Multigrid versnelling van de SIMPLE methode voor incompressibele stromingen

A. el Uasghiri
Technische Wiskunde, Numerieke Wiskunde

Technische Universiteit Delft
December 2000

Afstudeercomissie:
Prof.dr.ir.P. Wesseling (TU Delft)
Dr.ir.C. Vuik (TU Delft)
Dr.ir.P. Boerstoel (TNO-TPD)
Ir.E.H. Tazammourti (TNO-TPD)
Dr.A. Twerda (TNO-TPD)

Inhoudsopgave

1 Inleiding **2**

2 Bewegingsvergelijkingen **5**
2.1 Incompressibele stromingen 5
2.2 Continuiteitsvergelijking 5
2.3 Navier-Stokes vergelijkingen 6
2.4 Behoud van energie 7
2.5 Turbulente stromingen 8
2.6 Bewegingsvergelijkingen voor turbulente stromingen 9
2.7 $(k-\epsilon)$ model 10
2.7.1 Rand- en beginvoorwaarden 11
2.7.2 Wandfunctie 12

3 Discretisatie van de Navier-Stokes vergelijkingen **15**
3.1 Discretisatie 15
3.1.1 Orthogonaal rooster 15
3.1.2 Niet-orthogonaal rooster 22

4 Numerieke methoden **29**
4.1 De SIP solver 29

5 Oplosmethode voor de Navier-Stokes vergelijkingen **34**
5.1 De SIMPLE methode 34

6 Multigrid methode **40**
6.1 Multigrid algoritmen 44
6.1.1 Multigrid-SIMPLE algoritme 49
6.1.2 Interpolatie 50

7 Multigrid en 2D Laplace vergelijking **53**
7.1 De exacte oplossing 54
7.2 De numerieke oplossing 54
7.2.1 Implementatie van Gauss-Seidel 57
7.2.2 Convergentie 58

7.2.3 Resultaten Gauss-Seidel . . . 58
7.2.4 Conclusie . . . 59
7.3 Implementatie van de SIP solver . . . 60
7.3.1 Resultaten van de SIP solver . . . 61
7.3.2 Conclusie . . . 64
7.3.3 Gauss-Seidel t.o.v. SIP solver . . . 65
7.4 Multigrid methode . . . 65
7.4.1 Datastructuur . . . 67
7.4.2 Bilineaire interpolatie . . . 69
7.4.3 Resultaten MGM . . . 72

8 De X-stream code 78
8.1 Onderrelaxatie . . . 83
8.2 Convergentie cretiria . . . 84
8.3 Het genereren van niet-orthogonale grids . . . 85
8.4 Local gridverdfijning . . . 88
8.5 Het bepalen van de snelheid op het zijvlak in Colocated grid . . . 89
8.6 Testcase . . . 92

9 Multigrid methode in X-stream 103
9.1 Restrictie en prolongatie . . . 109
9.1.1 Restrictie . . . 110
9.1.2 Prolongatie . . . 113
9.1.3 Implementatie . . . 114
9.1.4 Resultaten . . . 117

