partial \overline{\rho}\tilde{u}_i}{\partial t} + \frac{\partial \overline{\rho}\tilde{u}_j\tilde{u}_i}{\partial x_j} = -\frac{\partial \overline{p}}{\partial x_i} + \frac{\partial \tilde{\tau}_{ij}}{\partial x_j} + \overline{\rho}g_i \quad , \tag{2.9}$$

waarbij $\overline{p}$ de druk voorstelt, g_i de zwaartekrachtversnelling in x_i-richting en $\tilde{\tau}_{ij}$ de spanningstensor, gegeven door:

$$\tilde{\tau}_{ij} = -(\mu + \mu_t)\,(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i}) + \frac{2}{3}(\mu + \mu_t)\,\frac{\partial \tilde{u}_i}{\partial x_j}\delta_{ij} + \frac{2}{3}\overline{\rho}\tilde{k}\delta_{ij} \quad , \tag{2.10}$$

hierbij zijn μ, δ_{ij}, μ_t en k respectievelijk de moleculaire dynamische viscositeit, de kronecker delta, de turbulente vicositeit en de turbulente kinetische energie. In subparagraaf 2.2.2 zullen we μ_t en k evalueren met behulp van een zogenaamd $(k - \epsilon)$turbulentie model.

Behoud van energie

Hier wordt behoud van energie beschreven door de volgende enthalpie vergelijking:

$$\frac{\partial}{\partial t}(\overline{\rho}\tilde{h} - \tilde{p}) + \frac{\partial}{\partial x_j}(\overline{\rho}\tilde{u}_j\tilde{h}) = \frac{\partial}{\partial x_j}((\frac{\lambda}{C_p} + \frac{\mu_t}{\sigma_h})\frac{\partial \tilde{h}}{\partial x_j}) + \tilde{S}_{rad} \quad , \tag{2.11}$$

waarbij λ de thermische conductiviteit is, C_p is de specifieke warmte van een gas (of een mengsel van gassen) bij constante druk en σ_h is het turbulente Prandtl-getal. $\tilde{S}_{rad}$ is de bronterm veroorzaakt door straling.

Behoud van chemische stoffen

Behoud van chemische stoffen wordt door de volgende vergelijking beschreven:

$$\frac{\partial}{\partial t}(\overline{\rho}\tilde{Y}_a) + \frac{\partial}{\partial x_j}(\overline{\rho}\tilde{u}_j\tilde{Y}_a) = \frac{\partial}{\partial x_j}((\overline{\rho}d_a + \frac{\mu_t}{\sigma_a})\frac{\partial \tilde{Y}_a}{\partial x_j}) + \tilde{S}_{a,reac} \quad , \tag{2.12}$$

hierbij is $\tilde{Y}_a$ het Favre gemiddelde voor de massafractie van een chemische stof a. Verder is $d_a$de diffusiecoëfficiënt voor een chemische stof en σ_a is het turbulente Schmidt-getal. De bronterm $\tilde{S}_{a,reac}$ geeft de productie en de destructie weer van een chemische stof a door chemische reacties.

2.2.2 Standaard $(k-\epsilon)$turbulentie model

Het standaard $(k-\epsilon)$turbulentie model legt het verband tussen de turbulente viscositeit μ_t enerzijds en de turbulente kinetische energie $\tilde{k}$ en de dissipatie ervan $\tilde{\epsilon}$ anderzijds:

$$\mu_t \;=\; C_\mu \overline{\rho} \tilde{k}^2 / \tilde{\epsilon} \quad , \tag{2.13}$$

waarbij C_μ een empirische constante is.
De turbulente kinetische energie en de dissipatie ervan voldoen aan de volgende vergelijkingen:

$$\frac{\partial}{\partial t}(\overline{\rho}\tilde{k}) \;+\; \frac{\partial}{\partial x_j}(\overline{\rho}\tilde{u}_j\tilde{k}) \;=\; \frac{\partial}{\partial x_j}((\mu + \frac{\mu_t}{\sigma_k})\frac{\partial \tilde{k}}{\partial x_j}) \;+\; P_k \;-\; \overline{\rho}\tilde{\epsilon} \quad , \tag{2.14}$$

respectievelijk

$$\frac{\partial}{\partial t}(\overline{\rho}\tilde{\epsilon}) \;+\; \frac{\partial}{\partial x_j}(\overline{\rho}\tilde{u}_j\tilde{\epsilon}) \;=\; \frac{\partial}{\partial x_j}((\mu + \frac{\mu_t}{\sigma_\epsilon})\frac{\partial \tilde{\epsilon}}{\partial x_j}) \;+\; \frac{\tilde{\epsilon}}{\tilde{k}}(C_{\epsilon 1}P_k - C_{\epsilon 2}\overline{\rho}\tilde{\epsilon}) \quad . \tag{2.15}$$

De produktie term P_k wordt gegeven door:

$$P_k \;=\; (\; \mu_t(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i}) \;-\; \frac{2}{3}\mu_t \frac{\partial \tilde{u}_i}{\partial x_j}\delta_{ij} \;-\; \frac{2}{3}\overline{\rho}\tilde{k}\delta_{ij} \;)\; \frac{\partial \tilde{u}_i}{\partial x_j} \quad . \tag{2.16}$$

De constanten, die voorkomen in het $(k$-$\epsilon)$model, hebben de volgende waarden:
$C_\mu \;=\; 0.09$, $\sigma_k \;=\; 1.0$, $C_{\epsilon 1} \;=\; 1.44$ en $C_{\epsilon 2} \;=\; 1.92$.

2.2.3 Toestandsvergelijking

De differentiaalvergelijkingen afkomstig van behoudswetten moeten nog aangevuld worden met de toestandsvergelijking voor een ideaal gas. Hiertoe beschouwen we de vloeistof als een mengsel van ideale gassen met massafractie Y_a. Er geldt:

$$p \;=\; \rho R T \sum_a \frac{Y_a}{M_a} \quad , \tag{2.17}$$

waarbij M_a de moleculaire massa van een chemische stof a voorstelt, R is de universele gasconstante en T is de temperetuur.

2.3 Turbulent verbrandingsmodel

Bij industriële gasovens heeft men te maken met verbrandingsreacties tussen verschillende gassen. Om dit te kunnen beschrijven zijn er verbrandingsmodellen ontwikkeld. Het model dat we hier kort zullen bespreken is het mixed=burnt model, ook wel f-model genoemd. Hierbij wordt er uitgegaan van de veronderstelling dat de verbrandigsreacties zeer snel verlopen ten opzichte van de

menging tussen de gassen.
We beperken ons tot de verbrandingsreactie tussen een brandstof (fuel) met massafractie Y_{fu} (bijvoorbeeld aardgas) en een oxidant met massafractie Y_{ox} (bijvoorbeeld zuurstof). Voor deze gassen geldt vergelijking (2.12) voor behoud van chemische stoffen. Onder de aanname dat de diffusiviteit van de gassen dezelfde is en dat 1 kg brandstof gecombineerd met s kg zuurstof in (s+1) kg produkt resulteert, kunnen de thermochemische variabelen (temperatuur T, enthalpie h, dichtheid ρ en de massafracties van de chemische stoffen Y_a) gekoppeld worden aan een scalaire variabele f, de zogeheten mengselfractie. Er geldt $f = 0$ in pure zuurstof en $f = 1$ in pure brandstof terwijl f een tussenliggende waarde heeft voor een mengsel van brand- en zuurstof. Het Favre gemiddelde van de scalar f wordt gedefinieerd als volgt:

$$\tilde{f} \;=\; \frac{\tilde{\xi} - \tilde{\xi}_A}{\tilde{\xi}_F - \tilde{\xi}_A} \quad , \tag{2.18}$$

waarbij de indices A en F de inlaat voor de oxidant respectievelijk de inlaat voor de brandstof weergeven in het geval van een gasoven. Zie Figuur 2.1.

Figuur 2.1: Schets van een glasoven. De inlaat A voor de brandstof en de inlaat F voor de oxidant zijn aangegeven

De scalar $\tilde{\xi}$ wordt gegeven door:

$$\tilde{\xi} \;=\; \tilde{Y}_{fu} \;-\; \tilde{Y}_{ox}/s \quad . \tag{2.19}$$

De mengselfractie $\tilde{f}$ voldoet aan de vogende vergelijking:

$$\frac{\partial}{\partial t}(\overline{\rho}\tilde{f}) \;+\; \frac{\partial}{\partial x_j}(\overline{\rho}\tilde{u}_j\tilde{f}) \;=\; \frac{\partial}{\partial x_j}((\overline{\rho}d_a + \frac{\mu_t}{\sigma_f})\frac{\partial \tilde{f}}{\partial x_j}) \quad . \tag{2.20}$$

Omdat vergelijking (2.20) geen bronterm bevat, wordt f een behouden scalar genoemd.
Als bij een verbrandingsreactie de brandstof en de zuurstof de stoichiometrische verhouding hebben, dat wil zeggen $\tilde{Y}_{fu} = \tilde{Y}_{ox}/s$, dan geldt $\tilde{\xi} = 0$. In dit geval heeft $\tilde{f}$ de stoichiometrische waarde:

$$f_{st} \;=\; -\frac{\tilde{\xi}_A}{\tilde{\xi}_F - \tilde{\xi}_A} \quad . \tag{2.21}$$

Nu kunnen de thermochemische variabelen gekoppeld worden aan de scalar f als volgt:

2.3.1 Massafractie van brandstof

$$Y_{fu} \;=\; \begin{cases} 0 & \text{voor} \quad 0 \le f \le f_{st} \;\;, \\ \\ Y_{fu,F}\;\frac{f-f_{st}}{1-f_{st}} & \text{voor} \quad f_{st} \le f \le 1 \;\;. \end{cases} \tag{2.22}$$

2.3.2 Massafractie van zuurstof

$$Y_{ox} \;=\; \begin{cases} Y_{ox,A}\;(1-\frac{f}{f_{st}}) & \text{voor} \quad 0 \le f \le f_{st} \;\;, \\ \\ 0 & \text{voor} \quad f_{st} \le f \le 1 \;\;. \end{cases} \tag{2.23}$$

Uit (2.22) en (2.23) zien we, bij een verbrandingsreactie, dat als een mengsel rijk is aan de brandstof, $f \geq f_{st}$, dan wordt alle zuurstof en een deel van de brandstof verbruikt terwijl alle brandstof en een deel van de zuurstof verbruikt wordt als het mengsel rijk is aan de zuurstof, $f \leq f_{st}$.

2.3.3 Mengsel-dichtheid

De dichtheid van het gasmengsel is gekoppeld aan de mengselfractie f:

$$\rho(f) \;=\; \frac{p}{RT(f)\sum_a \frac{Y_a(f)}{M_a}} \quad , \tag{2.24}$$

waarbij p de absolute druk is, M_a is de moleculaire massa van een chemische stof a en $T(f)$ is de instantane temperatuur, gegeven door (2.26).

2.3.4 Mengsel-enthalpie

De enthalpie van het gasmengsel is gekoppeld aan de mengselfractie f:

$$h(f) \;=\; \begin{cases} h_a \;+\; (\tilde{h}-h_a)\;\frac{f}{\tilde{f}} & \text{voor} \quad 0 \le f \le \tilde{f} \;\;, \\ \\ \tilde{h} \;+\; (h_F-\tilde{h})\;\frac{\tilde{f}-f}{\tilde{f}-1} & \text{voor} \quad \tilde{f} \le f \le 1 \;\;, \end{cases} \tag{2.25}$$

waarbij $\tilde{h}$ het Favre gemiddelde van de enthalpie is, gegeven door vergelijking (2.11).

2.3.5 Mengsel-temperetuur

De temperatuur van het gasmengsel is gekoppeld aan de mengselfractie f:

$$T(f) \;=\; T_{ref} \;+\; \frac{h(f) - Y_{fu}(f)h_0}{C_{pm,mix}(T,f)} \quad , \tag{2.26}$$

hierbij is T_{ref} een referentie temperatuur, $h(f)$ is de instantane mengsel-enthalpie, h_0 is de calorifische onderwaarde en $C_{pm,mix}(T,f)$ is het gemiddelde van de specifieke warmte van de mengsel over een temperatuurinterval $[T_{ref}, T]$ bij constante druk. Dit laatste wordt gegeven door:

$$C_{pm,mix}(T,f) \;=\; \frac{1}{T(f)-T_{ref}} \int_{T_{ref}}^{T(f)} \sum_a \left(\; Y_a(f) \; C_{p,a}(s) \; \right) ds \quad , \tag{2.27}$$

waarbij $C_{p,a}$ de specifieke warmte van een chemische stof a is bij constante druk.

We kunnen vergelijkingen (2.9), (2.11), (2.12), (2.14), (2.15) en (2.20) in de volgende behoudsvorm schrijven:

$$\frac{\partial}{\partial t}(\; \rho\phi \;) \;+\; \frac{\partial}{\partial x_j}(\; J_j \;) \;=\; S_\phi \quad , \tag{2.28}$$

met

$$J_j \;=\; \rho u_j \phi \;-\; \Gamma_\phi \, \frac{\partial \phi}{\partial x_j} \qquad \text{voor } j \;=\; 1,2,3 \quad , \tag{2.29}$$

waarbij ϕ voor $\tilde{u}_i$, $\tilde{h}$, $\tilde{Y}_a$, $\tilde{k}$, $\tilde{\epsilon}_a$ en $\tilde{f}$ staat en Γ_ϕ is de uitwisselingscoëfficiënt van een variabele ϕ (voor de eenvoud zullen we in het vervolg van het verslag de tildes boven de variabelen weglaten). De term S_ϕ stelt een bronterm voor.

Opmerkingen

1. Bij impulsvergelijking (2.9) is de drukafgeleide $\frac{\partial p}{\partial x_j}$ opgenomen in de bronterm.

2. Er geldt $S_\phi = 0$ voor vergelijking (2.20) van de mengselfractie f.

2.3.6 Begin- en randvoorwaarden

Om de gegeven mathematische beschrijving voor turbulente stromingen compleet te maken, moeten we vergelijkingen van de vorm (2.28) nog met begin- en randvoorwaarden aanvullen.

Beginvoorwaarden

Een beginvoorwaarde houdt in dat op begintijdstip t_0 de waarde van ϕ bekend is in termen van een gegeven functie ϕ_0, waarbij ϕ_0 een functie van de ruimtelijke variabelen kan zijn. Voor de snelheidscomponenten u_i moeten de beginwaardeen aan de continuïteitsvergelijking voldoen.

Randvoorwaarden

De meest voor de hand liggende randvoorwaarden, die opgelegd kunnen worden voor een variabele ϕ, zijn Dirichlet en Neumann randvoorwaarden. Als een rand van het stromingsgebied een vaste wand is, dan gaan de visceuze effecten een belangrijke rol spellen. Het gevolg hievan is dat vergelijkingen van het (k-ϵ)turbulentie model geen goede beschrijving meer geven voor turbulente kinetische energie k en dissipatie ervan ϵ in het wandgebied. Om dit probleem te ondervangen, worden de zogenaamde wandfuncties gebruikt. De hieruit verkregen waarden worden als randvoorwaarden voor k en ϵ opgelegd.

(i) $\underline{Dirichletrandvoorwaarde:}$

Als R een rand van een stromingsgebied voorstelt, dan wordt een Dirichlet randvoorwaarde gegeven door:

$$\phi(\underline{x}, t) = f_{\phi,1}(\underline{x}) \qquad \text{voor} \quad \underline{x} \in R \ , \tag{2.30}$$

hierbij is $f_{\phi,1}$ een gegeven functie. Als $f_{\phi,1} = 0$, dan wordt (2.30) een homogene randvoorwaarde genoemd; in dit geval kan de randvoorwaarde voor de snelheid geschreven worden als volgt:

$$u_{nor} = 0 \ , \tag{2.31}$$

$$u_{tan} = 0 \ , \qquad \text{no-slip voorwaarde} \tag{2.32}$$

waarbij u_{nor} de normaalcomponent en u_{tan} de tangentiële component van de snelheid zijn.

Voorwaarde (2.31) impliceert dat het fluidum niet door de rand kan stromen terwijl (2.32) impliceert dat de rand en de vloeistofdeeltjes er vlakbij dezelfde snelheid hebben. De no-slip voorwaarde geldt alleen voor stromingen met wrijving.

(ii) $\underline{Neumannrandvoorwaarde:}$

Zij n_j de j^{de} component van de naar buiten gerichte eenheidsnormaal van de rand R. Een Neumann randvoorwaarde wordt gegeven door:

$$(\Gamma_\phi \frac{\partial \phi}{\partial x_j})\ n_j(\underline{x}, t) = f_{\phi,2}(\underline{x}) \qquad \text{voor} \quad \underline{x} \in R \ , \tag{2.33}$$

hierbij is $f_{\phi,2}$ ook een gegeven functie. Als $f_{\phi,2} = 0$, dan wordt (2.33) een homogene Neumann randvoorwaarde genoemd. Bij enthalpie (temperatuur) betekent dit dat de rand geïsoleerd is.

(iii) $\underline{Wandfuncties:}$

Om randvoorwaarden voor turbulente kinetische energie k en dissipatie ervan ϵ op te leggen, gebruiken we de zogenaamde wandfuncties. Hiertoe beschouwen we de stroming in het wandgebied waarin de parallelle richting en de normaalrichting aangegeven zijn met x respectievelijk y. Hierbij kan de stroming opgevat worden als een één-dimensionaal Couette-stroming met snelheid U in x-richting, in formule:

$$\frac{\partial}{\partial y}((\mu + \mu_t)\frac{\partial U}{\partial y}) \;=\; 0 \qquad \text{voor } y > 0 \; . \tag{2.34}$$

We definiëren vervolgens de wandschuifspanning τ_w door:

$$\tau_w = ((\mu + \mu_t)\frac{\partial U}{\partial y}) \qquad \text{op} \quad y = 0 \quad . \tag{2.35}$$

Uit (2.34) en (2.35) volgt:

$$\frac{\partial U}{\partial y} \;=\; \frac{\tau_w}{\mu + \mu_t} \quad , \tag{2.36}$$

waarbij μ en μ_t de moleculaire dynamische respectievelijk de turbulente viscositeit zijn.
We definieren een zogenaamde wrijvingssnelheid U^*, een dimensieloze snelheid U^+ en een dimensieloze afstand tot de wand y^+:

$$U^* \;=\; \sqrt{\frac{\tau_w}{\rho}} \quad , \tag{2.37}$$

$$U^+ \;=\; \frac{U}{U^*} \quad , \tag{2.38}$$

$$y^+ \;=\; \frac{\rho y U^*}{\mu} \quad . \tag{2.39}$$

Substitutie van (2.38) en (2.39) in (2.36) resulteert in de volgende dimensieloze vergelijking:

$$\frac{\partial U^+}{\partial y^+} \;=\; \frac{\mu}{\mu + \mu_t} \quad . \tag{2.40}$$

We kunnen μ_t schatten met behulp van de zogenaamde Prandtl-mengweg hypothese:

$$\mu_t \;=\; \rho {l_m}^2 \left|\frac{\partial U}{\partial y}\right| \quad , \tag{2.41}$$

hierbij is l_m de mengweg, gegeven door $l_m = k_v y$ met $k_v = 0.42$ de Von Neumann constante. We kunnen μ_t uitdrukken in termen van de dimensieloze variabelen U^+ en y^+ door (2.38) en (2.39) te gebruiken. Er volgt:

$$\mu_t \;=\; \mu k_v^2 (y^+)^2 \left|\frac{\partial U^+}{\partial y^+}\right| \quad . \tag{2.42}$$

Het wandgebied waarin vergelijking (2.40) geldt, kan opgedeeld worden in twee sublagen, de logaritmische en de visceuze sublaag.

De logaritmische sublaag is het deel van het wandgebied waarin moleculaire dynamische viscositeit μ te verwaarlozen is ten opzichte van turbulente viscositeit μ_t. Dus we krijgen in plaats van (2.40) vergelijking (2.43):

$$\frac{\partial U^+}{\partial y^+} = \frac{\mu}{\mu_t} \quad . \tag{2.43}$$

Substitutie van (2.42) in (2.43) resulteert in de volgende vergelijking:

$$k_v^2 (y^+)^2 \left| \frac{\partial U^+}{\partial y^+} \right| \frac{\partial U^+}{\partial y^+} = 1 \quad , \tag{2.44}$$

met als oplossing:

$$U^+ = \frac{1}{k_v} \, ln(Ey^+) \qquad \text{voor} \quad y^+ \geq 11.5 \quad , \tag{2.45}$$

waarbij $E = 9.8$ een integratie constante is. De waarde van E is experimenteel bepaald.

De visceuze sublaag is het gebied vlakbij de wand waarin turbulente viscositeit μ_t te verwaarlozen is ten opzichte van moleculaire dynamische viscositeit μ. In dit geval wordt (2.40) gereduceerd tot de volgende vergelijking:

$$\frac{\partial U^+}{\partial y^+} = 1 \quad , \tag{2.46}$$

met als oplossing:

$$U^+ = y^+ \qquad \text{voor} \quad y^+ \leq 11.5 \quad , \tag{2.47}$$

hierbij hebben we $U^+ = 0$ voor $y^+ = 0$ als randvoorwaarde gebruikt.

In het wandgebied nemen we aan dat er een lokaal evenwicht tussen de produktie en de dissipatie van turbulente kinetische energie is. Dit betekent dat in vergelijking (2.14) van het $(k - \epsilon)$turbulentie model de convectie-diffusie termen te verwaarlozen zijn ten opzichte van $P_k - \rho\epsilon$. Aangezien de Couette-stroming stationair is ,verdwijnt $\frac{\partial}{\partial t}(\rho k)$ ook uit vergelijking(2.14). Hieruit volgt:

$$P_k - \rho\epsilon = 0 \quad , \tag{2.48}$$

waarbij de produktie term P_k bepaald kan worden door vergelijking (2.16) te beschouwen voor een Couette-stroming:

$$P_k = \mu_t \, (\frac{\partial U}{\partial y})^2. \tag{2.49}$$

Als we nu (2.49) respectievelijk (2.42) in (2.48) substitueren en vervolgens de dimensieloze variabelen U^+ en y^+ gebruiken, dan krijgen we de uitdrukking (2.50) voor ϵ :

$$\epsilon \ = \ (\frac{\rho}{\mu}) \ (k_v y^+)^2 \ (U^*)^4 \ \left|\frac{\partial U^+}{\partial y^+}\right| \ \Big(\frac{\partial U^+}{\partial y^+}\Big)^2 \quad , \tag{2.50}$$

terwijl we voor k de uitdrukking (2.51) verkrijgen door substitutie van (2.42) en (2.50) in (2.13):

$$k \ = \ (c_\mu)^{-1/2} \ (k_v y^+)^2 \ (U^*)^2 \ \Big(\frac{\partial U^+}{\partial y^+}\Big)^2 \quad . \tag{2.51}$$

Samenvattend, in het wandgebied worden randvoorwaarden voor turbulente kinetische energie k en dissipatie ervan ϵ opgelegd door substitutie van snelheidsprofielen (2.45) en (2.47) in (2.50) en (2.51):

$$\begin{cases} \epsilon \ = \ (\frac{\rho}{\mu}) \ (k_v y^+)^{-1} \ (U^*)^4 \quad , \\ \\ k \ = \ (c_\mu)^{-1/2} \ (U^*)^2 \quad . \end{cases} \qquad \text{voor} \quad y^+ \geq 11.5 \quad . \tag{2.52}$$

$$\begin{cases} \epsilon \ = \ (\frac{\rho}{\mu}) \ (k_v y^+)^{2} \ (U^*)^4 \quad , \\ \\ k \ = \ (c_\mu)^{-1/2} \ (k y^+)^2 \ (U^*)^2 \quad . \end{cases} \qquad \text{voor} \quad y^+ \leq 11.5 \quad . \tag{2.53}$$

2.4 Discretisatie van de convectie-diffusie vergelijking

2.4.1 Inleiding

De convectie-diffusie vergelijkingen uit paragraaf 2.2.1 van de vorm:

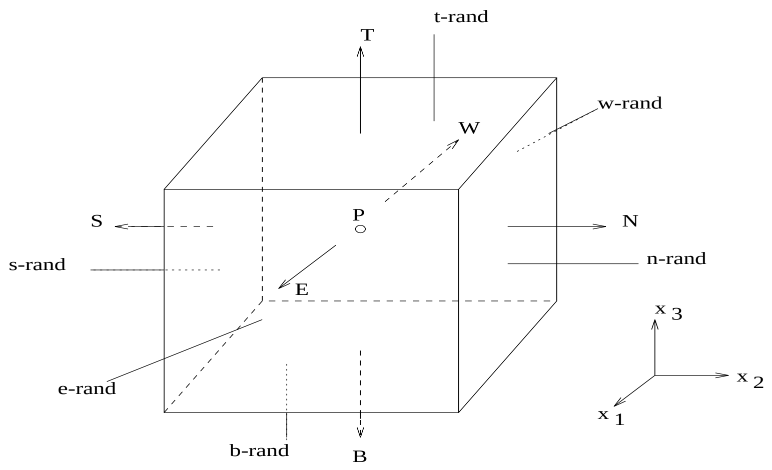
$$\frac{\partial}{\partial t}(\rho\phi) \ + \ \frac{\partial}{\partial x_j}(J_j) \ = \ S_\phi \quad , \tag{2.54}$$

met

$$J_j \ = \ \rho u_j \phi \ - \ \Gamma_\phi \frac{\partial \phi}{\partial x_j} \quad , \qquad j = 1,2,3 \tag{2.55}$$

vormen een stelsel van gekoppelde quasi-lineaire tweede orde partiële differentiaalvergelijkingen. Aangezien dit in het algemeen niet analytisch oplosbaar is, ligt het voor de hand om de oplossing te benaderen met behulp van numerieke oplosmethoden toegepast op de gediscretiseerde vergelijkingen. De bekendste methoden voor het discretiseren van een convectie-diffusie vergelijking zijn de eindige differentie methode (EDM), de eindige elementen methode (EEM) en de eindige volume methode (EVM).