10 Conclusies 132

11 Appendix A 134

12 Appendix B 141

13 Appendix C 157

Hoofdstuk 1

Inleiding

De TPD [1] (Technisch Pysische Dienst) te Delft is één van de 15 gevestigde instituten van TNO (Nederlandse Organisatie voor Toepaste Natuurwetenschappelijk Onderzoek) in Nederland. De TPD is zelf ook weer onderverdeeld in een aantal subdivisies, waarvan Proces Fysica (PF) er één van is. Binnen PF houdt men zich bezig met het modelleren van warmte en stromingen in alle soorten van industriële processen. Een voorbeeld hiervan zijn de glasovens waaraan veelvuldig door PF gerekend wordt.
Ten behoeve van het simuleren van de fysische processen, is men binnen TNO-TPD bezig het CFD pakket X-stream te ontwikkelen (zie [1]). Met dit software pakket hoopt men de turbulente stroming, de warmte vergelijking en het massatransport in een gecompliceerde geometrie te simuleren. Deze stromingen mogen zowel stationair als instationair zijn. Voordat er gerekend kan worden dient een rekenrooster gemaakt te worden. In X-stream wordt het rekenrooster in een aantal blokken (multiblok) verdeeld, waarbij elke blok uit een gestructureerde niet-orthogonale mesh bestaat. Binnen dit pakket is het mogelijk om een gegeven blok een aantal keer ten opzichte van de andere blokken te verfijnen, ook wel lokale verfijning genoemd. Lokale verfijning is nodig in deelgebieden van het rekendomein, waar we meer informatie over het gedrag van de stroming willen hebben; een voorbeeld van zo'n deelgebied is de grenslaag.
Voor het genereren van multiblok roosters wordt op dit moment gebruik gemaakt van de FLUENT gridgenerator. FLUENT is een CFD code die ook binnen TNO-TPD gebruikt wordt. De geproduceerde gridfile van FLUENT wordt naar een gridfile geconverteerd in een formaat dat voor X-stream leesbaar is. X-stream draait ook parallel, waarbij elke processor één of meer blokken toegewezen krijgt.
Het ontwikkelen van X-stream is tot een fase gevorderd waarin het simuleren van laminaire- en turbulente stromingen een feit is. Het model dat voor de simulatie gebruikt wordt in X-stream bestaat uit de Navier-Stokes vergelijkingen. Het geheel vormt een stelsel van niet-lineaire gekoppelde diffentiaalvergelijkingen, die niet analytisch kunnen worden opgelost. Voor het simuleren van turbulente stromingen

[1] www.tno.tpd.nl

wordt in X-stream gebruik gemaakt van het standaard $(k-\epsilon)$-model. Het $(k-\epsilon)$-model bestaat uit twee niet lineaire vergelijkingen, die ook niet analytisch op te lossen zijn. X-stream berust dan ook op numerieke methoden om een oplossing te bepalen. Het numerieke model in X-stream is gebaseerd op:

- Eindige Volume Methode (EVM): de behoudswetten (voor impuls, massa, energie enzovoort) worden met behulp van het theorema van Gauss geintegreerd over volume elementen (normaliter cellen genoemd). Dit houdt onder andere dat in alle omstandigheden per definitie aan alle behoudswetten voldaan wordt.

- Om de gekoppelde niet lineaire Navier-Stokes vergelijkingen op te lossen, wordt de $SIMPLE$ methode ($\underline{S}$emi $\underline{I}$mplicit $\underline{M}$ethod for $\underline{P}$ressure $\underline{L}$inked $\underline{E}$quations) gebruikt. Tijdens een buiteniteratie van SIMPLE worden de lineaire stelsels in X-stream met een iteratieve methode, bekend als SIP solver ($\underline{S}$trongly $\underline{I}$mplicit $\underline{P}$rocedure), opgelost, zie figuur 1.1.

Het flowdiagram in figuur 1.1 geeft de oplossingsprocedure van X-stream aan. X-stream is verbonden met pre- en postprocessing. Tussen de pre- en postprocessing zit het hoofdprogramma, zie figuur 1.1.

- De preprocessing bestaat uit de input-files, waarin problemen gedefinieerd kunnen worden. Deze file bevat de gridfile, de randvoorwaarden enzovoort. Hiermee wordt X-stream bestuurd.

- Het hoofdprogramma bestaat uit het oplossen van de gediscretiseerde stelsels. Hierin worden de berekeningen gevoerd.