In WISH3D is de eindige volume methode gebruikt om de partiële differentiaalvergelijkingen te discretiseren. Hierbij heeft men de technieken uit het boek van Patankar [1] gebruikt. We zullen hier ook deze methode gebruiken en beperken we ons daarbij tot een korte beschrijving van de methode. Meer details hierover zijn te vinden in [1]. Voor toepassingen van de eindige elementen methode en de eindige differentie methode verwijzen we naar [4] respectievelijk [5].
De manier waarop we hier de eindige volume methode zullen gebruiken is het volgende: we delen het drie-dimensionale rekengebied op in een bepaald aantal aan elkaar grenzende kleine kubussen, de controle-volumes. Zie figuur 2.1 voor een voorbeeld.



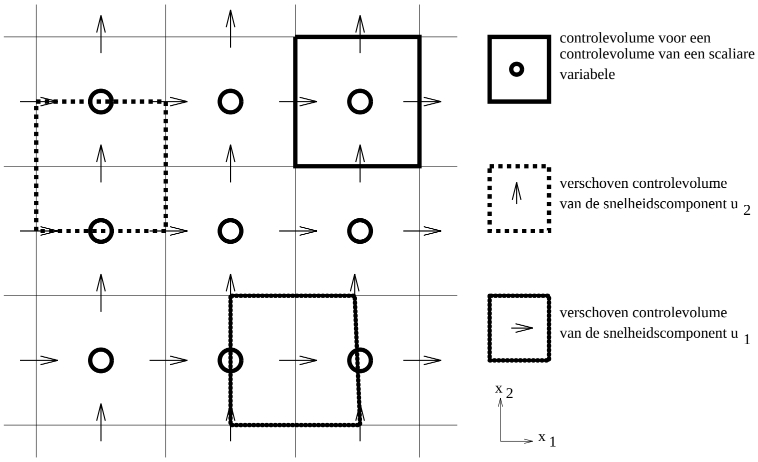
Figuur 2.2: Voorbeeld van een driedimensionale controlevolume

Hierin is P een roosterpunt (gridpunt) terwijl E(ast), W(est), N(orth), S(outh), T(op) en B(ottom) de buurroosterpunten voorstellen. Met e(ast), w(est), n(orth), s(outh), t(op) en b(ottom) worden de randen van de kubus weergegeven. Ieder controlevolume heeft als zwaartepunt een roosterpunt P waarmee een scalaire variabele ϕ geassociëerd is terwijl, gebruik makend van een zogenaamd verschoven rooster (staggered grid), een snelheidscomponent u_i geassociëerd wordt met een punt op een rand in het midden van de verbindingslijn tussen P en een buurroosterpunt zoals aangegeven is in figuur 3.2 .

2.4.2 Tijdsdiscretisatie met behulp van impliciete schema

Voor tijdsdiscretisatie zullen we het volledig impliciete schema gebruiken.
We beginnen eerst met het integreren van vergelijking (2.54) over een tijdsinterval [t , $t + \Delta t$] :

$$\int_t^{t+\Delta t} \frac{\partial}{\partial \tau}(\rho\phi)\ d\tau \ + \int_t^{t+\Delta t} \frac{\partial}{\partial x_j}(J_j)\ d\tau \ = \int_t^{t+\Delta t} S_\phi\ d\tau \quad . \tag{2.56}$$



Figuur 2.3: tweedimensionaal verschoven rooster

We beschouwen de integranden in de tweede en derde integraal als constanten over het interval [t , $t + \Delta t$]. Voor een impliciet schema nemen we de integranden gelijk aan de waarde op het nieuwe tijdstip en we krijgen de volgende uitdrukking:

$$(\; [\rho\phi]_P \; - \; [\rho\phi]_{P,0} \;) \; - \; \frac{\partial}{\partial x_j}(J_j) \; \Delta t \; = \; S_\phi \; \Delta t \quad , \tag{2.57}$$

hierbij is $[\rho\phi]_P$ de evaluatie van $\rho\phi$ op een roosterpunt P en $[\rho\phi]_{P,0}$ is de waarde van $[\rho\phi]_P$ op het oude tijdstip.

2.4.3 Discretisatie met behulp van de eindige volume methode

Nu kunnen we vergelijking (2.57) discretiseren door dit te integreren over een controlevolume V met volume ΔV :

$$\int_V (\; [\rho\phi]_P \; - \; [\rho\phi]_{P,0} \;) \; dV \; + \; \int_V (\; \frac{\partial}{\partial x_j}(J_j) \; \Delta t \;) \; dV \; = \; \int_V (\; S_\phi \; \Delta t \;) \; dV \quad . \tag{2.58}$$

We zullen iedere integraal van (2.58) kort uitwerken:

Het integreren van de bronterm

De bronterm S_ϕ kan geschreven worden als een lineaire functie van de variabele ϕ (zie blz. 48 en 49 van [1]):

$$S_\phi \; = \; S_C \; + \; S_P \phi_P \; , \tag{2.59}$$

waarbij S_C en S_P constanten zijn. Onder de aanname dat S_ϕ constant is over het controlevolume V, krijgen we het volgende:

$$\int_V (S_\phi \, \Delta t) \, dV \; = \; (S_C \; + \; S_P \phi_P) \, \Delta t \, \Delta V \quad . \tag{2.60}$$

Het integreren van de instationaire term

We nemen aan dat de integrand constant is over het het controlevolume V zodat we de integraal als volgt kunnen benaderen:

$$\int_V ([\rho\phi]_P \; - \; [\rho\phi]_{P,0}) \, dV \; = \; ([\rho\phi]_P \; - \; [\rho\phi]_{P,0}) \, \Delta V \; . \tag{2.61}$$

Het integreren van de convectie-diffusie term

De randen van V geven we aan met $R_1 = e$, $R_2 = w$, $R_3 = n$, $R_4 = s$, $R_5 = t$ en $R_6 = b$.
Met behulp van de divergentietheorema van Gauss kunnen we de volume-intgraal schrijven in termen van de som van de integralen over de randen $R_i, i = 1, ..., 6$, van het controlevolume V:

$$\int_V (\frac{\partial}{\partial x_j}(J_j)) \, dV \; = \; \sum_{i=1}^{6} \int_{A_{R_i}} (J_j \; n_j) \, dA_{R_i} \quad , \tag{2.62}$$

waarbij, gegeven een rand R_i, A_{R_i} de oppervlakte voorstelt en n_j de j^{de} component van de naar buiten gerichte eenheidsnormaal. Als we aannemen dat $(J_j \; n_j)$ constant is over de rand R_i , dan volgt:

$$\int_V (\frac{\partial}{\partial x_j}(J_j)) \, dV \; = \; \sum_{i=1}^{6} (J_j \; n_j)_{R_i} \, A_{R_i} \quad , \tag{2.63}$$

hierbij is $(J_j \; n_j)_{R_i} A_{R_i}$ de flux door de rand R_i. Als we deze met J_{R_i} noteren, dan schrijven we (2.63) als volgt:

$$\begin{aligned} \int_V (\frac{\partial}{\partial x_j}(J_j)) \, dV \; &= \; (\, J_e \; - \; J_w \,) \; + \; (\, J_n \; - \; J_s \,) \; + \\ &\quad + \; (\, J_t \; - \; J_b \,) \quad , \end{aligned} \tag{2.64}$$

hierbij wordt bijvoorbeeld de flux door rand e(ast) gegeven door:

$$J_e \; = \; (\, \rho u_1 A \,)_e \, \phi_e \; - \; (\, \Gamma_\phi A \,)_e \, (\, \frac{\partial\phi}{\partial x_1} \,)_e \quad . \tag{2.65}$$

De diffusie term $(\frac{\partial\phi}{\partial x_1})_e$ discretiseren we door het gebruik maken van centrale differenties terwijl de convectieve term ϕ_e gediscretiseerd wordt met een upwind of hybride schema. Het hybride schema (veel gebruikt in WISH3D) toegepast op J_e zegt het volgende:

$$\begin{aligned} J_e &= (\rho u_1 A)_e \; \phi_E \qquad (P_e)_e \leq -2 \; , \\ J_e &= (\rho u_1 A)_e \, \frac{\phi_P \; + \; \phi_E}{2} \; - \; (\Gamma_\phi A)_e \, \frac{\phi_E \; - \; \phi_P}{(\delta x_1)_e} \qquad -2 < (P_e)_e < 2 \; , \\ J_e &= (\rho u_1 A)_e \; \phi_P \qquad (P_e)_e \geq 2 \; , \end{aligned} \tag{2.66}$$

waarbij $(\delta x_1)_e$ de afstand voorstelt tussen het roosterpunt P en het buurroosterpunt E, zie figuur 2.2 . $(P_e)_e$ is het maas Peclet getal, gegeven door:

$$(P_e)_e \; = \; \frac{(\rho u_1 \delta x_1)_e}{(\Gamma_\phi)_e} \quad . \tag{2.67}$$

Voor evaluatie van de coefficiënten $(\rho u_1 A)_e$ en $(\Gamma_\phi A)_e$ verwijzen we naar [1].
We kunnen uitdrukking (2.66) eenvoudiger maken door invoering van de volgende notaties:

$$F \; = \; \rho u_1 A \;\; , \tag{2.68}$$

$$D \; = \; \frac{\Gamma_\phi A}{\delta x_1} \;\; , \tag{2.69}$$

$$a_E \; = \; D \; max\{ \, -(Pe)_e \; , \; 1 - \frac{1}{2}(Pe)_e \; , \; 0\} \quad . \tag{2.70}$$

Hiermee kunnen we (2.66) als volgt schrijven:

$$J_e \; = \; -a_E \phi_E \; + \; (a_E + (F)_e)\phi_P \quad . \tag{2.71}$$

Op analoge wijze kunnen we uitdrukkingen voor de fluxen door andere randen verkrijgen:

$$J_w \; = \; -a_W \phi_W \; + \; (a_W - (F)_w)\phi_P \;\; , \tag{2.72}$$

$$J_n \; = \; -a_N \phi_N \; + \; (a_N + (F)_n)\phi_P \;\; , \tag{2.73}$$

$$J_s \; = \; -a_S \phi_S \; + \; (a_S - (F)_s)\phi_P \;\; , \tag{2.74}$$

$$J_t \; = \; -a_T \phi_T \; + \; (a_T + (F)_t)\phi_P \;\; , \tag{2.75}$$

$$J_b \; = \; -a_B \phi_B \; + \; (a_B - (F)_b)\phi_P \;\; , \tag{2.76}$$

waarbij

$$a_W \; = \; max\{ \, (Pe)_w \; , \; 1 + \frac{1}{2}(Pe)_w \; , \; 0\} \;\; , \tag{2.77}$$

$$a_N \; = \; max\{ \, -(Pe)_n \; , \; 1 - \frac{1}{2}(Pe)_n \; , \; 0\} \;\; , \tag{2.78}$$

$$a_S \; = \; max\{ \, (Pe)_s \; , \; 1 + \frac{1}{2}(Pe)_s \; , \; 0\} \;\; , \tag{2.79}$$

$$a_T \; = \; max\{ \, -(Pe)_t \; , \; 1 - \frac{1}{2}(Pe)_t \; , \; 0\} \;\; , \tag{2.80}$$

$$a_B \; = \; max\{ \, (Pe)_b \; , \; 1 + \frac{1}{2}(Pe)_b \; , \; 0\} \quad . \tag{2.81}$$

Substitutie van deze uitdrukkingen in (2.58) resulteert uiteindelijk in de gediscretiseerde convectie-diffusie vergelijking:

$$-a_B\phi_B - a_S\phi_S - a_W\phi_W + a_P\phi_P + a_E\phi_E + a_N\phi_N + a_T\phi_T \;=\; b \;, \tag{2.82}$$

waarbij

$$\begin{aligned} a_P \;&=\; a_E \;+\; a_W \;+\; a_N \;+\; a_S \;+\; a_T \;+\; a_B \;+ \\ &\quad +\; \rho_P\,\frac{\Delta V}{\Delta t} \;-\; S_P\,\Delta V \;\;, && (2.83)\\ b \;&=\; S_C\,\Delta V \;+\; [\rho\phi]_{P,0}\,\frac{\Delta V}{\Delta t} \;\;. && (2.84)\end{aligned}$$

Bij het afleiden van uitdrukking (2.83) hebben we aangenomen dat $(F)_e = (F)_w$, $(F)_n = (F)_s$ en $(F)_t = (F)_b$.

2.4.4 Matrix-vektor notatie

De verkregen lineaire vergelijkingen kunnen in matrix-vektor notatie geschreven worden. Hiertoe beschouwen we eerst (2.82) voor ieder controle-volume van het rekengebied:

$$\begin{aligned} &-\,(a_B)_{i,j,k}\;\phi_{i,j,k-1} \;-\; (a_S)_{i,j,k}\;\phi_{i,j-1,k} \;-\; (a_W)_{i,j,k}\;\phi_{i-1,j,k} \;+ \\ &\qquad\qquad\qquad\qquad +\;(a_p)_{i,j,k}\;\phi_{i,j,k} \;+ \\ &+\,(a_E)_{i,j,k}\;\phi_{i+1,j,k} \;+\; (a_N)_{i,j,k}\;\phi_{i,j+1,k} \;+\; (a_T)_{i,j,k}\;\phi_{i,j,k+1} \;= \\ &\qquad\qquad\qquad\qquad =\; b_{i,j,k} \;\;, \qquad (2.85) \\ &\text{voor} \quad 1 \le i \le NI \;,\; 1 \le j \le NJ \;,\; 1 \le k \le NK \;\;, \end{aligned}$$

hierbij correspondeert de index i,j,k met roosterpunt P, $i-1,j,k$ met buurroosterpunt W, $i+1,j,k$ met buurroosterpunt E enzovoorts. De index i,j,k geeft de vergelijking weer die met het i^{de} roosterpunt op de j^{de} roosterlijn in het k^{de} roosteroppervlak correspondeert. Hierbij is een roosterlijn een verbindingslijn van roosterpunten in x_1- richting. Een roosteroppervlak is een oppervlak waarin roosterlijnen liggen.
Als we randvoorwaarden in vergelijking (2.85) verwerken zodanig dat de bekende waarden van ϕ in het rechter lid opgenomen worden, dan kunnen we deze vergelijking schrijven als een stelsel lineaire vergelijkingen van de vorm:

$$\boldsymbol{Ax} \;=\; \boldsymbol{b} \;\;, \tag{2.86}$$

waarbij x en b vektoren voorstellen met elementen $\phi_{i,j,k}$ respectievelijk $b_{i,j,k}$. Vektor b noemen we een bronvektor. De matrix $A \in \mathbf{R}^{n\times n}$, met $n =$

$NI \times NJ \times NK$, heeft de volgende vorm:

$$
\boldsymbol{A} = \begin{pmatrix}
\boldsymbol{A_1} & \boldsymbol{T_1} & & & & & \\
\boldsymbol{B_2} & \boldsymbol{A_2} & \boldsymbol{T_2} & & & & \\
 & \boldsymbol{B_3} & \boldsymbol{A_3} & \boldsymbol{T_3} & & & \\
 & & \cdot & \cdot & \cdot & & \\
 & & & \cdot & \cdot & \cdot & \\
 & & & & \cdot & \cdot & \cdot \\
 & & & & \boldsymbol{B_{NK-1}} & \boldsymbol{A_{NK-1}} & \boldsymbol{T_{NK-1}} \\
 & & & & & \boldsymbol{B_{NK}} & \boldsymbol{A_{NK}}
\end{pmatrix} .
$$

T_k , k=1,2,...,NK-1, is een diagonale matrix met elementen $-(a_T)_{i,j,k}$ voor $1 \leq i \leq NI$ en $1 \leq j \leq NJ$.
B_k , k=2,3,...,NK, is een diagonale matrix met elementen $-(a_B)_{i,j,k}$ voor $1 \leq i \leq NI$ en $1 \leq j \leq NJ$.
A_k , k=1,2,...,NK, is een blok-tridiagonale matrix, gegeven door:

$$
\boldsymbol{A_k} = \begin{pmatrix}
\boldsymbol{A_{1,k}} & \boldsymbol{N_{1,k}} & & & & & \\
\boldsymbol{S_{2,k}} & \boldsymbol{A_{2,k}} & \boldsymbol{N_{2,k}} & & & & \\
 & \boldsymbol{S_{3,k}} & \boldsymbol{A_{3,k}} & \boldsymbol{N_{3,k}} & & & \\
 & & \cdot & \cdot & \cdot & & \\
 & & & \cdot & \cdot & \cdot & \\
 & & & & \cdot & \cdot & \cdot \\
 & & & & \boldsymbol{S_{NJ-1,k}} & \boldsymbol{A_{NJ-1,k}} & \boldsymbol{N_{NJ-1,k}} \\
 & & & & & \boldsymbol{S_{NJ,k}} & \boldsymbol{A_{NJ,k}}
\end{pmatrix} .
$$

$N_{j,k}$, j=1,2,...,NJ-1 , is een diagonale matrix met diagonale elementen $(a_N)_{i,j,k}$.
$S_{j,k}$, j=2,3,...,NJ , is een diagonale matrix met diagonale elementen $(a_S)_{i,j,k}$.
$A_{j,k}$, j=1,2,...,NJ , is een tridiagonale matrix van de vorm:

$$
\boldsymbol{A_{j,k}} = \begin{pmatrix}
\boldsymbol{(a_P)_{1,j,k}} & \boldsymbol{(a_E)_{1,j,k}} & & & & & \\
\boldsymbol{(a_W)_{2,j,k}} & \boldsymbol{(a_P)_{2,j,k}} & \boldsymbol{(a_E)_{2,j,k}} & & & & \\
 & \cdot & \cdot & \cdot & & & \\
 & & \cdot & \cdot & \cdot & & \\
 & & & \cdot & \cdot & \cdot & \\
 & & & & \boldsymbol{(a_W)_{NI-1,j,k}} & \boldsymbol{(a_P)_{NI-1,j,k}} & \boldsymbol{(a_E)_{NI-1,j,k}} \\
 & & & & & \boldsymbol{(a_W)_{NI,j,k}} & \boldsymbol{(a_P)_{NI,j,k}}
\end{pmatrix} .
$$

In het volgende hoofdstuk gaan we oplostechnieken voor $Ax = b$ beschouwen.

Hoofdstuk 3

Numerieke oplosmethoden voor de lineaire stelsels die voorkomen bij WISH3D

3.1 Inleiding

In dit hoofdstuk zullen we vier onderwerpen bespreken.
Ten eerste een korte beschrijving van de belangrijkste bestaande WISH3D solvers, te weten Line-TDMA (Tri-Diagonal-Matrix Algorithm), Plane-TDMA en Space-TDMA. Dit zijn iteratieve methoden voor het oplossen van stelsels lineaire vergelijkingen van de vorm:

$$\boldsymbol{Ax} = \boldsymbol{b} \quad , \tag{3.1}$$

waarbij matrix A beschreven is het hoofdstuk 2. Een gedetailleerde beschrijving van Line-TDMA is te vinden in [1]. Voor een korte behandeling van Plane-TDMA en Space-TDMA verwijzen we naar [2]. WISH3D bevat ook een aantal bekende klassieke iteratieve methoden zoals Gauss-Jacobi, Gauss-Seidel en S.O.R. (Successive Over Relaxation). Voor een overzicht hierover verwijzen we naar [8, hoofdstuk 4].
Ten tweede geven we een beknopt overzicht van een aantal Krylov deelruimte methoden, namelijk CG (Conjugate Gradients), Bi-CGSTAB (Bi-CG STA-Bilized), GCR (Generalized Conjugate Residuals) en GMRES (Generalized Minimal RESiduals). (zie ook [8,hoofdstuk 5])
Ten derde geven we een beschrijving van een aantal preconditioners voor de hierboven genoemde Krylov methoden. Als preconditioners hebben we Diagonaal schaling, Line-TDMA, Plane-TDMA, Space-TDMA, ILU- (Incomplete LU-decompositie), MILU- (Modified ILU), en RILU-decompositie (Relaxed ILU) gekozen.
Als laatste zullen we de zogenaamde SIMPLE methode beschrijven aan de hand van de distributieve iteratie zoals deze behandeld is in [10].

3.2 WISH3D solvers

Het idee achter de methoden besproken in deze subparagraaf is het volgende: kies een geschikte niet-singuliere matrix P zodanig dat, bij een gegeven beginschatting x^0, het stationaire iteratieve proces:

$$\begin{aligned} x^0 &= \text{beginschatting} \ , \\ Px^m &= (P-A)x^{m-1} + b \ , \end{aligned} \tag{3.2}$$

tot een goede benadering x^M leidt voor de oplossing x ten koste van weinig rekenwerk. Bij het gegeven iteratieve proces stelt M het aantal iteraties voor.

Afhankelijk van de keuze van de matrix P hebben we te maken met één van de genoemde WISH3D solvers. Als bijvoorbeeld $P = \text{diag}(A)$, dan spreken we van het Gauss-Jacobi proces terwijl als $P = D - L$, dan spreken we van het Gauss-Seidel proces. Hierbij is de matrix $-L$ het strikte benedendriehoeksgedeelte van A. In de rest van de subparagraaf zullen we aandacht besteden aan Line-, Plane- en Space-TDMA. Hierbij worden de matrices, die beschreven staan in hoofdstuk 2 (subparagraaf 2.3), gebruikt.

3.2.1 Line-TDMA

Deze methode is gebaseerd op het idee om een Gauss-Seidel achtige iteratieve methode te combineren met een directe methode, TDMA. TDMA is een Gauss-eliminatie methode voor het oplossen van stelsels lineaire vergelijkingen van de vorm:

$$\boldsymbol{Cx} = \boldsymbol{b} \ , \tag{3.3}$$

waarbij C een tri-diagonale matrix is. Er wordt dus een LU-decompositie gemaakt voor matrix C zodat (3.3) als volgt opgelost kan worden:

$$Ly = b \quad \text{voorwaartse eliminatie,} \tag{3.4}$$

$$Ux = y \quad \text{terugsubstitutie.} \tag{3.5}$$

We merken op dat Gauss-eliminatie weing rekenwerk kost voor een tridiagonaal stelsel. Om Line-TDMA te beschrijven als een iteratief proces splitsen we de matrix A uit (3.1) als volgt:

$$\boldsymbol{A} = \boldsymbol{G} - \boldsymbol{L} - \boldsymbol{U} \ , \tag{3.6}$$

en vervolgens kiezen we $P = G - L$. Hieruit kunnen we de gelijkenis zien tussen deze splitsing en die van Gauss-Seidel, immers G speelt hier de rol van de diagonaal matrix D in de Gauss-Seidel iteratieve methode. Matrices G, L en U worden hierna beschreven:

1. $\boldsymbol{G = diag(G_1, ..., G_{NK})}$ is een blok-diagonale matrix met $\boldsymbol{G_k = diag(A_{1,k}, ..., A_{NJ,k})}$ waarbij matrices $\boldsymbol{A_{j,k}}$, j=1,...,NJ, beschreven zijn in hoofdstuk twee (subparagraaf 2.3.4).

2. $\boldsymbol{L}$ is een strikte blok-benedendriehoeksmatrix:

$$\boldsymbol{L} = \begin{pmatrix} \boldsymbol{L_1} & & & & \\ \boldsymbol{B_2} & \boldsymbol{L_2} & & & \\ & \cdot & \cdot & & \\ & & \cdot & \cdot & \\ & & & \cdot & \cdot \\ & & & \boldsymbol{B_{NK}} & \boldsymbol{L_{NK}} \end{pmatrix},$$

waarbij

- $\boldsymbol{B_k = diag(B_{2,k}, ..., B_{NJ,k})}$ met $\boldsymbol{B_{j,k} = diag((a_B)_{1,j,k}, ..., (a_B)_{NI,j,k})}$ voor j=2,...,NJ .
- $\boldsymbol{L_k}$, k=1,...,NK, zijn matrices van de volgende vorm:

$$\boldsymbol{L_k} = \begin{pmatrix} \mathbf{0} & & & & \\ \boldsymbol{S_{2,k}} & \mathbf{0} & & & \\ & \cdot & \cdot & & \\ & & \cdot & \cdot & \\ & & & \cdot & \cdot \\ & & & \boldsymbol{S_{NJ,k}} & \mathbf{0} \end{pmatrix},$$

hierbij is $\mathbf{0}$ een nul-matrix en $\boldsymbol{S_{j,k} = diag((a_S)_{1,j,k}, ..., (a_S)_{NI,j,k})}$ voor j=2,...,NJ .

3. $\boldsymbol{U}$ is een strikte blok-bovendriehoeksmatrix:

$$\boldsymbol{U} = \begin{pmatrix} \boldsymbol{U_1} & \boldsymbol{T_1} & & & \\ & \cdot & \cdot & & \\ & & \cdot & \cdot & \\ & & & \cdot & \cdot \\ & & & \boldsymbol{U_{NK-1}} & \boldsymbol{T_{NK-1}} \\ & & & & \boldsymbol{U_{NK}} \end{pmatrix},$$

waarbij:

- $\boldsymbol{T_k = diag(T_{1,k}, ..., T_{NJ-1,k})}$ met $\boldsymbol{T_{j,k} = diag((a_T)_{1,j,k}, ..., (a_T)_{NI,j,k})}$ voor j=1,...,NJ-1 .
- $\boldsymbol{U_k}$, k=1,...,NK, zijn matrices van de volgende vorm:

$$\boldsymbol{U_k} = \begin{pmatrix} \mathbf{0} & \boldsymbol{N_{1,k}} & & & \\ & \cdot & \cdot & & \\ & & \cdot & \cdot & \\ & & & \cdot & \cdot \\ & & & \mathbf{0} & \boldsymbol{N_{NJ-1,k}} \\ & & & & \mathbf{0} \end{pmatrix},$$

met $\boldsymbol{N_{j,k} = diag((a_N)_{1,j,k}, ..., (a_N)_{NI,j,k})}$ voor j=1,...,NJ-1.

We kunnen de vektor $\boldsymbol{x}$ van het stelsel $\boldsymbol{Ax = b}$ als volgt weergeven:

$$\boldsymbol{x} = \begin{pmatrix} \boldsymbol{x_1} \\ \vdots \\ \boldsymbol{x_{NK}} \end{pmatrix} \qquad \text{met} \qquad \boldsymbol{x_k} = \begin{pmatrix} \boldsymbol{x_{1,k}} \\ \vdots \\ \boldsymbol{x_{NJ,k}} \end{pmatrix} ,$$

waarbij $\boldsymbol{x_{j,k}}$ vektoren zijn met als elementen de onbekende waarden van een variabele $\boldsymbol{\phi}$ (zie hoofdstuk 2) op de roosterpunten van een gridlijn evenwijdig aan de x-as. Op dezelfde manier kunnen we de vektor $\boldsymbol{b}$ ook weergeven. We kunnen Line-TDMA dus opvatten als een blok Gauss-Seidel methode, waarbij in dit geval de diagonaal blokken overeenkomen met de roosterpunten op een gridlijn evenwijdig aan de x-as. We merken op dat ook Line-TDMA varianten gegeven kunnen worden voor gridlijnen evenwijdig aan de y of z-as. We spreken dan van de volgende Line-TDMA varianten:

1. Linezyx : De diagonaal blokken komen overeen met de roosterpunten op een gridlijn evenwijdig aan de x-as, waarbij de gridlijn in een oppervlak evenwijdig aan de y-as ligt.

2. Lineyzx : De diagonaal blokken komen overeen met de roosterpunten op een gridlijn evenwijdig aan de x-as, waarbij de gridlijn in een oppervlak evenwijdig aan de z-as ligt.

3. Linezxy : De diagonaal blokken komen overeen met de roosterpunten op een gridlijn evenwijdig aan de y-as, waarbij de gridlijn in een oppervlak evenwijdig aan de x-as ligt.

4. Linexzy : De diagonaal blokken komen overeen met de roosterpunten op een gridlijn evenwijdig aan de y-as, waarbij de gridlijn in een oppervlak evenwijdig aan de z-as ligt.

5. Lineyxz : De diagonaal blokken komen overeen met de roosterpunten op een gridlijn evenwijdig aan de z-as, waarbij de gridlijn in een oppervlak evenwijdig aan de x-as ligt.

6. Linexyz : De diagonaal blokken komen overeen met de roosterpunten op een gridlijn evenwijdig aan de z-as, waarbij de gridlijn in een oppervlak evenwijdig aan de y-as ligt.

We merken op dat er gerelaxeerd kan worden bij iedere iteratie van een Line-TDMA solver zoals aangegeven is bij het iteratieve proces van Linezyx. Daarbij is $\boldsymbol{\omega} \in [\boldsymbol{0}, \boldsymbol{1}]$ de relaxatie faktor.
Nu kunnen we eenvoudig nagaan dat de Linezyx solver beschreven kan worden door het volgende iteratieve proces:

Algoritme 3.1 (Linezyx)

$\boldsymbol{x}^0 =$ beginschatting ,
$\boldsymbol{B_{j,0} = 0}$
$\boldsymbol{T_{j,NK+1} = 0}$
$\boldsymbol{S_{0,k} = 0}$
$\boldsymbol{N_{NJ+1,0} = 0}$
for $\boldsymbol{1 \leq m \leq M}$ **do**
  for $\boldsymbol{k = 1, ..., NK}$ **do**
    for $\boldsymbol{j = 1, ..., NJ}$ **do**
      $\boldsymbol{A_{j,k} x^m_{j,k} = B_{j,k} x^m_{j,k-1} + S_{j,k} x^m_{j-1,k} + N_{j,k} x^{m-1}_{j+1,k} + T_{j,k} x^{m-1}_{j,k+1} + b_{j,k}}$
    end for
  end for
  $\boldsymbol{x^m_{j,k} = \omega x^m_{j,k} + (1-\omega) x^{m-1}_{j,k}}$
end for

waarbij het tri-diagonale stelsel van dit iteratieve proces exact opgelost wordt met behulp van (3.4) en (3.5). In WISH3D is het mogelijk om de binnenste en de buitenste lus van het algoritme een aantal keer te laten herhalen. Elke herhaling wordt een sweep genoemd. In het geval van Algoritme 3.1 spreken we van een sweep in de z-richting en een sweep in de y-richting. Een sweep in een tegengestelde richting is ook mogelijk. Zie Toepassing 2 . Voor de iteratieve processen van de andere Line-TDMA varianten verwijzen we naar de appendix A.

Beweringen

Voor alle beweringen geldt $\boldsymbol{\omega = 1}$.

1. Als $\boldsymbol{T_{j,k} = 0}$ en $\boldsymbol{N_{j,k} = 0}$ voor alle $\boldsymbol{1 \leq j \leq NJ}$ en $\boldsymbol{1 \leq k \leq NK}$ dan zijn Linezyx en Lineyzx exacte solvers.

2. Als $\boldsymbol{T_{i,k} = 0}$ en $\boldsymbol{E_{i,k} = 0}$ voor alle $\boldsymbol{1 \leq i \leq NI}$ en $\boldsymbol{1 \leq k \leq NK}$ dan zijn Linezxy en Linexzy exacte solvers.

3. Als $\boldsymbol{N_{i,j} = 0}$ en $\boldsymbol{E_{i,j} = 0}$ voor alle $\boldsymbol{1 \leq i \leq NI}$ en $\boldsymbol{1 \leq j \leq NJ}$ dan zijn Lineyxz en Linexyz exacte solvers.

We zullen alleen de bewering voor Linezyx bewijzen. De andere beweringen kunnen op dezelfde wijze bewezen worden.

Bewijs

We merken eerst op dat als Linezyx een exacte solver is dan moet gelden $\boldsymbol{x^1_{j,k} = x_{j,k}}$ Als $\boldsymbol{T_{j,k} = 0}$ en $\boldsymbol{N_{j,k} = 0}$ dan geldt:

$$\boldsymbol{A = G - L} \quad . \tag{3.7}$$

Uit het algoritme volgt voor k=1 en j=1 dat:

$$\boldsymbol{A_{1,1} x^1_{1,1} = b_{1,1}} \quad . \tag{3.8}$$

Uit (3.8) en de eerste bewering volgt dat $\boldsymbol{x^1_{1,1} = x_{1,1}}$ de exacte oplossing.
Voor k=1 en j=2 geldt:

$$\boldsymbol{A_{2,1} x^1_{2,1} = S_{2,1} x^1_{1,1} + b_{2,1}} \quad . \tag{3.9}$$

Omdat $\boldsymbol{x^1_{1,1} = x_{1,1}}$ volgt uit (3.8) en (3.9) dat $\boldsymbol{x^1_{2,1} = x_{2,1}}$ de exacte oplossing.
Analoog volgt $\boldsymbol{x^1_{j,1} = x_{j,1}}$ voor $\boldsymbol{3 \leq j \leq NJ}$.
Beschouw nu k=2 en j=1, dan geldt:

$$\boldsymbol{A_{1,2} x^1_{1,2} = B_{1,2} x^1_{1,1} + b_{1,2}} \quad , \tag{3.10}$$

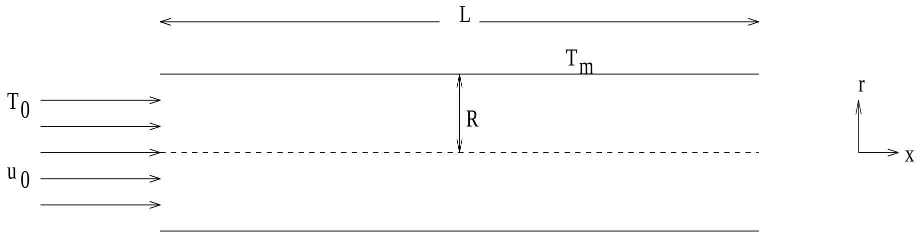
dus $\boldsymbol{x^1_{1,2} = x_{1,2}}$.
Als laatste kiezen we k=2 en j=1:

$$\boldsymbol{A_{2,2} x^1_{2,2} = B_{2,2} x^1_{2,1} + S_{2,2} x^1_{1,2} + b_{2,2}} \quad , \tag{3.11}$$

hieruit volgt $\boldsymbol{x^1_{2,2} = x_{2,2}}$.
Voor $\boldsymbol{3 \leq k \leq NK}$ en $\boldsymbol{3 \leq j \leq NJ}$ gaat het bewijs analoog.

Toepassing 1

In het hoofdstuk 4 zullen we een voorbeeld van een pijpstroming bekijken, waarin blijkt dat Linezxy een exacte solver is. Om dit te kunnen motiveren, beschouwen we een twee-dimensionale turbulente stroming, waarbij de geometrie geschetst is in Figuur 3.1 .

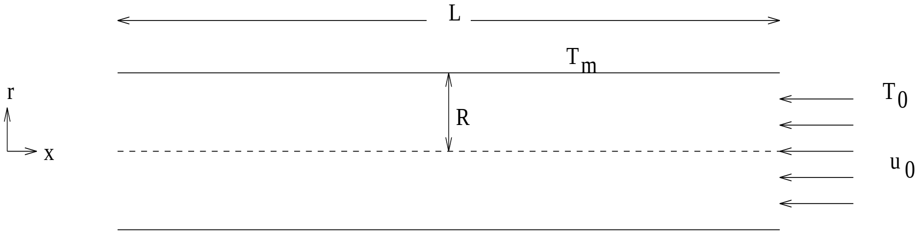


Figuur 3.1: Twee-dimensionale geometrie van een pijpstroming

De stroming is convectie dominant in de richting van de x-as, dit houdt in dat voor het maas-Peclet getal geldt $\boldsymbol{P_e > 2}$. Bij het het discretiseren van de convectieve termen met het hybride schema volgt dan voor de elementen van de matrices $\boldsymbol{E_{i,k}}$ (i=1,...,NI en k=1,...,NK) dat $\boldsymbol{(a_E)_{i,j,k} = Dmax\{-P_e, 1 - 1/2P_e, 0\} = 0}$. Er geldt dus $\boldsymbol{E_{i,k} = 0}$.
Aangezien de stroming twee-dimensionaal is, geldt er ook $\boldsymbol{T_{i,k} = 0}$.
Op basis van de tweede bewering kunnen we Linezxy als exacte solver voor het oplossen van de lineaire stelsels, die verkregen zijn door het discretiseren van de convectie-diffusie vergelijking uit het hoofdstuk 2.

Toepassing 2

We beschouwen dezelfde stroming als bij toepassing 1, behalve dat we nu de situatie hebben zoals geschetst in Figuur 3.2.



Figuur 3.2: Twee-dimensionale geoemetrie van een pijpstroming, waarbij de snelheid de tegengestelde x-richting heeft

De stroming is convectie dominant, maar nu in de tegengestelde richting van de x-as. Dit houdt in dat $\boldsymbol{P_e < -2}$. In dit geval geldt $\boldsymbol{W_{i,k} = 0}$ in plaats van $\boldsymbol{E_{i,k} = 0}$. Immers, de elementen van de matrices $\boldsymbol{W_{i,k}}$ worden gegeven door $\boldsymbol{(a_W)_{i,j,k} = Dmax\{P_e, 1 + 1/2P_e, 0\} = 0}$, terwijl voor de elementen van $\boldsymbol{E_{i,k}}$ geldt $\boldsymbol{(a_E)_{i,j,k} = Dmax\{P_e, 1 + 1/2P_e, 0\} \neq 0}$. Dus de Linezxy solver wordt gegeven door het volgende algoritme:

Algoritme 3.2 (Linezxy)

$\boldsymbol{x^0}$ = beginschatting ,
for $\boldsymbol{1 \leq m \leq M}$ **do**
  for $\boldsymbol{k = 1, ..., NK}$ **do**
    for $\boldsymbol{i = 1, ..., NI}$ **do**
      $\boldsymbol{A_{i,k}x^m_{i,k} = B_{i,k}x^m_{i,k-1} + E_{i,k}x^{m-1}_{i+1,k} + T_{i,k}x^{m-1}_{i,k+1} + b_{i,k}}$
    end for
  end for
end for

Uit Algoritme 3.2 kunnen we zien dat wegens de aanwezigheid van $\boldsymbol{E_{i,k}x^{m-1}_{i+1,k}}$ in het algoritme, kan Linezxy geen exacte solver zijn.
Om er voor te zorgen dat Linezxy een exacte solver wordt, kiezen we in plaats van de splitsing $\boldsymbol{A = P - U}$ met $\boldsymbol{P = G - L}$ de volgende splitsing van de matrix $\boldsymbol{A}$:

$$\boldsymbol{A = P - L} \quad , \tag{3.12}$$

waarbij $\boldsymbol{P = G - U}$.
Analoog aan de eerste splitsing leiden we het volgende algoritme af voor de Linezxy solver:

Algoritme 3.3 (Linezyx)

$\boldsymbol{x^0}$ = beginschatting ,

for $\boldsymbol{1 \leq m \leq M}$ **do**
  for $\boldsymbol{k = NK, ..., 1}$ **do**
    for $\boldsymbol{i = NI, ..., 1}$ **do**
      $\boldsymbol{A_{i,k}x_{i,k}^{m} = T_{i,k}x_{i,k+1}^{m} + E_{i,k}x_{i+1,k}^{m} + W_{i,k}x_{i-1,k}^{m-1} + B_{i,k}x_{i,k-1}^{m-1} + b_{i,k}}$
    end for
  end for
end for

Hierbij wordt één sweep in de tegengestelde z-richting en één sweep in de tegengestelde x-richting genomen.
Nu beweren we het volgende:
Als $\boldsymbol{W_{i,k} = 0}$ en $\boldsymbol{T_{i,k} = 0}$, dan is Linezxy een exacte solver.
Het bewijs hiervan is analoog aan het voorgaande bewijs.

Opmerking

In WISH3D kan de combinatie van beide toepassingen gebruikt worden. Dit houdt in dat men de mogelijkheid heeft om bijvoorbeeld sweeps kiezen zowel in de z-richting als in de tegengestelde z-richting.

3.2.2 Plane-TDMA

Plane-TDMA heeft drie varianten: Planex, Planey en Planez. In deze subparagraaf zullen we Planez beschrijven. De beschrijving van de twee andere varianten is analoog.
We splitsen de matrix $\boldsymbol{A}$ als volgt:

$$\boldsymbol{A = G - B - T} \quad . \tag{3.13}$$

en kiezen we vervolgens $\boldsymbol{P = G - B}$. De matrices $\boldsymbol{P}$, $\boldsymbol{B}$ en $\boldsymbol{T}$ worden beschreven als volgt:

1. $\boldsymbol{G = diag(A_1, ..., A_{NK})}$ is een blok tri-diagonale matrix waarbij de matrices $\boldsymbol{A_k}$, k=1,...,NK, gegeven zijn in hoofdstuk twee (subparagraaf 2.3.4) .

2. Matrix B :

$$\boldsymbol{B} = \begin{pmatrix} \mathbf{0} & & & & \\ \boldsymbol{B_2} & \mathbf{0} & & & \\ & \cdot & \cdot & & \\ & & \cdot & \cdot & \\ & & & \cdot & \cdot \\ & & & \boldsymbol{B_{NK}} & \mathbf{0} \end{pmatrix} .$$

3. Matrix T :

$$\boldsymbol{T} = \begin{pmatrix} \mathbf{0} & \boldsymbol{T_1} & & & & \\ & \cdot & \cdot & & & \\ & & \cdot & \cdot & & \\ & & & \cdot & \cdot & \\ & & & & \mathbf{0} & \boldsymbol{T_{NK-1}} \\ & & & & & \mathbf{0} \end{pmatrix} .$$

Nu kunnen we Plane-TDMA als een iteratief proces beschrijven:

$\boldsymbol{x^0}$ = beginschatting ,
for $\boldsymbol{1 \leq m \leq M}$ **do**
  for $\boldsymbol{k = 1, ..., NK}$ **do**
    $\boldsymbol{A_k\ x_k^m \ =\ B_k\ x_{k-1}^m\ +\ T_k\ x_{k+1}^{m-1}\ +\ b_k}$
  end for
end for

Hierbij kunnen we, analoog aan de Line-TDMA solvers, ook sweeps in de z-richting kiezen.
Hieruit zien we dat Plane-TDMA een vorm van blok Gauss-Seidel is, waarbij de blokken $\boldsymbol{A_k}$ overeen komen met het $\boldsymbol{k^{de}}$ vlak van het rekengebied. Het blok-tridiagonale stelsel $\boldsymbol{A_k\ x_k^m = b_k^m}$ wordt benaderend opgelost met een iteratieve methode, die gebaseerd is op de standaard incomplete LU-decompositie (ILU-decompositie) in het geval van een twee-dimensionaal probleem. De reden om hier niet de Gauss-decompositie $\boldsymbol{A_k = L_k U_k}$ te nemen is dat $\boldsymbol{L_k}$ en $\boldsymbol{U_k}$ veel minder ijl (sparse) zijn dan de matrix $\boldsymbol{A_k}$ (de band loopt vol). Dit kost veel rekenwerk en geheugen. De ILU-decompositie komt neer op het splitsen van $\boldsymbol{A_k}$ als volgt: $\boldsymbol{A_k = L_k D_k^{-1} U_k - R_k}$ en vervolgens het oplossen van het volgende blok tri-diagonaal stelsel:

$$\boldsymbol{L_k D_k^{-1} U_k x_k^m = R_k x_k^{m-1} + B_k x_{k-1}^m + T_k x_{k+1}^{m-1} + b_k.} \tag{3.14}$$

De matrices $\boldsymbol{L_k}$, $\boldsymbol{D_k^{-1}}$, $\boldsymbol{U_k}$ en $\boldsymbol{R_k}$ zijn als volgt gedefinieerd:

1. $\boldsymbol{L_k}$ bestaat uit het strikte benedendriehoeksgedeelte van $\boldsymbol{A_k}$ plus de diagonale matrix $\boldsymbol{D_k}$.

2. $\boldsymbol{U_k}$ bestaat uit het strikte bovendriehoeksgedeelte van $\boldsymbol{A_k}$ plus de diagonale matrix $\boldsymbol{D_k}$.

3. $\boldsymbol{D_k}$ is een diagonale matrix met elementen $\boldsymbol{d_{i,j,k}}$, die verkregen worden door $\boldsymbol{diag(L_k D_k^{-1} U_k) = diag(A)}$ te stellen. Hieruit volgt:

$$\begin{aligned} \boldsymbol{d_{i,j,k}} &= \boldsymbol{(a_P)_{i,j,k} - \frac{(a_W)_{i,j,k}\ (a_E)_{i-1,j,k}}{d_{i-1,j,k}}} + \\ & \quad - \boldsymbol{\frac{(a_S)_{i,j,k}\ (a_N)_{i,j-1,k}}{d_{i,j-1,k}}} , \end{aligned} \tag{3.15}$$

met $\boldsymbol{d_{1,1,1} = (a_P)_{1,1,1}}$, $\boldsymbol{(a_E)_{0,j,k} = 0}$ en $\boldsymbol{(a_N)_{i,0,k} = 0}$.

4. $\boldsymbol{R_k} = \boldsymbol{L_k D_k^{-1} U_k} - \boldsymbol{A_k}$. De matrix $\boldsymbol{R_k}$ heeft de volgende vorm:

$$\boldsymbol{R_k} = \begin{pmatrix} \mathbf{0} & \boldsymbol{R^N_{1,k}} & & & & \\ \boldsymbol{R^S_{2,k}} & \mathbf{0} & \boldsymbol{R^N_{2,k}} & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & \cdot \\ & & & \boldsymbol{R^S_{NJ-1,k}} & \mathbf{0} & \boldsymbol{R^N_{NJ-1,k}} \\ & & & & \boldsymbol{R^S_{NJ,k}} & \mathbf{0} \end{pmatrix},$$

waarbij

- matrices $\boldsymbol{R^S_{j,k}}$, j=2,...,NJ, gegeven worden door:

$$\boldsymbol{R^S_{j,k}} = \begin{pmatrix} \mathbf{0} & \boldsymbol{r^S_{1,j,k}} & & & \\ & \cdot & \cdot & & \\ & & \cdot & \cdot & \\ & & & \cdot & \cdot \\ & & & \mathbf{0} & \boldsymbol{r^S_{NI-1,j,k}} \\ & & & & \mathbf{0} \end{pmatrix},$$

  met $\boldsymbol{r^S_{i,j,k}} = \frac{\boldsymbol{(a_S)_{i,j,k}\ (a_E)_{i,j-1,k}}}{\boldsymbol{d_{i,j-1,k}}}$.

- en matrices $\boldsymbol{R^N_{j,k}}$, j=1,...,NJ-1, gegeven worden door:

$$\boldsymbol{R^N_{j,k}} = \begin{pmatrix} \mathbf{0} & & & & \\ \boldsymbol{r^N_{2,j,k}} & \mathbf{0} & & & \\ & \cdot & \cdot & & \\ & & \cdot & \cdot & \\ & & & \cdot & \cdot \\ & & & \boldsymbol{r^N_{NI,j,k}} & \mathbf{0} \end{pmatrix},$$

  met $\boldsymbol{r^N_{i,j,k}} = \frac{\boldsymbol{(a_W)_{i,j,k}\ (a_N)_{i-1,j,k}}}{\boldsymbol{d_{i-1,j,k}}}$.

We merken op dat door $\boldsymbol{L_k D_k^{-1} U_k} = \boldsymbol{A_k}$ te nemen, we de volgende situatie krijgen:

1. De matrix $\boldsymbol{L_k}$ bestaat afgezien van het strikte benedendriehoeks gedeelte van $\boldsymbol{A_k}$ plus de diagonale matrix $\boldsymbol{D_k}$ nog uit het benedendriehoeks gedeelte van $\boldsymbol{R_k}$.

2. De matrix $\boldsymbol{U_k}$ bestaat afgezien van het strikte bovendriehoeks gedeelte van $\boldsymbol{A_k}$ plus de diagonale matrix $\boldsymbol{D_k}$ nog uit het bovendriehoeks gedeelte van $\boldsymbol{R_k}$.

Dus de matrices $\boldsymbol{L_k}$ en $\boldsymbol{U_k}$ hebben nog extra elementen gekregen in de plaatsen waarin de nul-elementen horen. Dit wordt de fill-in genoemd. De matrix $\boldsymbol{R_k}$ bestaat dus uit de fill-in elementen.
Samenvattend, we kunnen Planez beschrijven met behulp van Algoritme 3.4, waarbij de benadering $\boldsymbol{x_k^m}$ gerelaxeerd wordt:

Algoritme 3.4 (Planez)

$\boldsymbol{x^0 = 0}$,
for $\boldsymbol{1 \leq m \leq M}$ **do**
  for $\boldsymbol{k = 1, ..., NK}$ **do**
    $\boldsymbol{L_k D_k^{-1} U_k x_k^m = B_k x_{k-1}^m + T_k x_{k+1}^{m-1} + R_k x_k^{m-1} + b_k}$
  end for
  $\boldsymbol{x_k^m = x_k^m + (1 - \omega) x_k^{m-1}}$
end for

waarbij het blok tri-diagonale stelsel opgelost kan worden als volgt:

$$\boldsymbol{L_k z_k^m} = \boldsymbol{B_k x_{k-1}^m + T_k x_{k+1}^{m-1} + R_k x_k^{m-1} + b_k} \tag{3.16}$$
$$\boldsymbol{U_k x_k^m} = \boldsymbol{D_k z_k^m} \ . \tag{3.17}$$

Voor de algoritmen van Planey en Planez verwijzen we naar Appendix B .

Opmerkingen

We hebben gezien in subparagraaf 3.2.2 dat als $\boldsymbol{T_k = 0}$ en $\boldsymbol{N_k = 0}$, dan zijn Linzyx en Linyzx exacte solvers. Deze bewering geldt niet voor de Planez solver, immers:
Als $\boldsymbol{N_k = 0}$ en $\boldsymbol{T_k = 0}$, dan volgt uit algoritme 3.3:

$$\boldsymbol{L_k D_k^{-1} U_k x_k^m = B_k x_{k-1}^m + \hat{R}_k x_k^{m-1} + b_k} \ \ , \tag{3.18}$$

waarbij de matrix $\boldsymbol{\hat{R}_k}$ alleen uit het onderste gedeelte van de matrix $\boldsymbol{R_k}$ bestaat, want aangezien $\boldsymbol{N_k = 0}$, geldt dat de elementen $\boldsymbol{r_{i,j,k}^N}$ van $\boldsymbol{R_k}$ gelijk zijn aan nul (zie beschrijving van de matrix $\boldsymbol{R_k}$).
Door de aanwezigheid van $\boldsymbol{\hat{R}_k x_k^{m-1}}$ in (3.18) kunnen we zien dat de Planez solver geen exacte solver is.
Analoog merken we op dat Planey geen exacte solver is indien geldt $\boldsymbol{T_j = 0}$ en $\boldsymbol{E_j = 0}$ en dat Planex geen exacte solver is indien geldt $\boldsymbol{T_i = 0}$ en $\boldsymbol{N_i = 0}$.

3.2.3 Space-TDMA

Analoog aan Plane-TDMA is Space-TDMA een iteratieve methode, die gebaseerd is op de standaard ILU-decompositie van de matrix $\boldsymbol{A}$ afkomstig van een drie-dimensionaal probleem. Dit kunnen we met behulp van het volgende iteratieve proces beschrijven:

$x^0 = 0$,
for $1 \leq m \leq M$ **do**
$LD^{-1}Ux^m = Rx^{m-1} + b$
end for

hierbij zijn L, D, U en R op dezelfde manier gedefinieerd als bij Plane-TDMA. De elementen van de diagonale matrix D worden gegeven door:

$$\begin{aligned} d_{i,j,k} = (a_P)_{i,j,k} - \frac{(a_W)_{i,j,k}\,(a_E)_{i-1,j,k}}{d_{i-1,j,k}} &+ \\ - \frac{(a_S)_{i,j,k}\,(a_N)_{i,j-1,k}}{d_{i,j-1,k}} &+ \\ - \frac{(a_B)_{i,j,k}\,(a_T)_{i,j,k-1}}{d_{i,j,k-1}} &\;, \end{aligned} \tag{3.19}$$

met $d_{1,1,1} = (a_P)_{1,1,1}$, $(a_E)_{0,j,k} = 0$, $(a_N)_{i,0,k} = 0$ en $(a_T)_{i,j,0} = 0$. We merken op dat Space-TDMA niet gerelaxeerd wordt in WISH3D.

3.3 Krylov deelruimte methoden

3.3.1 Inleiding

In de vorige paragraaf hebben we een aantal klassieke iteratieve methoden beschreven voor het oplossen van lineaire stelsels van de vorm $Ax = b$. In de huidige paragraaf zullen we in het kort enkele van de meest bekende Krylov deelruimte methoden beschrijven aan de hand van de theorie uit [6] en [7]. Om het idee hierachter te motiveren, gaan we uit van het klassieke iteratieve proces, dat we in vorige paragraaf tegenkwamen:

$$Px^m = (P - A)x^{m-1} + b \qquad \text{voor } m = 1, ..., M \;, \tag{3.20}$$

met P een niet-singuliere matrix. Het is eenvoudig om na te gaan dat het proces als volgt geschreven kan worden:

$$x^m = x + (I - P^{-1}A)(x^{m-1} - x) \qquad \text{voor } m = 1, ..., M \;, \tag{3.21}$$

waarbij x de exacte oplossing voorstelt. Als x^0 een gegeven beginschatting is, dan volgt hieruit:

$$x^m = x + (I - C)^m(x^0 - x) \qquad \text{voor } m = 1, ..., M \;, \tag{3.22}$$

met $C = P^{-1}A$.
We definieren de fout door $e^m = x - x^m$ en het residu door $r^m = b - Ax^m = Ae^m$ voor $m = 0, 1, ..., M$. We kunnen uitdrukking (3.22) anders schrijven door het residu en uitdrukking $C = P^{-1}A$ te gebruiken. We krijgen dan het volgende:

$$x^m = x - (I - C)^m C^{-1} P^{-1} r^0 \qquad \text{voor } m = 1, ..., M \;. \tag{3.23}$$

We kunnen $(I - C)^m$ zien als een m^{de} graads polynoom van C, waarbij de constante term gelijk is aan I: $p_m(C) = I + \sum_{i=1}^{m} \mu_i C^i$, $\mu_i \in \mathbf{R}$. We schrijven (3.15) daarom als volgt:

$$\begin{aligned} x^m &= x - p_m(C) C^{-1} P^{-1} r^0 \\ &= x - (I + \sum_{i=1}^{m} \mu_i C^i) C^{-1} P^{-1} r^0 \quad . \end{aligned} \tag{3.24}$$

Met $C^{-1} P^{-1} r^0 = x - x^0$ en $p_{m-1}(C) = \sum_{i=0}^{m-1} \mu_i C^i$ volgt:

$$x^m = x^0 - p_{m-1}(C) P^{-1} r^0 \qquad \text{voor } m = 1, ..., M \ , \tag{3.25}$$

Uit (3.16) kunnen we zien dat $x^m - x^0$ in de ruimte zit, opgespannen door de vektoren $\{P^{-1} r^0, C P^{-1} r^0, ..., C^{m-1} P^{-1} r^0\}$. In formule:

$$x - x^0 \in K^m \equiv K^m(C; P^{-1} r^0) \ , \tag{3.26}$$

waarbij $K^m(C; P^{-1} r^0)$ de Krylov ruimte is van dimensie m, horend bij C en $P^{-1} r^0$. K^m is een deelruimte van K^n, waarbij n gelijk is aan het aantal rijen (kolommen) van A . Deze ruimte wordt voorzien van een inwendig produkt $< .,. >$ en bijbehorende norm $||.||$. Uitgaande van een start vektor r^0 wordt de Krylov ruimte K^m voorzien van een basis $\{p^0, p^1, ..., p^{m-1}\}$. De vektoren p^i, i=0,...,m-1, worden de zoekrichtingen genoemd. Als $x^m \in K^m$, dan geldt: $x^m = \sum_{i=0}^{m-1} \alpha_i^{(m)} p^i$, $\alpha_i^{(m)} \in \mathbf{R}$. Voor bepaalde Krylov methoden kiezen we p^i zodanig dat $\alpha_i^{(m)}$ niet van m afhangt (bijvoorbeeld de CG en GCR methode), dan geldt: $x^m = \sum_{i=0}^{m-1} \alpha_i p^i = x^{m-1} + \alpha_{m-1} p^{m-1}$.
Nu is het de bedoeling om x^m te vinden zodanig dat $||x - x^m||$ minimaal is ten opzichte van een geschikte norm. Dit komt neer op het volgende minimaliseringsprobleem:

$$||x - x^m|| = min_{y \in K^m} ||x - y|| \ . \tag{3.27}$$

De benadering x^m, die aan (3.27) voldoet, heeft de volgende eigenschappen:

1. x^m is uniek.

2. $< x - x^m, y >= 0 \quad \forall y \in K^m$: orthogonaliteitsvoorwaarde.

De methoden, die het minimaliseringsprobleem (3.27) oplossen, worden de Krylov deelruimte methoden genoemd. We zullen enkele hiervan kort bespreken aan de hand van de theorie uit [6] en [7]. Aangezien deze methoden iteratief zijn zullen we het volgende stopcriterium gebruiken:

$$||b - A x^m||_2 \leq \epsilon ||b||_2 \qquad \text{met } 0 < \epsilon \ll 1 \ . \tag{3.28}$$

Als ϵ vervangen wordt door $\frac{\epsilon}{K_2(A)}$, waarbij $K_2(A)$ het conditiegetal van A voorstelt, dan kan men laten zien dat het volgende geldt voor de relatieve fout:

$$\frac{\|x - x^m\|_2}{\|x\|_2} \leq K_2(A)\frac{\|b - Ax^m\|_2}{\|b\|_2} \leq \epsilon \quad . \tag{3.29}$$

Hierbij is het conditiegetal in het algemeen niet bekend. Maar er zijn wel iteratieve methoden beschikbaar voor het benaderen van dit getal.

3.3.2 Geconjugeerde Gradiënten methode

De Geconjugeerde Gradiënten methode (CG methode) is bedoeld voor het oplossen van lineaire stelsels $Ax = b$, met A symmetrisch en positief definiet. In de praktijk wordt A meestal gepreconditioneerd met een matrix P. Matrix P moet ook symmetrisch en positief definiet zijn. Dan volgt $P = LL^T$.
We beschouwen $C = L^{-1}AL^{-T}$. Voor de CG methode geldt het volgende minimaliseringsprobleem:

$$\|x - x^m\|_C = min_{y \in K^m}\|x - y\|_C \quad , \tag{3.30}$$

waarbij $\|.\|_C$ de norm is, gedefinieerd door:

$$\|x\|_C^2 = < x, x >_C = < x, Cx > \quad . \tag{3.31}$$

Het oplossen van dit minimaliseringsprobleem leidt tot het volgende algoritme:

Algoritme 3.5 (CG)

Gegeven een lineair stelsel $Ax = b$, die gepreconditioneerd is met een matrix P^{-1}. Startend met een beginschatting x^0 wordt een benadering x^M berekend voor de exacte oplossing x met behulp van het volgende iteratieve proces:

$r^0 = b - Ax^0$
los op :$Pz^0 = r^0$
$\rho = < r^0, z^0 >$
while $(1 \leq m \leq M) \wedge (\|r^m\|_2 > \epsilon\|b\|_2)$ **do**
  $\sigma = < p^m, Ap^m >$
  $\alpha = \frac{\rho}{\sigma}$
  $x^{m+1} = x^m + \alpha p^m$
  $r^{m+1} = r^m - \alpha Ap^m$
  los op :$Pz^{m+1} = r^{m+1}$
  $\rho_o = \rho$
  $\rho = < r^{m+1}, z^{m+1} >$
  $\beta = \frac{\rho}{\rho_o}$

$p^{m+1} = z^{m+1} + \beta p^m$

end while

De CG methode heeft drie belangrijke eigenschappen:

1. **Optimaliteit :** De fout gemeten in de $\|.\|_C$ norm: $\|e^m\|_C = \|x - x^m\|_C$ is minimaal.

2. **Korte recurrenties :** Om de benadering x^m te bepalen, zijn alleen de gegevens uit $(m-1)^{de}$ iteratie nodig. Hieruit volgt dat het rekenwerk en de hoeveelheid geheugenruimte niet toenemen als het aantal iteraties toeneemt.

3. **Robuustheid :** Bij de m^{de} iteratie kan door nul gedeeld worden als $\rho = 0$ of $\sigma = 0$. Als $\rho = 0$ dan geldt $r^m = 0$; maar hieruit volgt dat x^m gelijk is aan de exacte oplossing x. Als $\sigma = 0$ dan geldt $x^m = x^{m-1}$ en hieruit volgt $x = x^n = x^{n-1} = ... = x^m = x^{m-1}$.

Convergentie

Als matrix C de eigenwaarden $\lambda_j \in \mathbf{R}$, j=1,...,n, heeft dan kan men laten zien dat er geldt:

$$\|e^m\|_C = 2\left(\frac{\sqrt{K_2(C)}-1}{\sqrt{K_2(C)}+1}\right)^m \|e^0\|_C \quad , \tag{3.32}$$

waarbij $K_2(C)$ het conditiegetal van C is, gedefinieerde door:

$$K_2(C) = \frac{max_{1\leq j\leq n}|\lambda_j|}{min_{1\leq j\leq n}|\lambda_j|} \quad . \tag{3.33}$$

We zien uit (3.22) dat het iteratieve proces van CG methode snel convergeert als het conditiegetal in de buurt van 1 is.

In de praktijk blijkt dat de coëfficiënten matrices vaak niet symmetrisch zijn. De CG methode is niet geschikt voor dit type van matrices. In de laatste jaren zijn er generalisaties van deze methode gekomen voor het oplossen van niet-symmetrische lineaire stelsels. We zullen hier een drietal hiervan kort beschrijven, namelijk de Bi-CGSTAB, GCR en GMRES methode. Voor meer details over deze methoden verwijzen we naar [7].

3.3.3 Bi-CGSTAB methode

In de vorige subparagraaf bleek dat het gebruik maken van korte recurrenties één van de grootste voordelen is van de CG methode. Om van dit voordeel gebruik te maken in het geval van niet-symmetrische lineaire stelsels, wordt er naast het originele lineaire stelsel $\boldsymbol{Ax} = \boldsymbol{b}$ ook een extra lineair stelsel $\boldsymbol{A}^T\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{b}}$ beschouwd, waarbij $\tilde{\boldsymbol{b}}$ een bepaalde vektor is, die verder geen rol speelt. Met de twee stelsels vormen we het volgende (bi)lineaire stelsel:

$$\boldsymbol{Bv} = \boldsymbol{w} \quad , \tag{3.34}$$

met

$$\boldsymbol{B} = \begin{pmatrix} \mathbf{0} & \boldsymbol{A}^T \\ \boldsymbol{A} & \mathbf{0} \end{pmatrix} \quad , \quad \boldsymbol{v} = \begin{pmatrix} \boldsymbol{x} \\ \tilde{\boldsymbol{x}} \end{pmatrix} \quad \text{en} \quad \boldsymbol{w} = \begin{pmatrix} \tilde{\boldsymbol{b}} \\ \boldsymbol{b} \end{pmatrix} \quad .$$

Aangezien de matrix $\boldsymbol{B}$ symmetrisch is, ligt het dus voor de hand om de CG methode toe te passen op (3.24). Dit zal resulteren in de Bi-CG methode (Bi-orthogonal Conjugate Gradient method), die aan de basis staat van de Bi-CGSTAB methode. Als $\boldsymbol{A}$ symmetrisch is dan reduceert de Bi-CG tot de CG methode. De verwachting is dus dat Bi-CG bijna dezelfde eigenschappen heeft als CG indien $\boldsymbol{A}$ bijna symmetrisch is.
Ondanks het gebruik maken van korte recurrenties, heeft Bi-CG het nadeel dat er per iteratie twee matrix-vektor produkten bepaald moeten worden. Een methode, die dit nadeel verkleint, is de CGS methode (Conjugate Gradient Squared method). Deze kan twee maal zo snel convergeren dan Bi-CG ten koste van bijna evenveel rekenwerk. Hierbij wordt er uitgegaan van het residu $\boldsymbol{r}^m$ afkomstig van Bi-CG. Voor dit residu geldt $\boldsymbol{r}^m = \boldsymbol{R}_m(\boldsymbol{A})\boldsymbol{r}^0$, waarbij $\boldsymbol{R}_m(\boldsymbol{A})$ een $\boldsymbol{m}^{de}$ graads polynoom is. In [6] is bewezen dat voor het residu $\tilde{\boldsymbol{r}}^m$ afkomstig van CGS geldt: $\tilde{\boldsymbol{r}}^m = (\boldsymbol{R}_m(\boldsymbol{A}))^2\boldsymbol{r}^0$. Op grond hiervan verwachten we als $\boldsymbol{r}^m$ naar nul toegaat dan zal $\tilde{\boldsymbol{r}}^m$ twee maal zo snel naar nul toegaan.
Het grootste nadeel van CGS is dat als de convergentie van Bi-CG onregelmatig gedrag vertoont, deze onregelmatigheid versterkt wordt in de convergentie van CGS. Om dit gedrag te stabiliseren wordt de Bi-CGSTAB methode (Bi-CG STABilized) gebruikt. Hiertoe wordt het residu afkomstig van CGS nu geschreven als $\tilde{\boldsymbol{r}}^m = \boldsymbol{Q}_m(\boldsymbol{A})\boldsymbol{R}_m(\boldsymbol{A})\boldsymbol{r}^0$, waarbij het polynoom $\boldsymbol{Q}_m(\boldsymbol{A})$ gedefinieerd is door:

$$\boldsymbol{Q}_m(\boldsymbol{A}) = \prod_{i=1}^{m}(\boldsymbol{I} - \omega_i\boldsymbol{A}) = \boldsymbol{Q}_{m-1}(\boldsymbol{A})(\boldsymbol{I} - \omega_m\boldsymbol{A}) \quad , \tag{3.35}$$

hierbij wordt het gewicht $\omega_i \in \mathbf{R}$ zo gekozen dat $||\tilde{\boldsymbol{r}}||_2$ minimaal is.
We kunnen de Bi-CGSTAB methode beschrijven aan de hand van het volgende algoritme:

Algoritme 3.6 (Bi-CGSTAB)

Gegeven een lineair stelsel $\boldsymbol{Ax} = \boldsymbol{b}$, gepreconditioneerd met een matrix $\boldsymbol{P}^{-1}$. Gestart met een beginschatting $\boldsymbol{x}^0$ wordt een benadering $\boldsymbol{x}^M$ berekend voor de exacte oplossing $\boldsymbol{x}$ met behulp van het volgende iteratieve proces:

$\tilde{r}^0 = b - Ax^0$
$p^0 = q^0 = 0$
$\rho = \omega = \alpha = 1$
while $(1 \leq m \leq M) \wedge (\|\tilde{r}^m\|_2 > \epsilon\|b\|_2)$ **do**
$\rho_o = \rho$
$\rho = < \tilde{r}^0, \tilde{r}^{m-1} >$
$\beta = \frac{\alpha\rho}{\omega\rho_o}$
$z = p^m - \beta p^{m-1}$
los op : $Pp^m = \tilde{r}^{m-1} - \beta\omega q^{m-1}$
$\sigma = < \tilde{r}^0, Ap^m >$
$\alpha = \frac{\rho}{\sigma}$
los op : $Ps^m = \tilde{r}^{m-1} - \alpha q^m$
$\omega = \frac{<As^m, s^m>}{<As^m, As^m>}$
$x^m = x^{m-1} + \alpha p^m + \omega s^m$
$\tilde{r}^m = s - \omega As^m$
end while

Ondanks het gebruik maken van korte recurrenties kan men laten zien dat de Bi-CGSTAB methode niet robuust en optimaal is.

We zullen in de volgende paragraaf twee methoden bespreken, die wel robuust en optimaal zijn. Maar het nadeel van deze methoden is het gebruik maken van lange recurrenties, dat wil zeggen dat het rekenwerk en de benodigde geheugenruimte toenemen met het aantal iteraties.

3.3.4 Krylov methoden met lange recurrenties

We zullen twee varianten van Krylov methoden met lange recurrenties kort bespreken, namelijk GCR (Generalized Conjugate Residuals) en GMRES (Generalized Minimal RESiduals). Bij deze methoden wordt de oplossing x benaderd met een vektor $x^m = x^0 + Ay^m$, waarbij $y^m = y$ aan het volgende minimaliseringsprobleem voldoet:

$$\|e^m\|_{A^TA} = \|r^m\|_2 = min_{y \in K^m}\|r^0 - Ay\|_2 \quad , \tag{3.36}$$

met $K^m \equiv K^m(A, r^0$ de Krylov ruimte van dimensie m.

a. GCR methode

We schrijven de benadering x^m als volgt:

$$x^m = x^0 + \sum_{i=0}^{m-1} \gamma_i r^i \quad , \tag{3.37}$$

hierbij hebben we de residuen r^i als zoekrichtingen gekozen. Hieruit volgt het residu r^m:

$$r^m = r^0 - \sum_{i=0}^{m-1} \gamma_i A r^i \quad . \tag{3.38}$$

We definieren de ruimte $\mathcal{V}_m = span\{v^1, v^2, ..., v^m\}$, waarbij $\{v^1, v^2, ..., v^m\}$ een orthonormale basis is verkregen door het toepassen van het Gram-Schmidt proces op de vektoren $\{Ar^0, Ar^1, ..., Ar^{m-1}\}$

$\underline{Gram - Schmidtproces}$

$x^0 =$ beginschatting
$r^0 = b - Ax^0$
for $j = 1, 2, ..., m$ **do**
  $q^j = Aq^{j-1} - \sum_{i=1}^{j-1} < Ar^{j-1}, v^i > v^i$
  $v^j = \frac{q^j}{\|q^j\|_2}$
end for

In dit geval geldt voor het residu:

$$r^m = r^0 - \sum_{j=1}^{m} \alpha_{j-1} v^j \quad . \tag{3.39}$$

Hiermee kunnen we het minimalisringsprobleem (3.) als volgt weergeven:

$$\|r^m\|_2 = min_{y \in \mathcal{V}_m} \|r^0 - y\|_2 \quad . \tag{3.40}$$

Het residu r^m voldoet aan de volgende orthogonaliteitseigenschap:

$$< r^m, v^j >= 0 \qquad j = 1, 2, ..., m. \tag{3.41}$$

Met behulp van (3.41), (3.39) en de orthogonaliteitseigenschap van de vektoren v^j kunnen we de coëfficiënten α_j eenvoudig bepalen:

$$\alpha_{j-1} =< r^{j-1}, v^j > \qquad j = 1, 2, ..., m. \tag{3.42}$$

De hier besproken GCR methode, toegepast op het gepreconditioneerde lineaire stelsel $P^{-1}Ax = P^{-1}b$, resulteert in het volgende algoritme:

Algoritme 3.7 (GCR)

Gegeven een lineair stelsel $Ax^m = b$, gepreconditioneerd met de matrix P^{-1}. Startend met een beginschatting x^0 wordt een benadering x^M voor de exacte oplossing x met behulp van het volgende iteratieve proces:

$r^0 = b - Ax^0$
while $(1 \leq m \leq M) \wedge (\|r^m\|_2 > \epsilon \|b\|_2)$ **do**
  los op : $Pz^m = r^{j-1}$

$p^m = Az^m$
$p^m = p^m - \sum_{i=1}^{m-1} < p^m, v^i > v^i$
$q^m = z^m - \sum_{i=1}^{m-1} < p^m, v^i > w^i$
$v^m = p^m / \|p^m\|_2$
$w^m = q^m / \|p^m\|_2$
$x^m = x^{m-1} + < r^{m-1}, v^m > w^m$
$r^m = r^{m-1} - < r^{m-1}, v^m > v^m$
end while

Om de hoeveelheid rekenwerk en benodigde geheugenruimte te verlagen worden een van de volgende technieken toegepast:

1. ***Herstarten***: na k iteraties wordt algoritme 3.3 herstart. De zoekrichtingen v^k en w^k worden eerst verwijderd, daarna wordt het algoritme opnieuw toegepast met beginschatting $x^0 = x^k$.

2. ***trunkeren***: laat slechts k zoekrichtingen toe. Als het aantal iteraties groter is dan k, dan wordt een oude zoekrichting v^{k_m} (en w^{k_m}) overgeschreven. Hierbij geldt $k_m \leq k$.

Convergentie

We kunnen de fout e^m schrijven als $e^m = R_m(A)e^0$ met $R_m(0) = 1$, waarbij R_m een m^{de} graadspolynoom is. Hieruit volgt $r^m = R_m(A)r^0$. We definieren vervolgens de volgende klasse van polynomen: $[p_m, 1]$, dit betekent dat als $p \in [p_m, 1]$ dan is p een polynoom van de graad m met $p(0) = 1$. Het blijkt dat (3.26) equivalent is met het volgende minimaliseringsprobleem:

$$\|r^m\|_2 = min_{p \in [p_m,1]} \|p(A)r^0\|_2 \quad , \tag{3.43}$$

Nu vermelden we de volgende stelling uit [6] over de convergentie van GCR:
Stelling: Zij λ_j, j=1,...,n, de eigenwaarden van de matrix A. Definieer $\epsilon^{(m)}$ door:

$$\epsilon^{(m)} = min_{p \in [p_m,1]} max_{\lambda_j} |p(\lambda_j)| \quad . \tag{3.44}$$

Stel dat A diagonalizeerbaar is. Schrijf A nu als volgt: $A = X \Lambda X^{-1}$, waarbij $\Lambda = diag(\lambda_1, ..., \lambda_n)$ dan geldt:

1. $\|r^m\|_2 \leq K_2(X) \epsilon^{(m)} \|r^0\|_2$,

2. Als bovendien alle eigenwaarden (in het complexe vlak) binnen een cirkel liggen met middelpunt $C \in \mathbf{R}$, $C > 0$ en een straal R zodat $R < C$ dan geldt:

$$\epsilon^{(m)} \leq \left(\frac{R}{C}\right)^m \quad . \tag{3.45}$$

b. GMRES methode

Voor de eenvoud zullen we de GMRES methode beschrijven zonder preconditionering, dat wil zeggen dat voor de preconditioneringsmatrix geldt: $\boldsymbol{P} = \boldsymbol{I}$.
Zij $\{\boldsymbol{v}^1, \boldsymbol{v}^2, ..., \boldsymbol{v}^m\}$ een orthonormale basis voor de Krylov ruimte $\boldsymbol{K}^m(\boldsymbol{A}; \boldsymbol{r}^0)$, verkregen met behulp van het Gram-Schmidt proces:

Gram − Schmidt proces

$\boldsymbol{x}^0 =$ beginschatting
$\boldsymbol{r}^0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^0$
$\boldsymbol{v}^1 = \frac{\boldsymbol{r}^0}{\|\boldsymbol{r}^0\|_2}$
for $\boldsymbol{j} = \boldsymbol{1}, \boldsymbol{2}, ..., \boldsymbol{m}$ **do**
  $\boldsymbol{p}^j = \boldsymbol{A}\boldsymbol{v}^j - \sum_{i=1}^{j-1} < \boldsymbol{A}\boldsymbol{v}^j, \boldsymbol{v}^i > \boldsymbol{v}^i$
  $\boldsymbol{h}_{j+1,j} = \|\boldsymbol{p}^{j+1}\|_2$
  $\boldsymbol{v}^j = \frac{\boldsymbol{p}^j}{\boldsymbol{h}_{j+1,j}}$
end for

We definieren de matrix $\boldsymbol{V}_m \in \mathbf{R}^{n\times m}$ door $\boldsymbol{V}_m = (\boldsymbol{v}^1 ... \boldsymbol{v}^m)$ en de matrix $\boldsymbol{H}_m$ door $\boldsymbol{H}_m = (\boldsymbol{h}_{i,j})$, waarbij

$$\boldsymbol{h}_(\boldsymbol{i}, \boldsymbol{j}) =< \boldsymbol{A}\boldsymbol{v}^j, \boldsymbol{v}^i > \qquad \boldsymbol{j} = \boldsymbol{1}, ..., \boldsymbol{m} \quad \text{en} \quad \boldsymbol{i} = \boldsymbol{1}, ..., \boldsymbol{j}. \tag{3.46}$$

Gebruik makend van deze definities kunnen we nagaan dat na m iteraties van het Gram-Schmidt proces de volgende uitdrukking geldt:

$$\boldsymbol{A}\boldsymbol{V}_m = \boldsymbol{V}_{m+1}\tilde{\boldsymbol{H}}_m, \tag{3.47}$$

hierbij is $\tilde{\boldsymbol{H}}_m \in \mathbf{R}^{m+1\times m}$ de Hessenberg matrix, gegeven door:

$$\tilde{\boldsymbol{H}}_m = \begin{pmatrix} \boldsymbol{H}_m \\ \boldsymbol{h}_{m+1,m}\boldsymbol{e}_m^T \end{pmatrix} \quad ,$$

met $\boldsymbol{e}_m$ de $\boldsymbol{m}^{de}$ eenheidsvektor.
Nu kunnen we de benadering $\boldsymbol{x}^m$ voor de exacte oplossing $\boldsymbol{x}$ als volgt bepalen:

$$\boldsymbol{x}^m = \boldsymbol{x}^0 + \boldsymbol{V}_m\boldsymbol{y}^m \quad , \tag{3.48}$$

zodanig dat $\boldsymbol{y}^m = \boldsymbol{y}$ aan het volgende minimaliseringsprobleem voldoet:

$$\begin{aligned} \|\boldsymbol{r}^m\|_2 &= \boldsymbol{min}_{y\in\mathbf{R}^m}\|\boldsymbol{r}^0 - \boldsymbol{A}\boldsymbol{V}_m\boldsymbol{y}^m\|_2 \\ &= \boldsymbol{min}_{y\in\mathbf{R}^m}\|\boldsymbol{r}^0 - \boldsymbol{V}_{m+1}\tilde{\boldsymbol{H}}_m\boldsymbol{y}\|_2 \quad . \end{aligned} \tag{3.49}$$

In (3.49) hebben we (3.47) gebruikt om het onderste gelijkteken te verkrijgen. We kunnen (3.41) verder uitwerken door op te merken dat de orthogonale matrices de norm $\|.\|_2$ invariant laten:

$$\|\boldsymbol{r}^0 - \boldsymbol{V}_{m+1}\tilde{\boldsymbol{H}}_m\boldsymbol{y}\|_2 = \|\boldsymbol{V}_{m+1}\|\boldsymbol{r}^0\|_2\boldsymbol{e}_1 - \boldsymbol{V}_{m+1}\tilde{\boldsymbol{H}}_m\boldsymbol{y}\|_2$$

$$
\begin{aligned}
&= \|\boldsymbol{V}_{m+1}(\|\boldsymbol{r}^0\|_2\boldsymbol{e}_1 - \tilde{\boldsymbol{H}}_m\boldsymbol{y})\|_2 \\
&= ((\|\boldsymbol{r}^0\|_2\boldsymbol{e}_1 - \tilde{\boldsymbol{H}}_m\boldsymbol{y})^T\boldsymbol{V}_{m+1}^T\boldsymbol{V}_{m+1}(\|\boldsymbol{r}^0\|_2\boldsymbol{e}_1 - \tilde{\boldsymbol{H}}_m\boldsymbol{y}))^{1/2} \\
&= \|\ \|\boldsymbol{r}^0\|_2\boldsymbol{e}_1 - \tilde{\boldsymbol{H}}_m\boldsymbol{y}\ \|_2 \quad , \qquad (3.50)
\end{aligned}
$$

waarbij het vierde gelijkteken uit $\boldsymbol{V}_{m+1}^T\boldsymbol{V}_{m+1} = \boldsymbol{I}_{m+1}$ volgt.
Dus, het minimaliseringsprobleem (3.41) wordt:

$$\|\boldsymbol{r}^m\|_2 = \|\ \|\boldsymbol{r}^0\|_2\boldsymbol{e}_1 - \tilde{\boldsymbol{H}}_m\boldsymbol{y}\ \|_2 \quad . \qquad (3.51)$$

Tot slot vermelden we dat GMRES het voordeel heeft dat de toename van het rekenwerk en de geheugenruimte per iteratie minder is ten opzichte van GCR. Een nadeel is dat GMRES niet getrunkeerd kan worden.

3.4 Preconditioners voor de Krylov solvers van WISH3D

In de vorige paragraaf hebben we een viertal Krylov solvers (CG, Bi-CGSTAB, GCR en GMRES) voorgesteld voor het oplossen van stelsels vergelijkingen van de vorm:

$$\boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{P}^{-1}\boldsymbol{b} \quad , \qquad (3.52)$$

waarbij $\boldsymbol{P}$ een preconditioneringsmatrix (preconditioner) is. Een preconditioner maakt de onderlinge ligging van de in absolute waarde kleinste en grootste eigenwaarde van de matrix $\boldsymbol{A}$ gunstig, hetgeen een convergentieversnelling kan opleveren. In deze paragraaf zullen we een aantal preconditioners aangeven, waaronder de matrix $\boldsymbol{P}$ horend bij de WISH3D solvers die besproken zijn in paragraaf 3.2 .

3.4.1 Gauss-Jacobi preconditioner

Dit wordt ook wel een diagonaalschalings preconditioner genoemd. Hierbij kiezen we $\boldsymbol{P} = \boldsymbol{diag}(\boldsymbol{A})$. Het voordeel hiervan is dat dit tot het versnellen van de convergentie kan leiden ten koste van weinig rekenwerk en geheugenruimte als $\boldsymbol{A}$ diagonaal dominant is.

3.4.2 Line-TDMA preconditioner

We kiezen de matrix $\boldsymbol{P}$ zoals deze beschreven is in subparagraaf 3.2.1 . Bij een drie-dimensionaal probleem en afhankelijk van de gekozen gridlijn heeft Line-TDMA preconditioner zes varianten, te weten de Linexyz, Linexzy, Lineyxz, Lineyzx, Linezxy en Linezyx preconditioner. Het voordeel van Line-TDMA ten opzichte van de Gauss-Jacobi preconditioner is dat de matrix $\boldsymbol{P}$ meer op $\boldsymbol{A}$ lijkt dan $\boldsymbol{diag}(\boldsymbol{A})$. De verwachting is dus dat Line-TDMA tot een betere convergentieversnelling leidt dan de Gauss-Jacobi preconditioner, maar wel ten koste van meer rekenwerk en geheugenruimte. Het nadeel van deze preconditioner is dat $\boldsymbol{P}$ niet symmetrisch is voor twee- en drie-dimensionale problemen, waardoor $\boldsymbol{P}$ niet als preconditioner gekozen kan worden voor de CG methode. Om een symmetrische Line-TDMA preconditioner te verkrijgen kunnen we $\boldsymbol{P}$ als volgt kiezen:

$$\boldsymbol{P} = (\boldsymbol{G} - \boldsymbol{U})\boldsymbol{G}^{-1}(\boldsymbol{G} - \boldsymbol{L}) \quad , \tag{3.53}$$

waarbij de matrices $\boldsymbol{G}$, $\boldsymbol{U}$ en $\boldsymbol{L}$ beschreven zijn in subparagraaf 3.2.1 . We merken op dat wegens symmetrie geldt $\boldsymbol{L}^{\boldsymbol{T}} = \boldsymbol{U}$.

3.4.3 Plane-TDMA preconditioner

We kiezen de matrix $\boldsymbol{P}$ zoals deze beschreven is in subparagraaf 3.2.2. Voor een drie-dimensionale probleem heeft Plane-TDMA drie varianten, namelijk Planex, Planey en Planez. Bij veel twee- en drie-dimensionaal problemen, die met WISH3D uitgerekend worden, convergeert Plane-TDMA beter dan Line-TDMA solver. Op grond hiervan verwachten we dat Plane-TDMA een betere convergentieversnelling oplevert dan de Line-TDMA preconditioner, maar wel ten koste van meer rekentijd en geheugenruimte in verband met het bepalen van de diagonale matrices $\boldsymbol{D_k}$ (zie subparagraaf 3.2.2). De Plane-TDMA preconditioner is niet symmetrisch voor een drie-dimensionaal probleem. In het geval van de CG methode zouden we, net zoals bij de Line-TDMA preconditioner, $\boldsymbol{P} = (\boldsymbol{G} - \boldsymbol{T})\boldsymbol{G}^{-1}(\boldsymbol{G} - \boldsymbol{B})$ kiezen, waarbij de matrices $\boldsymbol{G}$, $\boldsymbol{T}$ en $\boldsymbol{B}$ beschreven zijn in subparagraaf 3.2.2 . Maar aangezien de matrix $\boldsymbol{G}$ een blok struktuur heeft, kost het oplossen van een stelsel $\boldsymbol{P}\boldsymbol{w} = \boldsymbol{v}$ te veel rekenwerk. We nemen daarom genoegen met $\boldsymbol{P} = \boldsymbol{G}$ als preconditioner voor de CG methode. We merken op dat deze beperking alleen voor drie-dimensionale problemen geldt.

3.4.4 Space-TDMA (Standaard ILU) preconditioner

In subpragraaf 3.2.3 hebben we gezien dat de Space-TDMA solver een iteratieve methode is, die gebasserd is op het standaard ILU-decompositie. De Space-TDMA preconditioner is daarom niets anders dan de standaard ILU preconditioner, die gegeven wordt door $\boldsymbol{P} = \boldsymbol{L}\boldsymbol{D}^{-1}\boldsymbol{U}$, waarbij de matrices $\boldsymbol{L}$, $\boldsymbol{D}$ en $\boldsymbol{U}$ beschreven zijn in subparagraaf 3.2.3 . Het voordeel van de Space-TDMA (Standaard ILU) preconditioner is dat deze wel symmetrisch is voor een symmetrische matrix.

3.4.5 MILU preconditioner

In [7] (hoofdstuk 4) heeft men gemerkt dat voor tweede-orde elliptische randvoorwaarde problemen geldt voor de ILU preconditioner dat $\boldsymbol{K_2}(\boldsymbol{P}^{-1}\boldsymbol{A}) = \boldsymbol{K_2}(\boldsymbol{A}) = \mathcal{O}(\boldsymbol{h}^{-2})$, waarbij $\boldsymbol{h}$ de discretisatie afstand voorstelt in het geval van een uniform rooster. Om het conditiegetal tot een orde lager te reduceren wordt de fill-in, die bij de ILU-decompositie optreedt, bij de hoofddiagonaal elementen opgeteld en vervolgens wordt de volgende stap in de ILU-decompositie gemaakt. Het enige verschil met de ILU-decompositie is dat nu voor de hoofddiagonaal elementen van $\hat{U}$ het volgende geldt: $\hat{L}\hat{U}e = Ae$, waarbij $e = (1, ..., 1)^T$. Hieruit volgt:

$$\begin{aligned} d_{i,j,k} &= (a_P)_{i,j,k} - \frac{(a_W)_{i,j,k}\sigma_{i-1,j,k}}{d_{i-1,j,k}} + \\ &\quad - \frac{(a_S)_{i,j,k}\sigma_{i,j-1,k}}{d_{i,j-1,k}} + \\ &\quad - \frac{(a_B)_{i,j,k}\sigma_{i,j,k-1}}{d_{i,j,k-1}} \quad , \end{aligned} \tag{3.54}$$

met $\sigma_{i,j,k} = (a_E)_{i,j,k} + (a_N)_{i,j,k} + (a_T)_{i,j,k}$.
De hier beschreven preconditioner heet MILU (Modified ILU) preconditioner.

3.4.6 RILU preconditioner

De RILU (Relaxed ILU) preconditioner is gebaseerd op hetzelfde idee als de MILU preconditioner. Het verschil is dat nu $\sigma_{i,j,k}$ als volgt gedefinieerd wordt:

$$\begin{aligned} \sigma_{i-1,j,k} &= (a_E)_{i-1,j,k} + (1-\omega)((a_N)_{i-1,j,k} + (a_T)_{i-1,j,k}) \quad , \\ \sigma_{i,j-1,k} &= (a_N)_{i,j-1,k} + (1-\omega)((a_E)_{i,j-1,k} + (a_T)_{i,j-1,k}) \quad , \\ \sigma_{i,j,k-1} &= (a_T)_{i,j,k-1} + (1-\omega)((a_E)_{i,j,k-1} + (a_N)_{i,j,k-1}) \quad , \end{aligned}$$

waarbij $\omega \in [0, 1]$ de relaxatie faktor voorstelt. We merken op dat RILU identiek is aan ILU als $\omega = 1$, terwijl als $\omega = 0$ dan is RILU identiek aan MILU. Het voordeel van RILU is dus dat een aantal ILU varianten verkregen kan worden door het varieren van de relaxatie faktor.

3.5 Het oplossen van de Navier-Stokes vergelijkingen met SIMPLE methode

Bij het schrijven van de impulsvergelijkingen (2.9) in de vorm van de convectie-diffusie vergelijking (2.28) hebben we de drukafgeleide $\frac{\partial p}{\partial x_i}$ opgenomen in de bronterm. Het probleem hierbij is dat de snelheidscomponenten niet uitgerekend kunnen worden zonder dat de druk bekend is. Om dit probleem te ondervangen beschouwen we eerst de Navier-Stokes vergelijkingen (de impulsvergelijkingen en de continuïteitsvergelijking) uit hoofdstuk 2:

$$\frac{\partial \rho u_i}{\partial t} + \frac{\partial}{\partial x_j} = S_i - \frac{\partial p}{\partial x_i} \qquad i = 1, 2, 3 \tag{3.55}$$

$$J_j = \rho u_j u_i - \Gamma_{u_i} \frac{\partial u_i}{\partial x_j} \qquad j = 1,2,3 \ .$$

$$\frac{\partial \rho}{\partial t} + \frac{\partial u_j}{\partial x_j} = 0 \ . \tag{3.56}$$

We merken op dat nu de bronterm S_i geen drukafgeleide bevat.
Het discretiseren van (3.55) en (3.56) met behulp van de technieken uit hoofdstuk 2 levert de volgende lineaire stelsels op:

$$Q_1 u_1 + G_1 p = b_1 \ , \tag{3.57}$$

$$Q_2 u_2 + G_2 p = b_2 \ , \tag{3.58}$$

$$Q_3 u_3 + G_3 p = b_3 \ , \tag{3.59}$$

$$G_1^* u_1 + G_2^* u_2 + G_3^* u_3 = b_4 \ . \tag{3.60}$$

We schrijven deze stelsels als één lineair stelsel:

$$Ay = b \ , \tag{3.61}$$

waarbij

$$A = \begin{pmatrix} Q_1 & 0 & 0 & G_1 \\ 0 & Q_2 & 0 & G_2 \\ 0 & 0 & Q_3 & G_3 \\ G_1^* & G_2^* & G_3^* & 0 \end{pmatrix} \ , \ y = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ p \end{pmatrix} \ , \ b = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix} \ .$$

We merken op dat voor Navier-Stokes vergelijkingen de matrix A van de snelheidscomponenten $u_i, i = 1,2,3$ afhangt. Voor het oplossen van het stelsel (3.61) wordt in WISH3D de iteratieve methode SIMPLE (Semi Implicit Method for Pressure-Linked Equations) of de variant daarop SIMPLER (SIMPLE Revised) gebruikt. We zullen in deze paragraaf de twee methoden beschrijven aan de hand van de distributieve iteratie methode, die beschreven is in [10, blz. 244-255]. Hierbij merken we op dat door de aanwezigheid van de blok nul-matrix in de hoofddiagonaal van A is het niet mogelijk om een splitsingsmatrix P (zie voor de analogie paragraaf 3.2) te vinden, die eenvoudig inverteerbaar is. Hiertoe wordt A met een postconditioneringsmatrix C vermenigvuldigd zodat de volgende splitsing gemaakt kan worden:

$$AC = P - (P - AC) \ , \tag{3.62}$$

waarbij

$$C = \begin{pmatrix} I & 0 & 0 & -D_1^{-1} G_1 \\ 0 & I & 0 & -D_2^{-1} G_2 \\ 0 & 0 & I & -D_3^{-1} G_3 \\ 0 & 0 & 0 & I \end{pmatrix} \ , \ P = \begin{pmatrix} Q_1 & 0 & 0 & 0 \\ 0 & Q_2 & 0 & 0 \\ 0 & 0 & Q_3 & 0 \\ G_1^* & G_2^* & G_3^* & R \end{pmatrix} \ ,$$

met $\boldsymbol{D_i = diag(Q_i)(i = 1,2,3)}$ en $\boldsymbol{R = -\sum_{i=1}^{3} G_i^* D_i^{-1} G_i}$.
Nu kunnen we het stelsel (3.61) oplossen met het volgende iteratieve proces:

$$\boldsymbol{y^{n+1} = y^n + CP^{-1}(b - Ay^n) \qquad n = 1,2,...,N} \ . \tag{3.63}$$

Dit kunnen we als volgt schrijven:

for $\boldsymbol{n = 1,...,N}$ **do**
$\boldsymbol{r^n = b - Ay^n}$
$\boldsymbol{P\delta\tilde{y} = r^n}$
$\boldsymbol{\delta y = C\delta\tilde{y}}$
$\boldsymbol{y^{n+1} = y^n + \delta y}$
end for

Uitgaande van dit proces kunnen we de SIMPLE methode beschrijven aan de hand van het volgende algoritme:

SIMPLE algoritme

$\boldsymbol{p^0}$ = begin schatting
$\boldsymbol{u_i^0 = 0 \qquad i = 1,2,3}$
for $\boldsymbol{n = 1,...,N}$ **do**
$***$ bepaal $\boldsymbol{r^n}$ $***$
$\boldsymbol{r_i^n = b_i - Q_i u_i^n - G_i p^n \qquad i = 1,2,3}$
$\boldsymbol{r_4^n = b_4 - \sum_{i=1}^{3} G_i^* u_i^n}$

$***$bepaal $\boldsymbol{P\delta\tilde{y} = r^n}$ $***$
$\boldsymbol{Q_i \delta\tilde{u}_i = r_i^n \qquad i = 1,2,3}$
$\boldsymbol{R\delta\tilde{p} = r_4^n - \sum_{i=1}^{3} G_i^* \delta\tilde{u}_i}$

$***$bepaal $\boldsymbol{\delta y = C\delta\tilde{y}}$ $***$
$\boldsymbol{\delta u_i = \delta\tilde{u}_i - D_i^{-1} G_i \delta\tilde{p} \qquad i = 1,2,3}$
$\boldsymbol{\delta p = \delta\tilde{p}}$

$***$bepaal $\boldsymbol{y^{n+1} = y^n + \delta y}$ $***$
$\boldsymbol{u_i^{n+1} = u_i^n + \delta u_i \qquad i = 1,2,3}$
$\boldsymbol{p^{n+1} = p^n + \delta p}$
end for

De SIMPLER methode wordt beschreven door het volgende iteratieve proces:

$$\begin{aligned} \boldsymbol{r^n} &= \boldsymbol{b - Ay^n} \\ \boldsymbol{y^{n+1}} &= \boldsymbol{y^n + CP^{-1}(r^n - AH^{-1}y^n) \qquad n = 1,2,...,N} \end{aligned} \tag{3.64}$$

met

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{D}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{D}_2 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{D}_3 & \mathbf{0} \\ \boldsymbol{G}_1^* & \boldsymbol{G}_2^* & \boldsymbol{G}_3^* & \boldsymbol{R} \end{pmatrix} .$$

Hieruit volgt het SIMPLER algoritme:

SIMPLER algoritme

$\boldsymbol{p}^0 =$ begin schatting
$\boldsymbol{u}_i^0 = \mathbf{0} \qquad i = 1, 2, 3$
for $n = 1, \dots, N$ **do**
  $* * *$bepaal $\qquad \boldsymbol{r}^n = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{y}^n * **$
  $\boldsymbol{r}_i^n = \boldsymbol{b}_i - \boldsymbol{Q}_i\boldsymbol{u}_i^n - \boldsymbol{G}_i\boldsymbol{p}^n \qquad i = 1, 2, 3$
  $\boldsymbol{r}_4^n = \boldsymbol{b}_4 - \sum_{i=1}^3 \boldsymbol{G}_i^*\boldsymbol{u}_i^n$

  $* * *$bepaal $\qquad \delta\bar{\boldsymbol{y}} = \boldsymbol{H}^{-1}\boldsymbol{y}^n * **$
  $\boldsymbol{D}_i\delta\bar{\boldsymbol{u}}_i = \boldsymbol{r}_i^n \qquad i = 1, 2, 3$
  $\boldsymbol{R}\delta\bar{\boldsymbol{p}} = \boldsymbol{r}_4^n - \sum_{i=1}^3 \boldsymbol{G}_i^*\delta\bar{\boldsymbol{u}}_i$

  $* * *$bepaal $\qquad \bar{\boldsymbol{r}}^n = \boldsymbol{r}^n - \boldsymbol{A}\boldsymbol{H}^{-1}\boldsymbol{r}^n * **$
  $\bar{\boldsymbol{r}}_i^n = \boldsymbol{r}_i - \boldsymbol{Q}_i\delta\bar{\boldsymbol{u}}_i^n - \boldsymbol{G}_i\delta\bar{\boldsymbol{p}} \qquad i = 1, 2, 3$
  $\bar{\boldsymbol{r}}_4^n = \boldsymbol{r}_4 - \sum_{i=1}^3 \boldsymbol{G}_i^*\delta\bar{\boldsymbol{u}}_i^n$

  $* * *$bepaal $\qquad \delta\tilde{\boldsymbol{y}} = \boldsymbol{P}^{-1}\bar{\boldsymbol{r}}^n * **$
  $\boldsymbol{Q}_i\delta\tilde{\boldsymbol{u}}_i = \bar{\boldsymbol{r}}_i^n \qquad i = 1, 2, 3$
  $\boldsymbol{R}\delta\tilde{\boldsymbol{p}} = \bar{\boldsymbol{r}}_4^n - \sum_{i=1}^3 \boldsymbol{G}_i^*\delta\boldsymbol{u}_i$

  $* * *$bepaal $\qquad \delta\boldsymbol{y} = \boldsymbol{C}\delta\tilde{\boldsymbol{y}} * **$
  $\delta\boldsymbol{u}_i = \delta\tilde{\boldsymbol{u}}_i - \boldsymbol{D}_i^{-1}\boldsymbol{G}_i\delta\tilde{\boldsymbol{p}}$
  $\delta\boldsymbol{p} = \delta\tilde{\boldsymbol{p}}$

  $* * *$bepaal $\qquad \boldsymbol{y}^{n+1} = \boldsymbol{y}^n + \delta\boldsymbol{y} * **$
  $\boldsymbol{u}_i^{n+1} = \boldsymbol{u}_i^n + \delta\boldsymbol{u}_i$
  $\boldsymbol{p}^{n+1} = \boldsymbol{p}^n + \delta\boldsymbol{p}$
end for

Opmerkingen

1. Binnen een SIMPLE/SIMPLER iteratie worden de lineaire stelsels van de vorm
$$\boldsymbol{K}\boldsymbol{x}^n = \boldsymbol{s} \tag{3.65}$$
opgelost voor de snelheidscomponenten $\boldsymbol{u}_i$, de druk $\boldsymbol{p}$ en ook voor de andere variabelen zoals de enthalpie $\boldsymbol{h}$, de kinetische energie $\boldsymbol{k}$ en dis-

sipatie van de energie $\boldsymbol{\epsilon}$. De matrix $\boldsymbol{K}$ en de bronvektor $\boldsymbol{s}$ hangen van de snelheidscomponenten af. Bij iedere SIMPLE/SIMPLER iteratie worden $\boldsymbol{K}$ en $\boldsymbol{s}$ geconstrueerd door de benadering van de snelheidscomponenten uit de voorige iteratie te gebruiken. We schrijven (3.65) daarom als volgt:

$$\boldsymbol{K}^{(n)}\boldsymbol{x}^n = \boldsymbol{s}^{(n)} \tag{3.66}$$

2. In WISH3D is de matrix $\boldsymbol{R}$ (de druk-matrix) symmetrisch positief semi-definiet ($\boldsymbol{R}$ is singulier). Dit houdt in dat $\boldsymbol{R}$ één eigenwaarde heeft die gelijk is aan nul. Bij deze eigenwaarde hoort de eigenvektor $\boldsymbol{v} = (\boldsymbol{1}, \boldsymbol{1}, ..., \boldsymbol{1})^{\boldsymbol{T}}$. De oorzak hiervan is dat in WISH3D de uitstroomrand ook als Dirichlet randvoorwaarde behandeld. De singulariteit van $\boldsymbol{R}$ kan problemen opleveren wanneer we de geconjugeerde gradiënten methode gebruiken voor het oplossen van de druk. Om dit probleem te ondervangen kunnen we één van de volgende methoden gebruiken:
   methode 1 : we gebruiken in plaats van de bronterm $\boldsymbol{b_4}$ de bronterm $\tilde{\boldsymbol{b}}_{\boldsymbol{4}}$, die verkregen is door de projectie van $\boldsymbol{b_4}$ op de eigenvektor $\boldsymbol{v}$:

$$\tilde{\boldsymbol{b}}_{\boldsymbol{4}} = \boldsymbol{b_4} - \frac{\boldsymbol{b_4^T v}}{\|\boldsymbol{v}\|_2}\boldsymbol{v} \quad . \tag{3.67}$$

   methode 2 : we kiezen voor de druk een referentiewaarde in een roosterpunt $(\boldsymbol{i_0}, \boldsymbol{j_0}, \boldsymbol{k_0})$ zodat we in dit punt de druk niet behoefen uit te rekenen (voor de motivatie zie [1, bladzijde 131]). In dit geval krijgen we een druk-matrix, die de de eigenwaarde nul niet heeft.

3. In WISH3D wordt meestal SIMPLER gebruikt. De reden hierachter is dat SIMPLER een goede convergentie blijkt te geven vegeleken met SIMPLE. Dit punt wordt uitgebreid gemotiveerd door Patenkar [1, blz. 131,132 & 134].

Stopcriterium

Voordat een SIMPLE/SIMPLER iteratie begint wordt er eerst gecontroleerd of alle variabelen aan het volgende stopcriterium voldoen:

$$\|\boldsymbol{r}^{\boldsymbol{n}}\|_1 = \|\boldsymbol{s}^{(n)} - \boldsymbol{Q}^{(n)}\boldsymbol{x}^{\boldsymbol{n}}\|_1 \leq \boldsymbol{\epsilon} \quad , \tag{3.68}$$

met $\boldsymbol{0} < \boldsymbol{\epsilon} < \boldsymbol{1}$.

Hoofdstuk 4

Test problemen

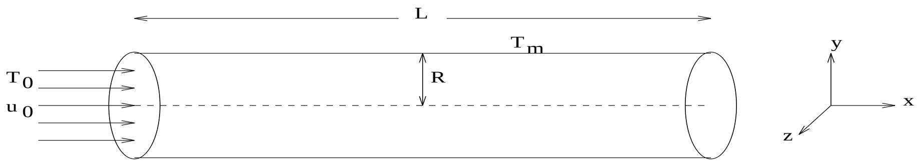
4.1 Inleiding

In dit hoofdstuk zullen we de solvers toepassen die in hoofdstuk 3 besproken zijn. We starten met een eenvoudige pijpstroming en vergelijken het aantal SIMPLER iteraties voor de verschillende keuzen van de oplosmethoden. Daarna zijn er experimenten uitgevoerd aan een 2D glassmeltoven. Hierbij zullen we naast het beschouwen van het aantal SIMPLER iteraties ook aandacht besteden aan de totale rekentijd, die een een experiment kost.

4.2 Pijpstroming

4.2.1 Probleemomschrijving

Beschouw een stationaire luchtstroming in de as-richting van een ronde rechte pijp met lengte L en straal R. De geometrie is geschetst in Figuur 4.1
.



Figuur 4.1: stroming door een rechte ronde pijp

De symbolen in Figuur 4.1 en andere materiële eigenschappen van de stroming zijn in Tabel 4.1 omgeschreven:

symbol	omschrijving	waarde
u_0	instroomsnelheid	1.0 (m/s)
T_0	temperatuur bij de instroomrand	20 ^{o}C
T_m	temperatuur van de pijpwand	50 ^{o}C
L	lengte van de pijp	5.0 (m)
R	straal van de pijp	0.1 (m)
μ	dynamische viscositeit van de lucht	1.824 10^{-5} (kg/ms)
ρ	dichtheid van de lucht	1.16 (kg/m^3)

Tabel 4.1: Omschrijving van enkele grootheden van het probleem.

De stroming heeft twee belangrijke eigenschappen:

1. De stroming is turbulent: als we de instroomsnelheid u_0 en de diameter van de pijp $D = 2R$ respectievelijk als een snelheidsschaal en een lengteschaal nemen, dan geldt voor het Reynolds getal:

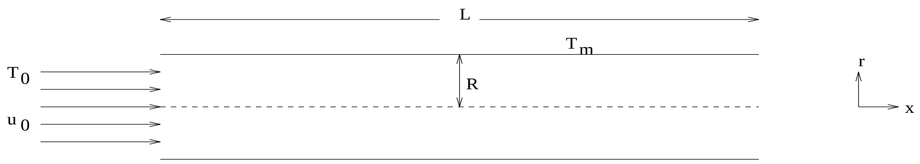
$$R_e = \frac{2R\rho u_0}{\mu} \simeq 12720 > 2400 \quad . \tag{4.1}$$

2. De stroming is compressibel met een laag Mach getal: de dichtheid ρ kan variëren ten gevolge van de verandering van de temperatuur T. Voor het Mach getal geldt:

$$M_a = \frac{u_0}{v} \simeq 0.003 \quad , \tag{4.2}$$

waarbij v de geluidssnelheid voorstelt.

Aangezien de stroming rotatie symmetrisch is, kunnen we cylinder coördinaten in plaats van cartesische coördinaten gebruiken. In dit geval kunnen we volstaan met het beschouwen van het bovenste gedeelte van de pijp zoals geschetst is in Figuur 4.2 .



Figuur 4.2: pijpstroming met cylinder coördinaten

We noteren de snelheid in de axiale richting met u en in de radiale richting met v. We merken op dat het probleem gereduceerd is van een driedimensionaal naar een twee-dimensionaal probleem.
De stroming wordt beschreven door de volgende vergelijkingen en randvoorwaarden:

Continuïteitsvergelijking

$$\frac{\partial}{\partial x}(\rho u)+\frac{1}{r}\frac{\partial}{\partial r}(r\rho v)=0 \quad . \tag{4.3}$$

Convectie-diffusie vergelijking

$$\frac{\partial}{\partial x}(\rho u\phi)+\frac{1}{r}\frac{\partial}{\partial r}(r\rho v\phi)=\frac{\partial}{\partial x}(\Gamma_\phi\frac{\partial\phi}{\partial x})+\frac{1}{r}\frac{\partial}{\partial r}(r\Gamma_\phi\frac{\partial\phi}{\partial r}) \quad r, \tag{4.4}$$

waarbij de scalaire variabele ϕ de snelheidscomponeneten u en v, de turbulente kinetische energie k, de dissipatie ervan ϵ en de enthalpie h voorstelt.

Randvoorwaarden

De randvoorwaarden van het gegeven probleem zijn in Tabel 4.2 opgesomd:

variabele	instroomrand	uitstroomrand	pijpwand	symmetrie-as
snelheid u	$u=u_0$	$\frac{\partial u}{\partial x}=0$	$u=0$	$\frac{\partial u}{\partial r}=0$
snelheid v	$v=0$	$v=0$	$v=0$	$v=0$
temperatuur T	$T=20$	$\frac{\partial T}{\partial x}=0$	$T=50$	$\frac{\partial T}{\partial r}=0$
kinetische energie k	$k=0.0054$	$k=0$	$k=0$	$\frac{\partial k}{\partial r}=0$
dissipatie ϵ	$\epsilon=0.0217$	$\epsilon=0$	$\epsilon=0$	$\frac{\partial \epsilon}{\partial r}=0$

Tabel 4.2: Randvoorwaarden.

4.2.2 Numerieke oplossing

We lossen het gegeven probleem numeriek op zoals aangegeven in Tabel 4.3:

Rekenrooster	(25×10)-rooster
Discretisatie methode	eindige volume methode & hybride schema
Oplosmethode	SIMPLER , zie paragraaf 3.5
Lineaire solvers	zie resultaten

Tabel 4.3: Numerieke methode voor het oplossen van het probleem.

De pijpstroming, die we hier bekijken, is van het type convectie gedomineerde stromingen. Dit heeft als gevolg $(\boldsymbol{Pe})_{e} > \mathbf{2}$, waarbij $(\boldsymbol{Pe})_{e}$ het maas-Péclet getal is voor de x-richting. Op basis van Toepassing 1 uit Hoofdstuk 3 kunnen we de lineaire stelsels van de vorm (4.3) exact oplossen met de Linezxy solver . Dit houdt in dat we in alle testen de variabelen $\boldsymbol{u}$, $\boldsymbol{v}$, $\boldsymbol{k}$, $\boldsymbol{\epsilon}$ en $\boldsymbol{h}$ met de Linezxy solver zullen oplossen . Aangezien de druk-matrix geen convectieve termen bevat, voldoen de Line-TDMA solvers niet aan de beweringen uit Hoofdstuk 3. Dat wil zeggen dat een Line-TDMA variant geen exacte solver zal zijn voor de druk. Om te bekijken welke solver het meest geschikt is voor de convergentie van SIMPLER, zullen we dan de druk iteratief oplossen met verschillende solvers.

4.2.3 Resultaten

We zullen de testen in twee categoriën verdelen. Bij de eerste categorie wordt de druk met een WISH3D solver opgelost. Bij de tweede categorie wordt de CG methode gebruikt, waarbij de bronterm geprojecteerd wordt op de eigenvektor $\boldsymbol{v} = (\mathbf{1}, \mathbf{1}, ..., \mathbf{1})^{\boldsymbol{T}}$ (zie paragraaf 3.5). Bij het vergelijken tussen de testen zullen we de rekentijden niet beschouwen aangezien deze erg kort zijn. In alle testen worden de volgende relaxatiefaktoren gebruikt voor de variabelen. Hierbij gaat het om de relaxatiefaktoren van de bestaande WISH3D solvers (zie paragraaf 3.2)

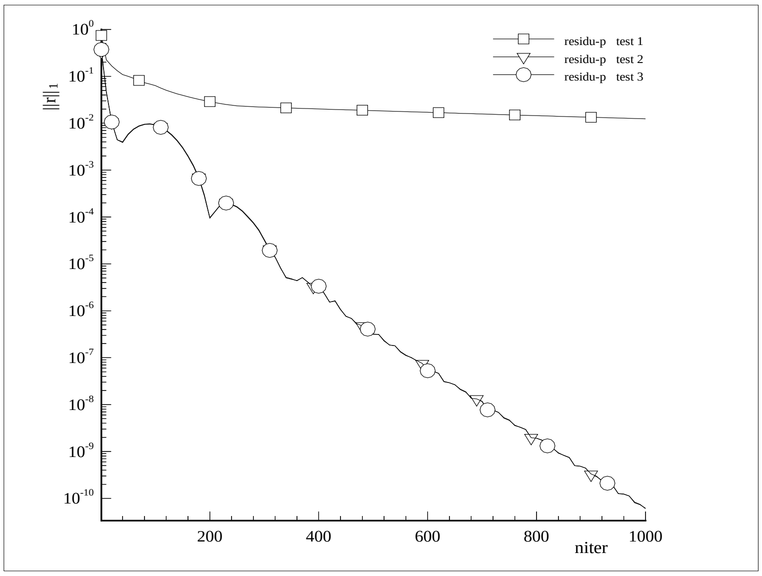
variabele	$\boldsymbol{u}$	$\boldsymbol{v}$	$\boldsymbol{p}$	$\boldsymbol{k}$	$\boldsymbol{\epsilon}$	$\boldsymbol{h}$
relaxatiefaktor	0.8	0.8	1.0	0.5	0.5	0.980

De testen van de eerste categorie zijn in Tabe 4.4 gegeven.

Test	solver	binnen iteraties (sweeps)	rekentijd (in sec)
Test 1	Linzyx	1 sweep in de y-richting 0 sweep in de tegengestelde y-richting 5 sweep in de z-richting 0 sweep in de tegengestelde z-richting	40.8
Test 2	Planez	1 sweep in de z-richting 1 sweep in de tegengestelde z-richting	40.6
Test 3	Planez	2 sweep in de z-richting 0 sweep in de tegengestelde z-richting	42.2

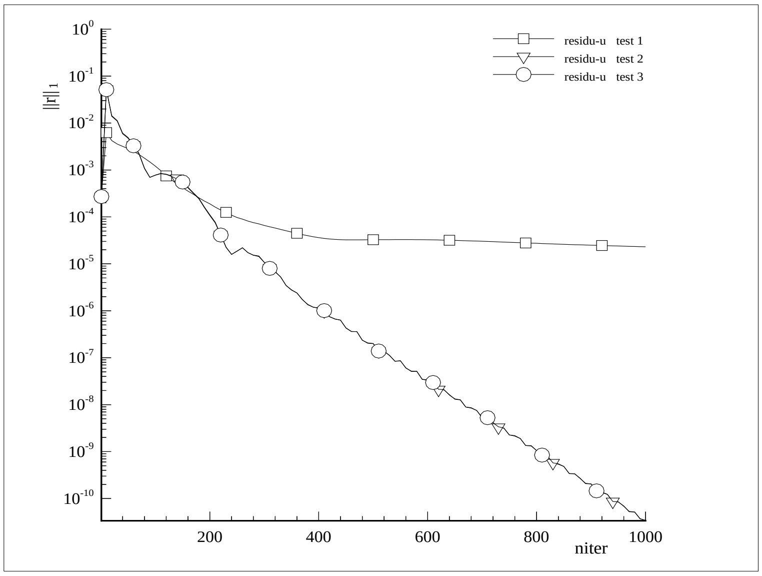
Tabel 4.4: Testen van de eerste categerie.

Om te kunnen vergelijken tussen deze testen hebben we de residunorm $\|\boldsymbol{r}\|_{\mathbf{1}}$ van de druk uitgezet tegen het aantal SIMPLER iteraties. Zie Figuur 4.3.

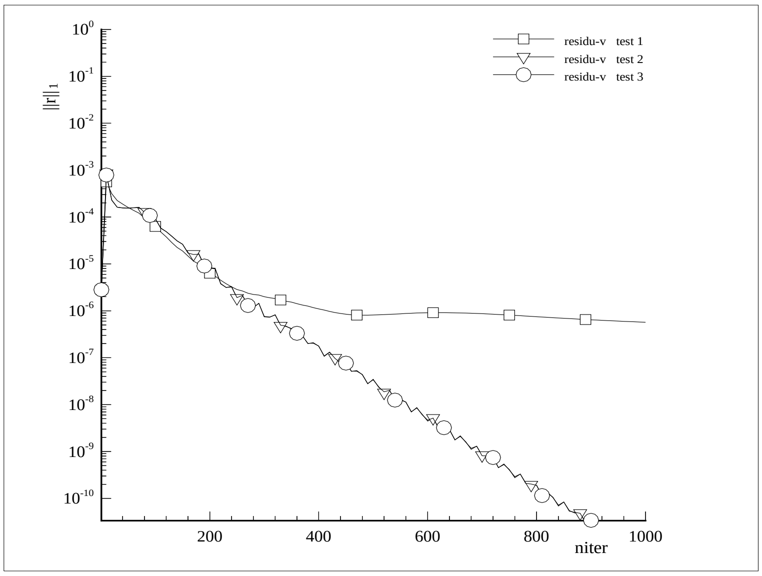


Figuur 4.3: De convergentie van de druk geresulteerd uit de testen van de eerste categorie.

Hieruit kunnen we zien dat Test 2 en Test 3 een betere convergentie opleveren dan Test 1. Dat wil zeggen dat, voor het oplossen van de druk, de Planez solver geschikter is dan de linezyx solver. Deze conculsie is bevestigd ook door testen, die we niet in dit verslag hebben opgenomen. Een andere conclusie is dat de convergentie niet beïnvloed is door de keuze van de sweeps in de z-richting (zie Tabel 4.5 en Tabel 4.6). Uit de testen is ook gebleken dat de convergentie bij de snelheidscomponenten beïnvloed wordt door het convergentiegedrag van de druk. Zie Figuur 4.4 en Figuur 4.5.



Figuur 4.4: De convergentie van de snelheidscomponent $\boldsymbol{u}$ geresulteerd uit de testen van de eerste categorie.



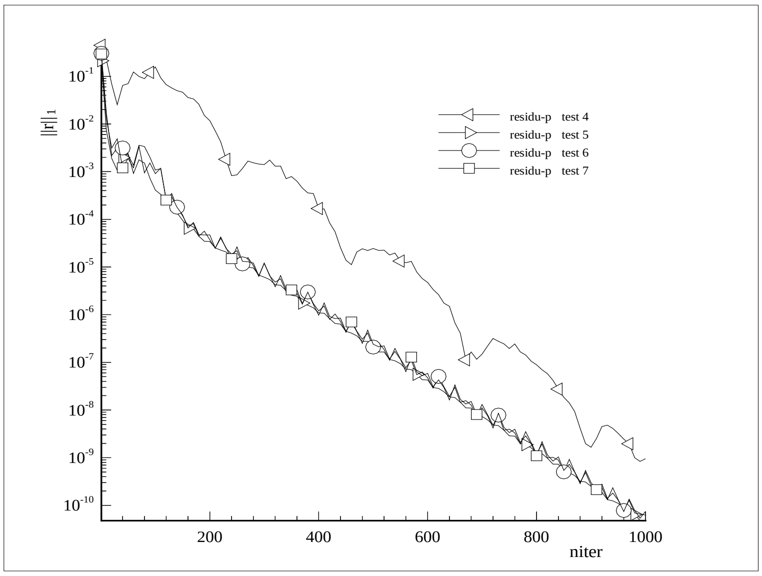
Figuur 4.5: De convergentie van de snelheidscomponent $\boldsymbol{v}$ geresulteerd uit de testen van de eerste categorie.

De testen van de tweede categorie zijn gegeven in Tabel 4.5

Testen	solver	preconditionering	binnen iteraties	rekentijd (in sec)
Test 4	CG	Gauss-Jacobi (diagonaalschaling)	10	45.0
Test 5	CG	RILU relaxatiefaktor=0.975	10	55.3
Test 6	CG	RILU relaxatiefaktor=0.975	5	47.6
Test 7	CG	standaard ILU (Planez)	5	49.4

Tabel 4.5: Testen van de tweede categorie

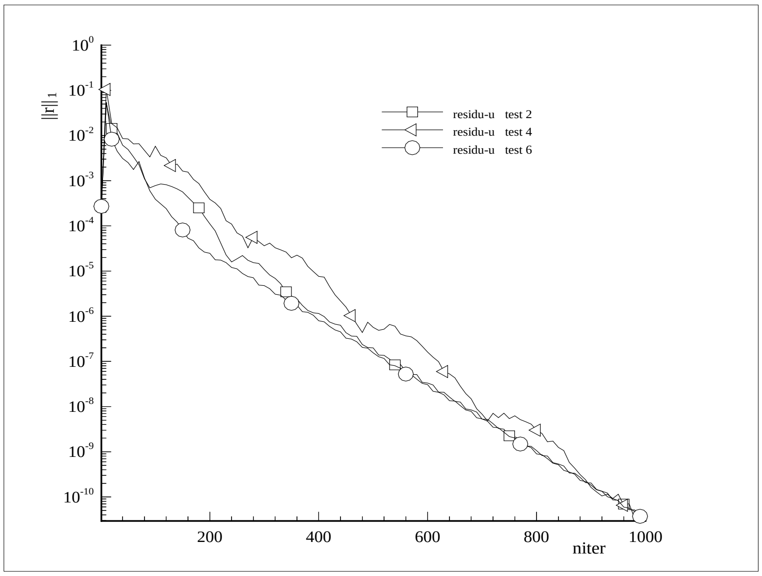
Het convergentiegedrag van de druk, die verkregen is door de testen van de tweede categorie, is te zien in Figuur 4.6.



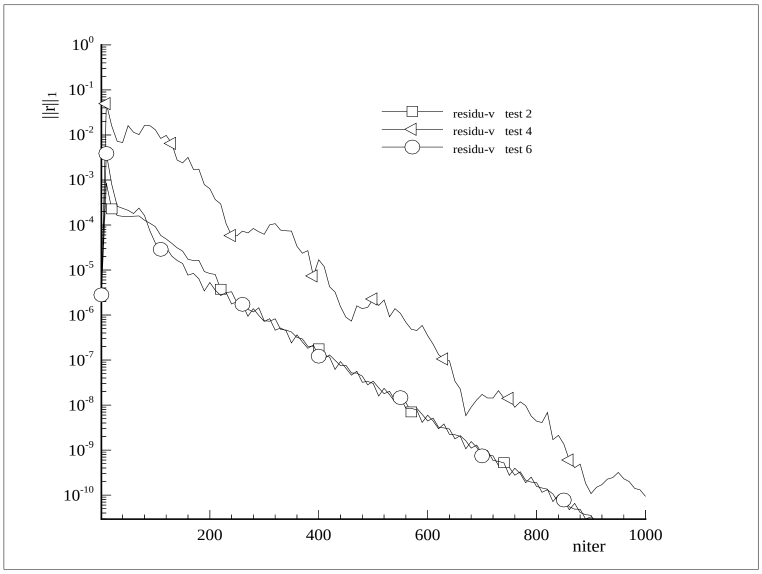
Figuur 4.6: De convergentie van de druk geresulteerd uit de testen van de tweede categorie.

Hieruit blijkt dat de convergentie verkregen door het preconditioneren van de CG methode met standaard ILU en RILU een betere convergentiegedrag vertoont ten opzichte van de Gauss-Jacobi preconditionering. Maar deze laatste is aantrekkelijk wanneer men de rekentijd wil reduceren door de CG methode te parallelliseren. We zien ook dat we bijna dezelfde convergentie verkrijgen uit Test 5 en Test 6. Dat wil zeggen dat 5 en 10 binnen iteraties van de CG solver hetzelfde resultaat opleveren.

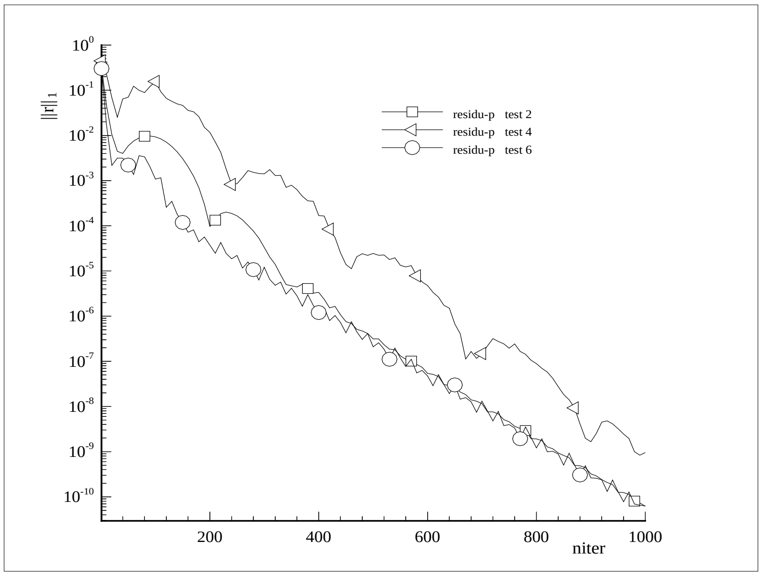
Als laatste hebben we Test 2 uit de eerste categorie vergeleken met Test 4 en Test 6 uit de tweede categorie voor de snelheidscomponenten $\boldsymbol{u}$ en $\boldsymbol{v}$, de druk $\boldsymbol{p}$ en de enthalpie $\boldsymbol{h}$. Zie Figuur 4.7 tot en met Figuur 10.



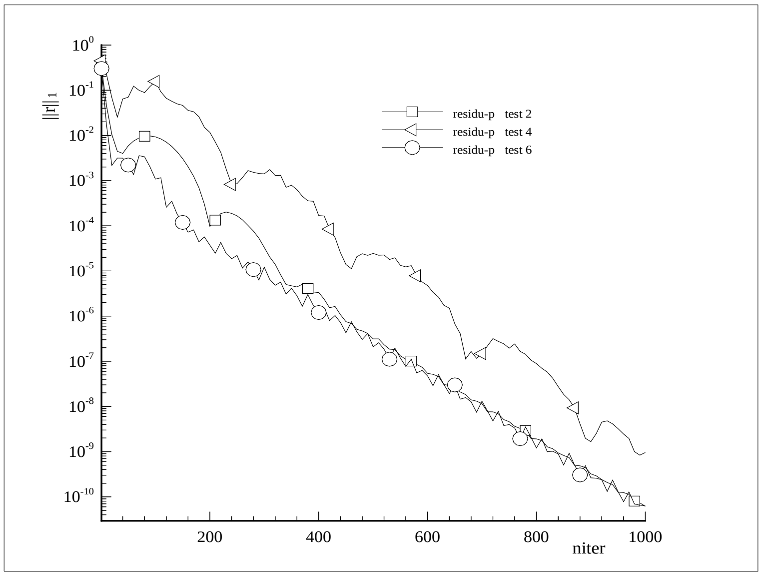
Figuur 4.7: De convergentie van de snelheidscomponent $\boldsymbol{u}$ geresulteerd uit Test 2, Test 4 en Test 6.



Figuur 4.8: De convergentie van de snelheidscomponent $\boldsymbol{v}$ geresulteerd uit Test 2, Test 4 en Test 6.



Figuur 4.9: De convergentie van de druk $\boldsymbol{p}$ geresulteerd uit Test 2, Test 4 en Test 6.



Figuur 4.10: De convergentie van de enthalpie $\boldsymbol{h}$ geresulteerd uit Test 2, Test 4 en Test 6.

We zien dat het oplossen van de druk met de Planez solver (Test 2) en de CG solver (Test 6) bijna dezelfde convergentie opleveren. Het schijnt dat SIMPLER genoeg heeft aan een klein aantal binnen iteraties (2 sweeps) om een redelijke convergentie te verkrijgen. Dit verklaart waarschijnlijk het feit dat de CG solver niet beter doet dan de Planez solver.

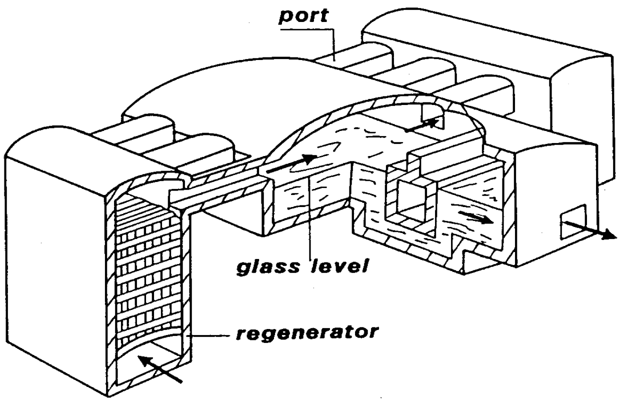
4.2.4 Conclusie

In het testprobleem hebben we gezien dat, binnen een SIMPLER iteratie, de stelsels van alle variabelen behalve de druk exact opgelost kunnen worden met de Linzxy solver. Maar uit de testen is gebleken dat de convergentie van de druk bepalend is voor de convergentie bij de andere variabelen. Hieruit concluderen we dat het goed oplossen van de druk een belangrijke faktor is bij het uitrekenen van problemen zoals de pijpstroming, die we hier hebben besproken. Maar aan de andere kant hebben we gemerkt dat we niet konden profeteren van de CG solver aangezien een klein aantal binnen iteraties van de Planez solver genoeg blijkt te zijn om een redelijke convergentie van SIMPLER te verkrijgen. Dit feit wordt ook bevestigd door de eerdere opmerking dat 5 en 10 binnen iteraties van de CG solver dezelfde convergentie oplgeleverd hebben.

4.3 Regenerative cross-fired glassmeltoven

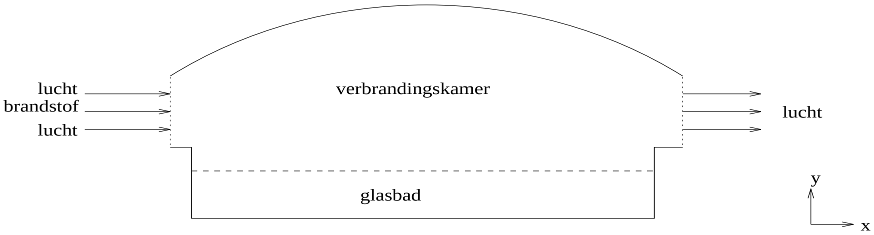
4.3.1 Probleemomschrijving

In deze paragraaf zullen we een aantal testen uitvoeren op een (Regenerative cross-fired) glassmeltoven zoals geschetst is in Figuur 4.11.



Figuur 4.11: Regenerative cross-fired glassmeltoven.

We zullen ons beperken tot een tweedimensionale geometrie. Zie Figuur 4.12.



Figuur 4.12: Tweedimensionale geometrie van een Regenerative cross-fired glassmeltoven.

Voor het simuleren van de oven zijn de volgende onderdelen belangrijk:

1. het glasbad met vloeibare glas.
2. de verbrandingskamer die zich boven het glasbad bevindt. De kamer heeft een maximale hoogte van 2.405 (m), een lengte van 7.270 (m) en een breedte van 3.300 (m).
3. de inlaat voor de brandstof en de lucht (zuurstof) en

4. de uitlaat voor de rookgassen.

Voor details over dit soort ovens verwijzen naar [2] en [9].
De stroming en de verbranding binnen de oven worden beschreven door de modellen besproken in hoofdstuk 2. Hierbij worden de volgende variabelen uitgerekend:

variabele	omschrijving
$\boldsymbol{u}$	snelheidscomponent in de x-richting
$\boldsymbol{v}$	snelheidscomponent in de y-richting
$\boldsymbol{p}$	druk
$\boldsymbol{h}$ ($\boldsymbol{T}$)	enthalpie (temperatuur)
$\boldsymbol{k}$	kinetische enrgie
$\boldsymbol{\epsilon}$	dissipatie van energie
$\boldsymbol{f}$	mengselfractie (zie paragraaf 1.3)

Tabel 4.6: Variabelen van het probleem

De randvoorwaarden, die bij deze variabelen horen, worden beschreven in Tabel 4.7 .

variabele	glasbadoppervlakte	wanden	instroomrand	uitstroomrand
$\boldsymbol{u}$	$\boldsymbol{u = 0}$	$\boldsymbol{u = 0}$	profiel	$\boldsymbol{\frac{\partial u}{\partial n} = 0}$
$\boldsymbol{u}$	$\boldsymbol{v = 0}$	$\boldsymbol{v = 0}$	profiel	$\boldsymbol{\frac{\partial v}{\partial n} = 0}$
$\boldsymbol{p}$	$\boldsymbol{\frac{\partial p}{\partial n} = 0}$	$\boldsymbol{\frac{\partial p}{\partial n} = 0}$	$\boldsymbol{\frac{\partial p}{\partial n} = 0}$	$\boldsymbol{\frac{\partial p}{\partial n} = 0}$
$\boldsymbol{h}$	$\boldsymbol{T = T_{glas}}$	wandfunctie	profiel	$\boldsymbol{\frac{\partial T}{\partial n} = 0}$
$\boldsymbol{k}$	wandfuncties	wandfuncties	profiel	$\boldsymbol{\frac{\partial k}{\partial n} = 0}$
$\boldsymbol{\epsilon}$	wandfuncties	wandfuncties	profiel	$\boldsymbol{\frac{\partial \epsilon}{\partial n} = 0}$
$\boldsymbol{f}$	$\boldsymbol{\frac{\partial f}{\partial n} = 0}$	$\boldsymbol{\frac{\partial f}{\partial n} = 0}$	profiel	$\boldsymbol{\frac{\partial f}{\partial n} = 0}$

Tabel 4.7: Randvoorwaarden.

4.3.2 Numerieke oplossing

We lossen het gegeven probleem op zoals aangegeven in Tabel 4.8.

Rekenrooster	(116 $\times$ 57)-rooster
Discretisatie methode	eindige volume & hybride schema
Oplosmethode	SIMPLER
Lineaire solvers	zie resultaten

Tabel 4.8: Numerieke methode voor het oplossen van het probleem.

4.3.3 Resultaten

Gemotiveerd door de testen op de pijpstroming, zullen we ons bij het huidige testprobleem richten op het uitzoeken welke solvers een goede convergentie geven met zo weinig mogelijk rekentijd bij het oplossen van de druk en de snelheidscomponenten. Voor de andere variabelen blijkt Linezyx een geschikte solver te zijn.
Analoog aan de pijpstroming zullen we de testen verdelen in twee categoriën. Bij de eerste categorie wordt de druk met de Planez solver opgelost. Bij de tweede categorie wordt de CG methode gebruikt. Het is gebleken dat de projectie van de bronterm op de eigenvektor $\boldsymbol{v} = \mathbf{(1, 1, ..., 1)^T}$ een slechte convergentie oplevert voor bijvoorbeeld de snelheidscomponenten. Om dit te kunnen motiveren hebben we twee identieke testen (Test 7 en Test 9) met elkaar vergeleken (zie paragraaf 4.3.3). Het enige verschil tussen de twee testen is dat bij Test 9 de projectie van de bronterm op de eigenvektor $\boldsymbol{v}$ gebruikt is (dit zullen we de projectie-methode noemen) terwijl bij Test 7 de volgende methode (referentie-methode):
De rij en de kolom, die corresponderen met het roosterpunt waarin een referentiewaarde van de druk gegeven is, worden uit de druk-matrix verwijderd (zie Opmerkingen in paragraaf 3.5).
Als eerste hebben we een test uitgevoerd met 1000 SIMPLER iteraties, waarbij de Planez solver gebruikt is voor het oplossen van alle variabelen. Dit heeft een rekentijd van 2700 seconden gekost. De verkregen benaderingen voor de variabelen worden als beginschattingen gebruikt bij het opstarten van de testen die hierna volgen. De reden van deze aanpak is dat de Krylov solvers hiermee een betere convergentie blijken op te leveren en minder rekentijd te kosten dan wanneer het rekenproces met willekeurige beginschattingen start. Dat deze aanpak minder rekentijd kost, volgt uit het feit dat de Krylov solvers over het algemeen meer rekenwerk verrichten dan de WISH3D solvers.
Tenzij anders vermeld, worden in alle testen de volgende relaxatiefaktoren gebruikt voor de variabelen. Hierbij gaat het om de relaxatiefaktoren van de bestaande WISH3D solvers. (zie paragraaf 3.2)

variabele	$\boldsymbol{u}$	$\boldsymbol{v}$	$\boldsymbol{p}$	$\boldsymbol{k}$	$\boldsymbol{\epsilon}$	$\boldsymbol{h}$	$\boldsymbol{f}$
relaxatiefaktor	0.6	0.6	1.0	0.7	0.7	0.8	1.0

De testen van de eerste categorie zijn in Tabel 4.9 tot en met Tabel 4.13 gegeven.

variabele	solver	binnen iteraties (sweeps)
$\boldsymbol{u}$	Planez	3 sweep in de z-richting 0 sweep in de tegengestelde z-richting
$\boldsymbol{v}$	Planez	3 sweep in de z-richting 0 sweep in de tegengestelde z-richting
$\boldsymbol{p}$	Planez	1 sweep in de z-richting 1 sweep in de tegengestelde z-richting
$\boldsymbol{k}$,$\boldsymbol{\epsilon}$,$\boldsymbol{h}$,$\boldsymbol{f}$	Linezxy	1 sweep in de x-richting 0 sweep in de tegengestelde x-richting 1 sweep in de z-richting 0 sweep in de tegengestelde z-richting

rekentijd = 18589 sec

Tabel 4.9: Test 1 uit de eerste categorie.

De testen 2, 3 & 4 uit de eerste categorie zijn identiek aan Test 1 behalve dat de snelheidscomponenten anders opgelost zijn.

variabele	solver	binnen iteraties (sweeps)
$\boldsymbol{u}$	GCR	3 binnen iteraties
$\boldsymbol{v}$	GCR	3 binnen iteraties

rekentijd = 19133 sec

Tabel 4.10: Test 2 uit de eerste categorie. Hierbij wordt de GCR solver gepreconditioneerd met Linezxy.

variabele	solver	binnen iteraties (sweeps)
$\boldsymbol{u}$	Bi-CGSTAB	3 binnen iteraties
$\boldsymbol{v}$	Bi-CGSTAB	3 binnen iteraties

rekentijd = 20356 sec

Tabel 4.11: Test 3 uit de eerste categorie. Hierbij wordt de Bi-CGSTAB solver gepreconditioneerd met Linezxy.

variabele	solver	binnen iteraties (sweeps)
$\boldsymbol{u}$	GCR	5 binnen iteraties
$\boldsymbol{v}$	GCR	5 binnen iteraties

rekentijd = 26335 sec

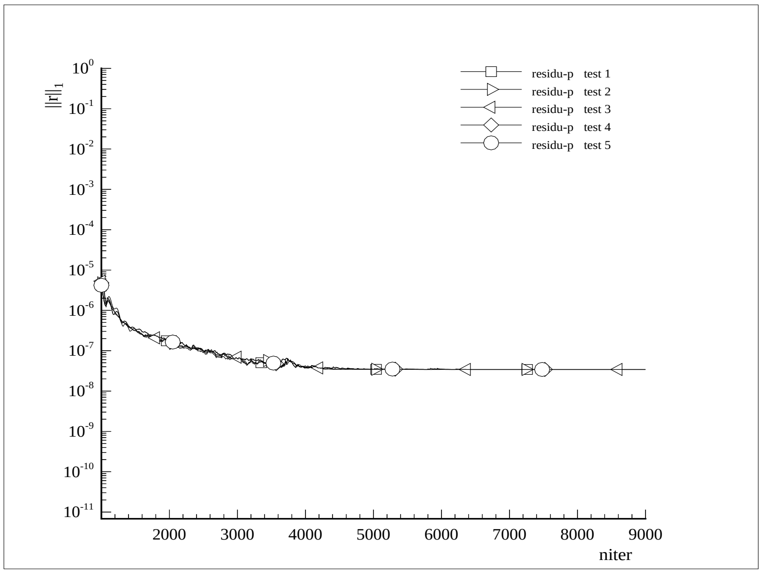
Tabel 4.12: Test 4 uit de eerste categorie. Hierbij wordt de GCR solver gepreconditioneerd met de standaard ILU-preconditionering (Planez).

variabele	solver	binnen iteraties (sweeps)
$\boldsymbol{u}$	Planez	5 sweep in de z-richting 0 sweep in de tegengestelde z-richting
$\boldsymbol{v}$	Planez	5 sweep in de z-richting 0 sweep in de tegengestelde z-richting
$\boldsymbol{p}$	Planez	1 sweep in de z-richting 1 sweep in de tegengestelde z-richting
$\boldsymbol{k}$,$\boldsymbol{\epsilon}$,$\boldsymbol{h}$,$\boldsymbol{f}$	Planez	1 sweep in de z-richting 1 sweep in de tegengestelde z-richting

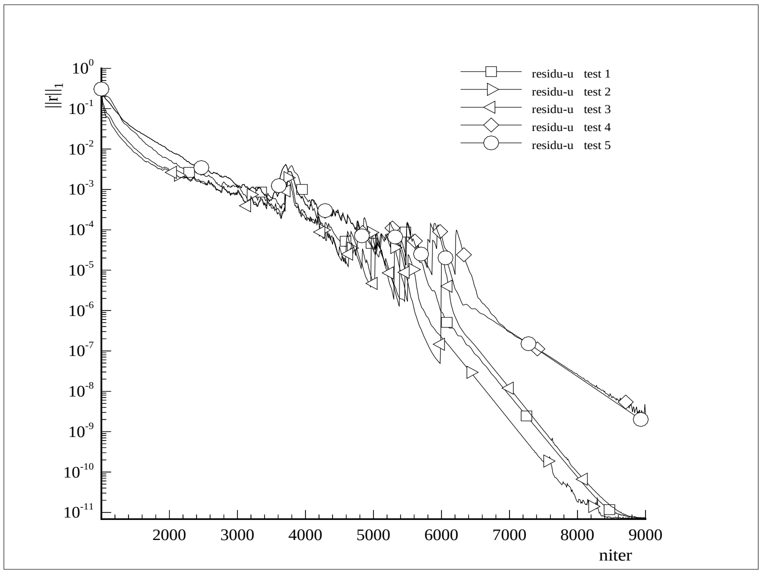
rekentijd = 23046 sec

Tabel 4.13: Test 5 uit de eerste categorie.

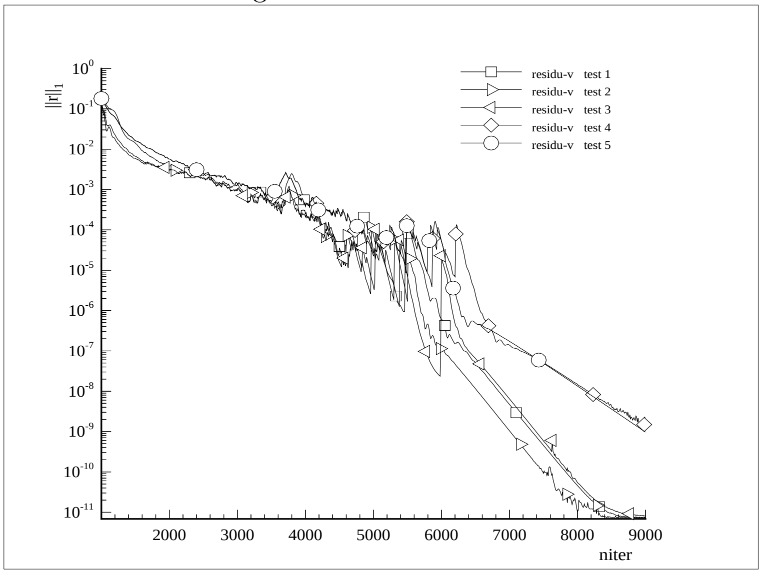
De testen uit de eerste categorie zijn met elkaar vergeleken ten aanzien van de convergentie van de druk en de snelheiscomponenten. Het resultaat hiervan zien we in Figuur 4.13 , Figuur 4.14 en Figuur 4.15.



Figuur 4.13: De convergentie van de druk geresulteerd uit de testen van de eerste categorie.



Figuur 4.14: De convergentie van de snelheidscomponent $\boldsymbol{u}$ geresulteert uit de testen van de eerste categorie.



Figuur 4.15: De convergentie van de snelheidscomponent $\boldsymbol{v}$ geesulteert uit de testen van de eerste categorie.

Hieruit zien we dat het convergentiegedrag van de druk hetzelfde is voor alle testen van de eerste categorie en dat na $\mathbf{4000}$ SIMPLER iteraties krijgt de residunorm $||\boldsymbol{r}||_{\mathbf{1}}$ een constante waarde van orde $\mathbf{10^{-8}}$. Dit blijft voorlopig een onduidelijk verschijnsel. Wat de snelheidscomponenten betreft, blijkt dat Test 1, Test 2 en Test 3 een betere convergentie opleveren. Als we de rekentijden beschouwen, dan is het duidelijk dat de voorkeur naar Test 1 en Test 2 gaat.

Test 6 en Test 7 van de tweede categorie zijn in Tabel 4.14 en Tabel 4.15 gegeven. Deze testen zijn wederom identiek aan Test 1 behalve dat de snelheidscomponenten en de druk anders opgelost zijn. Test 8 is gegeven in Tabel 4.16 .

variabele	solver	binnen iteraties (sweeps)
$\boldsymbol{u}$	GCR	10 binnen iteraties
$\boldsymbol{v}$	GCR	10 binnen iteraties
$\boldsymbol{p}$	CG	50

rekentijd = 35885 sec

Tabel 4.14: Test 6 uit de tweede categorie. Hierbij worden de GCR en CG solver gepreconditioneerd met de Gauss-Jacobi preconditionering (diagonaalschaling).

variabele	solver	binnen iteraties (sweeps)
$\boldsymbol{u}$	Linezxy	1 sweep in de x-richting 0 sweep in de tegengestelde x-richting 1 sweep in de z-richting 0 sweep in de tegengestelde z-richting
$\boldsymbol{v}$	Linezxy	1 sweep in de x-richting 0 sweep in de tegengestelde x-richting 1 sweep in de z-richting 0 sweep in de tegengestelde z-richting
$\boldsymbol{p}$	CG	30 binnen iteraties

rekentijd = 36804 sec

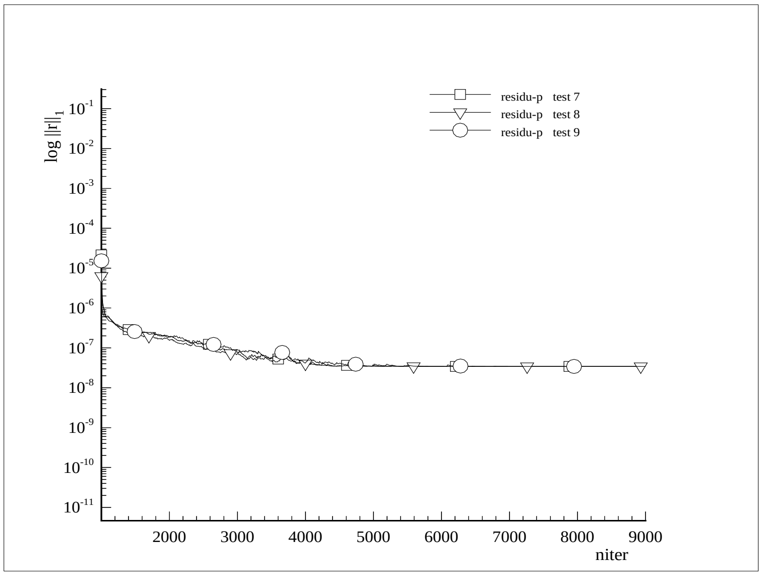
Tabel 4.15: Test 7 uit de tweede categorie. Hierbij wordt de CG solver gepreconditioneerd met RILU met relaxatiefaktor $\boldsymbol{\omega} = \mathbf{0.975}$.

variabele	solver	binnen iteraties (sweeps)
$\boldsymbol{u}$	GCR	10 binnen iteraties
$\boldsymbol{v}$	GCR	10 binnen iteraties
$\boldsymbol{p}$	CG	30 binnen iteraties
$\boldsymbol{k}$	Linezxy	1 sweep in de x-richting 0 sweep in de tegengestelde x-richting 1 sweep in de z-richting 0 sweep in de tegengestelde z-richting
$\boldsymbol{\epsilon}$	Linzxy	1 sweep in de x-richting 0 sweep in de tegengestelde x-richting 1 sweep in de z-richting 0 sweep in de tegengestelde z-richting
$\boldsymbol{h}$	GCR	10 binnen iteraties
$\boldsymbol{f}$	Linzxy	1 sweep in de x-richting 0 sweep in de tegengestelde x-richting 1 sweep in de z-richting 0 sweep in de tegengestelde z-richting

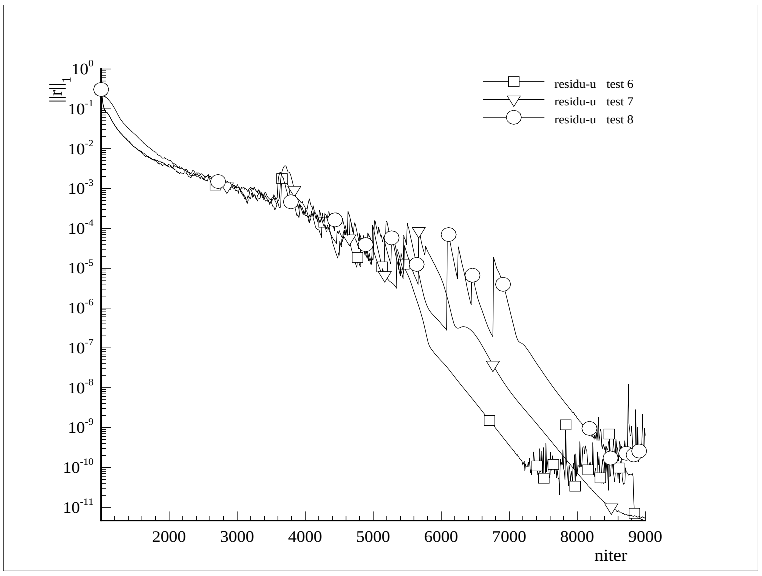
rekentijd = 62578 sec

Tabel 4.16: Test 8 uit de tweede categorie. Hierbij wordt de CG solver gepreconditioneerd met RILU met relaxatiefaktor $\boldsymbol{\omega = 0.975}$. De GCR solver wordt gepreconditioneerd met Linezxy. De relaxatiefaktor voor de snelheidscomponenten is gelijk aan 0.8 .

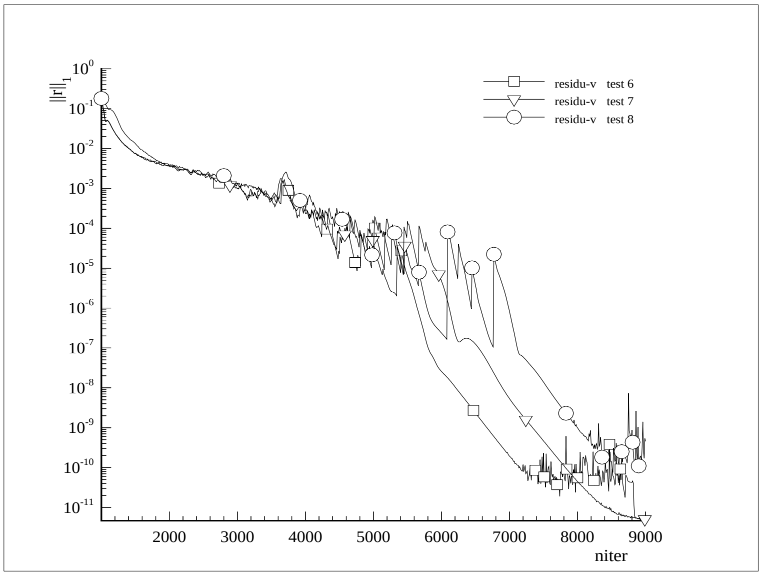
We hebben de testen uit de tweede categorie met elkaar vergeleken aan de hand van Figuur 4.16, Figuur 4.17 en Figuur 4.18 .



Figuur 4.16: De convergentie van de druk gersulteerd uit de testen van de tweede categorie.



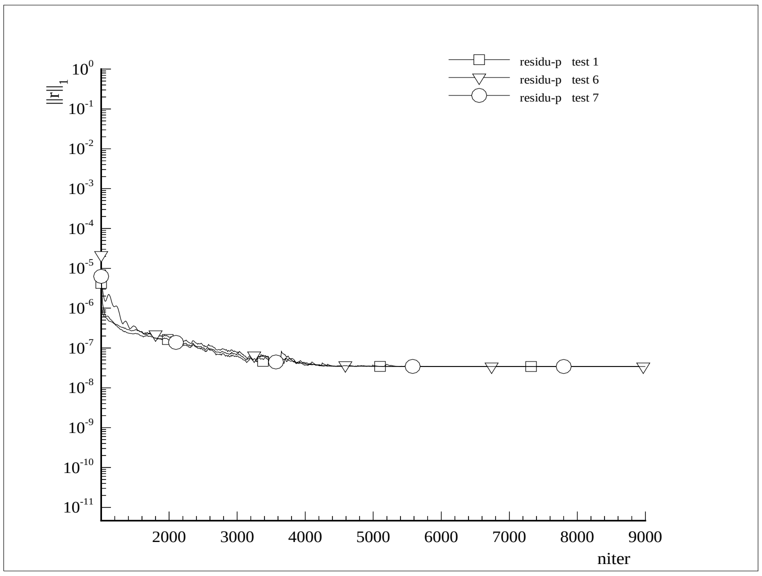
Figuur 4.17: De convergentie van de snelheidscomponent $\boldsymbol{u}$ gersulteerd uit de testen van de tweede categorie.



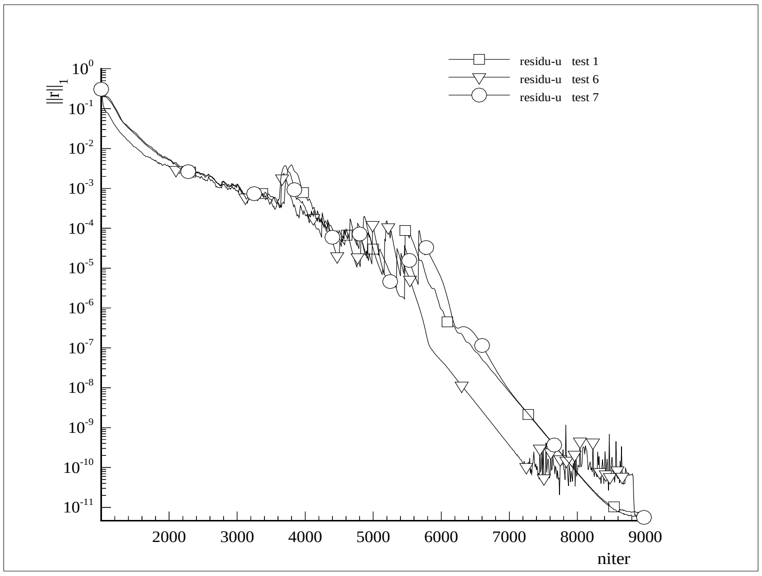
Figuur 4.18: De convergentie van de snelheidscomponent $\boldsymbol{v}$ gersulteerd uit de testen van de tweede categorie.

We zien dat de convergentie van de druk blijft hetzelfde gedrag vertonen als bij de testen van de eerste categorie. Bij de snelheidscomponenten krijgen we wat betere convergentie uit Test 6 en Test 7. De twee testen kosten ook minder rekentijd dan Test 8. Wanneer men de rekentijd wil reduceren door de CG en GCR te parallelliseren, dan is Test 6 voordeliger aangezien we hierin de Gauss-Jacobi preconditionering gebruikt hebben. Het is opmerkelijk om te constateren dat Test 7 een beter resultaat oplevert dan Test 8. Dit ligt waarschijnlijk aan het verschil tussen de gekozen relaxatiefaktoren voor de snelheiscomponenten bij beide testen.

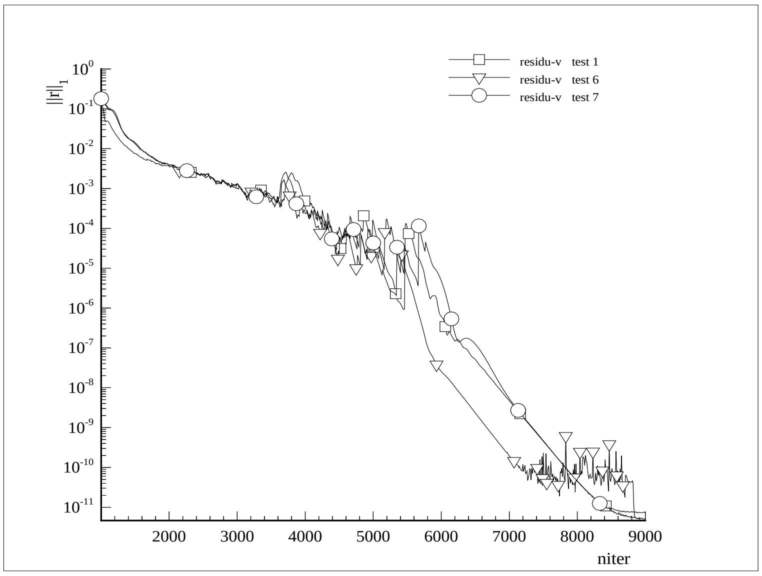
We sluiten het testprobleem met het vergelijken tussen Test 1, Test 6 en Test 7 voor alle variabelen. Zie Figuur 4.19 tot en met Figuur 4.25 . In Figuur 4.26 en Figuur 4.27 worden de snelheid en de temperatuur in het z-vlak van de oven getoond. Hierbij hebben we de data van Test 7 gebruikt.



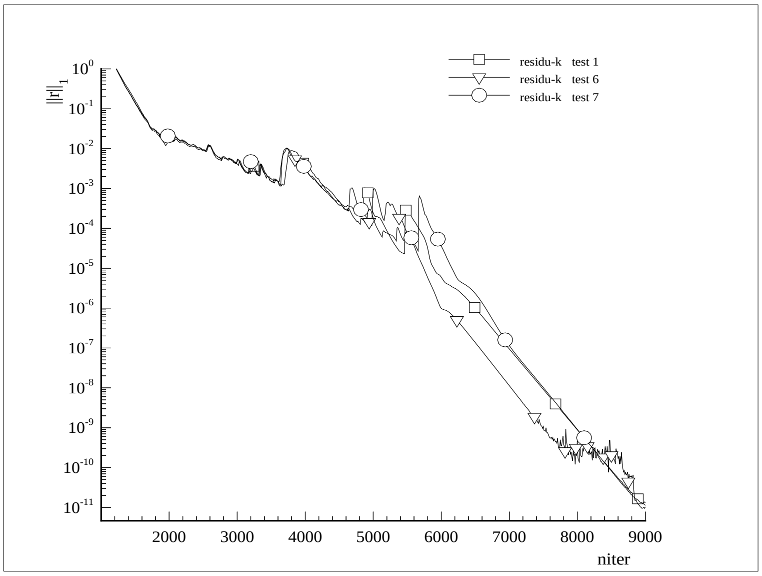
Figuur 4.19: De convergentie van de druk $\boldsymbol{p}$ geresulteerd uit Test 1, Test6 en Test 7.



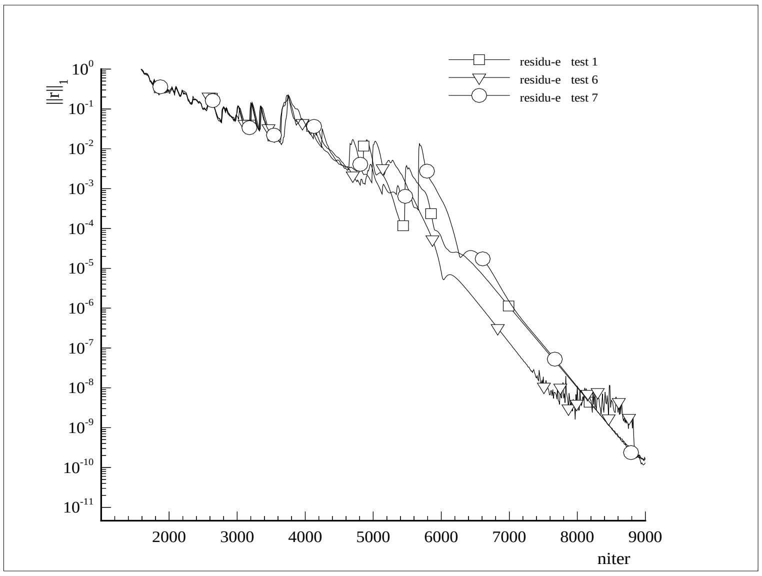
Figuur 4.20: De convergentie van de snelheidscomponent $\boldsymbol{u}$ geresulteerd uit Test 1, Test6 en Test 7.



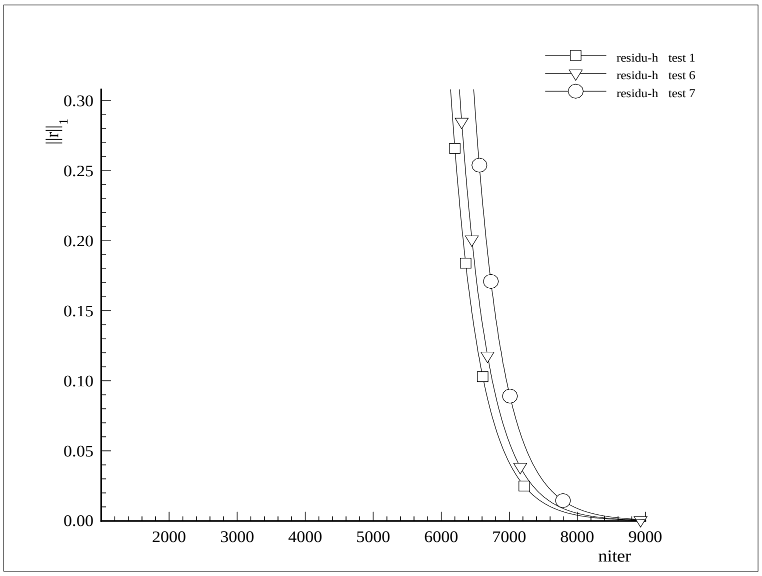
Figuur 4.21: De convergentie van de snelheidscomponent $\boldsymbol{v}$ geresulteerd uit Test 1, Test6 en Test 7.



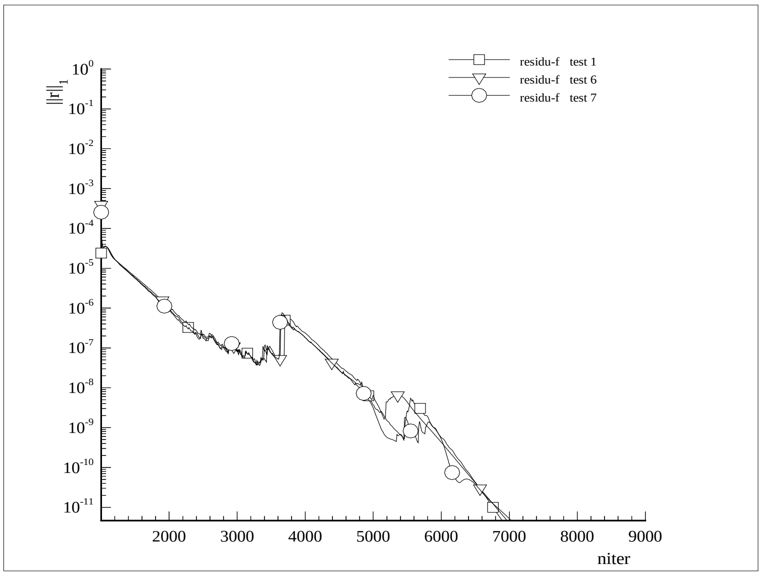
Figuur 4.22: De convergentie van de kinetische energie $\boldsymbol{k}$ geresulteerd uit Test 1, Test6 en Test 7.



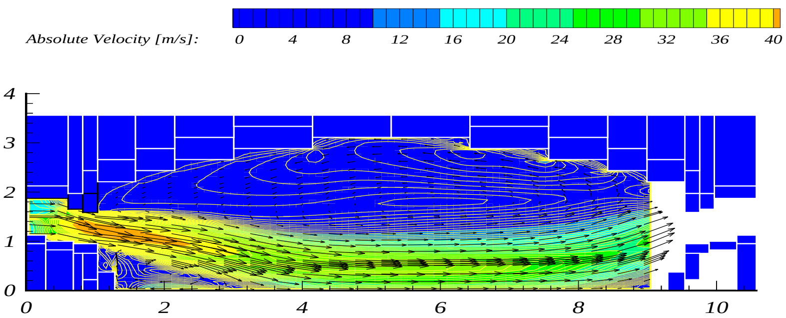
Figuur 4.23: De convergentie van de dissipatie van energie $\boldsymbol{\epsilon}$ geresulteerd uit Test 1, Test6 en Test 7.



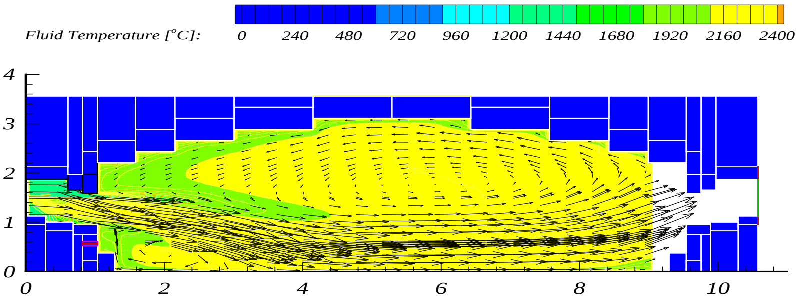
Figuur 4.24: De convergentie van de enthalpie $\boldsymbol{h}$ geresulteerd uit Test 1, Test6 en Test 7.



Figuur 4.25: De convergentie van de mengselfractie $\boldsymbol{f}$ geresulteerd uit Test 1, Test6 en Test 7.



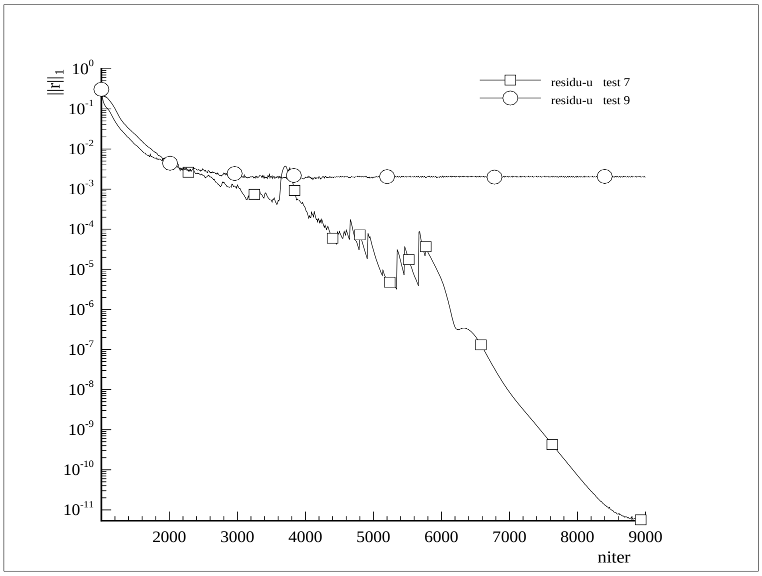
Figuur 4.26: De snelheid in het z-vlak van de cross-fired glassmeltoven.



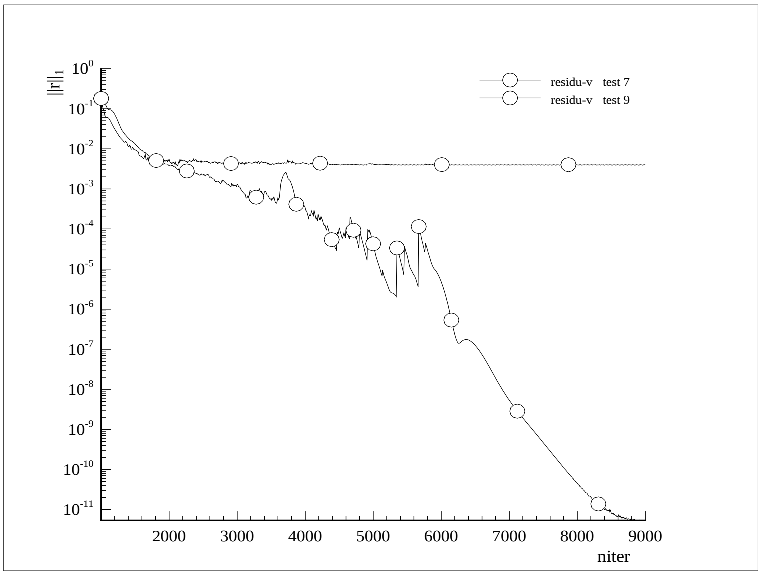
Figuur 4.27: Temperatuur in het z-vlak van de cross-fired glassmeltoven.

4.3.4 Het vergelijken tussen de projectie-methode en de referentie-methode

We hebben de projectie-methode en de referentie-methode met elkaar vergeleken aan de hand van het uitvoeren van Test 7 voor beide methoden. Dat de projectie-methode niet goed werkt, blijkt uit Figuur 4.28 en Figuur 4.29 .



Figuur 4.28: Het vegelijken tussen de projectie-methode en de referentie-methode voor de snelheisdcomponent $\boldsymbol{u}$ aan de hand van Test 7.



Figuur 4.29: Het vegelijken tussen de projectie-methode en de referentie-methode voor de snelheisdcomponent $\boldsymbol{v}$ aan de hand van Test 7.

Hoofdstuk 5

Conclusies en aanbevelingen voor verder onderzoek

5.1 Conclusies

Onderzocht was de convergentie van SIMPLER verkregen door het oplossen van lineaire stelsels met de bestaande WISH3D solvers en de Krylov solvers. Hierbij is de aandacht besteed aan de volgende onderwerpen:

1. Het beschrijven van de WISH3D solvers aan de hand van lineaire algebra. Hieruit zijn twee belangrijke conclusies getrokken:

    - Het is gebleken dat binnen een SIMPLER iteratie een Line-TDMA variant een exacte solver is voor twee-dimensionale stromingen die convectie dominant zijn. Zie subparagraaf 3.2.1 .
    - Het is ook gebleken dat Plane-TDMA en Space-TDMA iteratieve methoden zijn die gebaseerd zijn op standaard incomplete LU-decompositie. Zie subparagraaf 3.2.2 en 3.2.3 .

2. Het inbouwen van een aantal Krylov solvers in WISH3D. Dit zijn de CG, Bi-CGSTAB, GCR en GMRES solver. De CG solver is geschikt voor de het oplossen van de druk aangezien de druk-matrix symmetrisch is. Het is echter gebleken dat deze matrix singulier is. Hoe de singulariteit van de druk-matrix ondervangen kan worden verwijzen we naar de opmerkingen uit paragraaf 3.5 . De rest van de Krylov solvers zijn bedoeld voor het oplossen van de andere variabelen.

3. Het testen van een paar WISH3D solvers en Krylov solvers op twee testproblemen:

    - Pijpstroming: hierbij is gebleken dat de stroming convectie dominant is in de x-richting. Op basis van de beweringen uit subparagraaf 3.2.1 hebben we de Line-TDMA variant Linezxy gebuikt voor het oplossen van alle variabelen behalve de druk. Voor het oplossen van de druk hebben we met verschillende solvers getest. Het is gebleken dat de Line-TDMA varianten geen geschikte solvers zijn voor de druk. Daarentegen zijn een Plane-TDMA variant (in dit geval de Planez solver) en de CG solver wel goede solvers voor het oplossen van deze variabele. Aan de andere kant konden we niet constateren dat de CG solver een betere convergentie van SIMPLER oplevert (maar niet slechter). Een mogelijke verklaring hiervan is dat een klein aantal binnen iteraties (2 binnen iteraties) genoeg is om een redelijke convergentie van SIMPLER te verkrijgen.

- Regenerative cross-fired glassmeltoven: hierij is weer gebleken dat Linezxy een geschikte (maar geen exacte) solver is voor het oplossen van alle variabelen behalve de druk. Het belangrijkste resultaat van dit testprobleem is dat een goede convergentie van SIMPLER te verkrijgen is door het preconditioneren van de Krylov solvers CG en GCR met Gauss-Jacobi. Deze conclusie is heel belangrijk in verband met het verder onderzoek.

5.2 Aanbevelingen voor verder onderzoek

Hierbij zijn twee belangrijke onderwerpen te noemen die onderzocht kunnen worden:

1. Het prallelliseren van de Krylov solvers om de rekentijd te reduceren. We hebben gezien dat het preconditioneren van de Krylov solvers met Gauss-Jacobi tot een redelijke convergentie van SIMPLER leidt. Dit is een gunstig gegeven aangezien de Krylov solvers gepreconditioneerd met Gauss-Jacobi betrekkelijk eenvoudig te parallelliseren zijn.

2. Het versnellen van de SIMPLER methode met behulp van een solver, die gabaseerd is op een Krylov methode voor ***niet−lieaire stelsels***. Met deze aanpak kan men veel minder SIMPLER iteraties verwachten om dezelfde convergentie te verkrijgen als bij het oplossen van ***lineaire stelsels binnen de SIMPLER iteraties***. Hierbij verwijzen we naar de volgende literatuurlijst:

   - P.N. Brown and Y.Saad. ***Hybrid Krylov methods for non linear systems of equations***. SIAM J. Sci. Stat. Comp., 11:450-481, 1990.
   - C.G. Broyden. ***A new method of solving non linear simultaneous equations***. The Computer J., 12:94-99, 1969.

- C.T. Kelley.***Iterative methods for linear and nonlinear equations***. Frontiers in applied mathematics 16. SIAM, Philadelphia, 1995.
- T. Kerkhoven and Y. Saad. ***On acceleration methods for coupeld nonlinear elleptic systems***. Num. Math., 60:525-548, 1992.

Appendix A

Algoritmen voor de Line-TDMA varianten

Algoritme A1 (Lineyzx)

$\boldsymbol{x}^0 =$ beginschatting ,
while $\boldsymbol{1 \leq m \leq M}$ **do**
  for $\boldsymbol{j = 1, ..., NJ}$ **do**
    for $\boldsymbol{k = 1, ..., NK}$ **do**
      $\boldsymbol{A_{j,k} x^m_{j,k} = B_{j,k} x^m_{j,k-1} + S_{j,k} x^m_{j-1,k} + N_{j,k} x^{m-1}_{j+1,k} + T_{j,k} x^{m-1}_{j,k+1} + b_{j,k}}$
    end for
  end for
end while

Algoritme A2 (Linezxy)

$\boldsymbol{x}^0 =$ beginschatting ,
while $\boldsymbol{1 \leq m \leq M}$ **do**
  for $\boldsymbol{k = 1, ..., NK}$ **do**
    for $\boldsymbol{i = 1, ..., NI}$ **do**
      $\boldsymbol{A_{i,k} x^m_{i,k} = B_{i,k} x^m_{i,k-1} + W_{i,k} x^m_{i-1,k} + E_{i,k} x^{m-1}_{i+1,k} + T_{i,k} x^{m-1}_{i,k+1} + b_{i,k}}$
    end for
  end for
end while

Algoritme A3 (Linexzy)

$\boldsymbol{x}^0 =$ beginschatting ,
while $\boldsymbol{1 \leq m \leq M}$ **do**
  for $\boldsymbol{i = 1, ..., NI}$ **do**
    for $\boldsymbol{k = 1, ..., NK}$ **do**
      $\boldsymbol{A_{i,k} x^m_{i,k} = B_{i,k} x^m_{i,k-1} + W_{i,k} x^m_{i-1,k} + E_{i,k} x^{m-1}_{i+1,k} + T_{i,k} x^{m-1}_{i,k+1} + b_{i,k}}$
    end for
  end for
end while

Algoritme A4 (Lineyxz)

$x^0 =$ beginschatting ,
while $1 \leq m \leq M$ **do**
  for $j = 1, ..., NJ$ **do**
    for $i = 1, ..., NI$ **do**
      $A_{i,j}x^m_{i,j} = S_{i,j}x^m_{i,j-1} + W_{i,j}x^m_{i-1,j} + E_{i,j}x^{m-1}_{i+1,j} + N_{i,j}x^{m-1}_{i,j+1} + b_{i,j}$
    end for
  end for
end while

Algoritme A5 (Linexyz)

$x^0 =$ beginschatting ,
while $1 \leq m \leq M$ **do**
  for $i = 1, ..., NI$ **do**
    for $j = 1, ..., NJ$ **do**
      $A_{i,j}x^m_{i,j} = S_{i,j}x^m_{i,j-1} + W_{i,j}x^m_{i-1,j} + E_{i,j}x^{m-1}_{i+1,j} + N_{i,j}x^{m-1}_{i,j+1} + b_{i,j}$
    end for
  end for
end while

Appendix B

Algoritmen voor de Plane-TDMA varianten

Algoritme B1: Planex

$\boldsymbol{x^0 = 0}$,
for $\boldsymbol{1 \le m \le M}$ **do**
  for $\boldsymbol{i = 1, ..., NI}$ **do**
    $\boldsymbol{L_i D_i^{-1} U_i x_i^m = W_i x_{i-1}^m + T_E x_{i+1}^{m-1} + R_i x_i^{m-1} + b_i}$
  end for
  $\boldsymbol{x_i^m = x_i^m + (1-\omega) x_i^{m-1}}$
end for

Algoritme B2: Planey

$\boldsymbol{x^0 = 0}$,
for $\boldsymbol{1 \le m \le M}$ **do**
  for $\boldsymbol{j = 1, ..., NJ}$ **do**
    $\boldsymbol{L_j D_j^{-1} U_j x_j^m = S_j x_{j-1}^m + N_j x_{j+1}^{m-1} + R_j x_j^{m-1} + b_j}$
  end for
  $\boldsymbol{x_j^m = x_j^m + (1-\omega) x_j^{m-1}}$
end for

Bibliografie

[**1**] Patankar, S.V., Numerical Heat Transfer and Fluid Flow, Hemisphere, New York, 1980.

[**2**] L. Post, proefschrift: Modelling of Flow and Combustion in a Glass Melting Furnace, TU Delft, 1988.

[**3**] F.T.M. Nieuwstadt, Turbulentie: Inleiding in de theorie en toepassing van turbulente stromingen, Epsilon, Utrecht, 1992.

[**4**] J. Van Kan, A. Segal, Numerieke Methoden voor Partiële Differentiaalvergelijkingen, Delftse Uitgevers Maatschappij b.v. , Delft, 1989.

[**5**] P. Wilders, P. Wesseling, Dictaat: Numerieke Stromingsleer A, TU Delft, 1994.

[**6**] H.A. van der Vorst, C. Vuik, Dictaat: Voortgezette Numerieke Lineaire Algebra, TU Delft, 1994.

[**7**] A.M. Bruaset, A survey of preconditioned iterative methods, Pitman researsch notes in mathematics series 328, Lonman Scientific and Technical, Harlow, 1995.

[**8**] E.H. Tazammourti, TNO-rapport FSP-RPT-960096: Krylov solvers voor WISH3D, TNO-TPD, 1996.

[**9**] G.P. Boerstoel, proefschrift: Modelling of gas-fired furnaces, TU Delft, 1997.

[**10**] P. Wesseling, An Introduction to Multigrid Methods, John Wiley & Sons, Chichester, 1992.