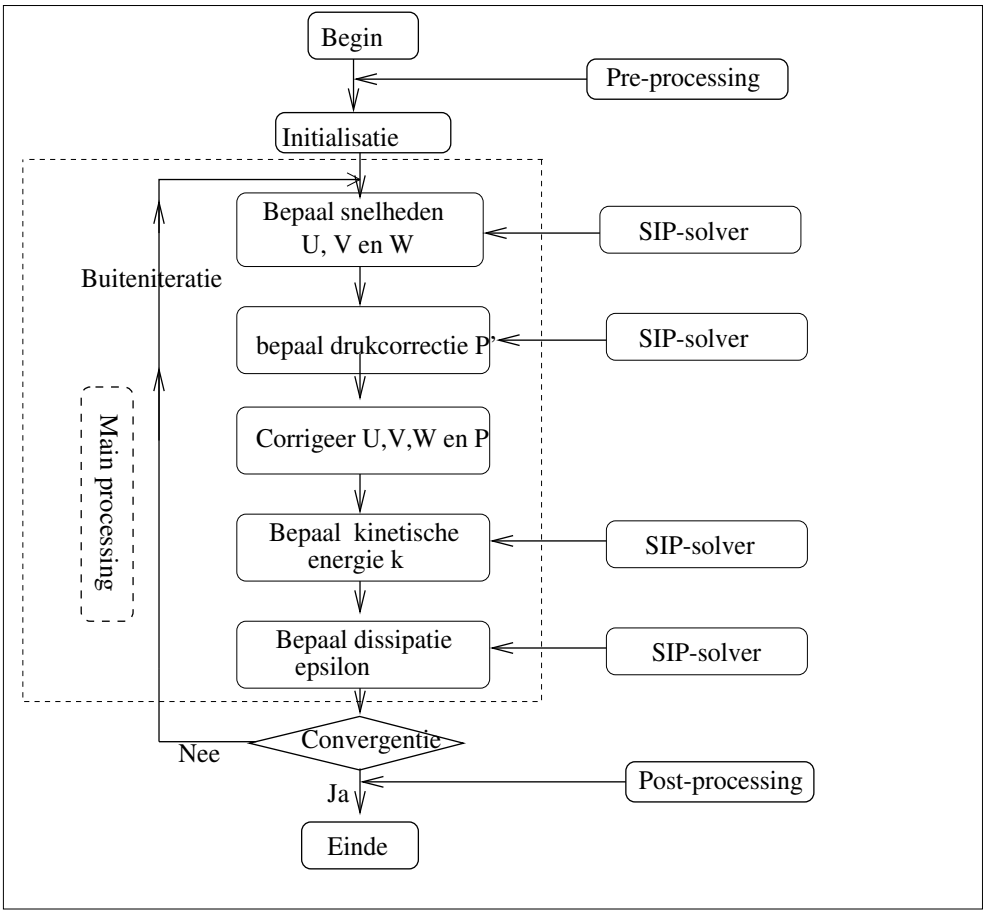
- De postprocessing bestaat uit de output-files, waarin de data verkregen in het hoofdprogramma geschreven wordt. Deze data kunnen we visualiseren.

Het doel

Oplosmethode SIMPLE is waarschijnlijk de meest gebruikte methode om incompressibele vergelijkingen op te lossen. Het nadeel van de SIMPLE methode is dat deze methode steeds minder efficiënt wordt naarmate het rekenrooster verfijnd wordt. De reden van de trage convergentie is ondermeer de zwakke koppeling tussen de variabelen en de niet-lineariteit van de op te lossen vergelijkingen. De convergentie van SIMPLE is sterk afhankelijk van het aantal roosterpunten. Om convergentie van SIMPLE in X-stream te versnellen zullen we een Multigrid methode gebruiken.

Algemene opzet

In hoodstuk 2 van dit verslag wordt het mathematische model beschreven. Er wordt aandacht besteed aan de incompressibele Navier-Stokes vergelijkingen en het standaard $(k-\epsilon)$ model. De partiële differentiaal vergelijkingen worden aangevuld met



Figuur 1.1: De oplossingsprocedure van X-stream

begin- en randvoorwaarden. Het discretiseren van de gegeven partiële differentiaal vergelijkingen met behulp van de Eindige Volume Methode met een 'colocated arrangement van de onbekenden' wordt in hoofdstuk 3 behandeld. Hieruit worden stelsels lineaire vergelijkingen verkregen van de vorm $A\phi = b$, waarbij A een ijle matrix is. De vector ϕ bevat de onbekenden. In hoofdstuk 4 wordt een iteratieve methode de SIP solver behandeld. In hoofdstuk 5 wordt voor het bepalen van een stelsel vergelijkingen voor de druk, een druk-correctie methode, SIMPLE, beschreven. In hoofdstuk 6 wordt aandacht besteed aan de Multigrid methode en het versnellen van SIMPLE met een Multigrid methode. In hoofdstuk 7 wordt implementatie en validatie van multigrid aan de hand van 2D Laplace vergelijking behandeld. In hoofdstuk 8 wordt de X-stream code beschreven. In hoofdstuk 9 wordt implementatie en validatie van multigrid in X-stream beschreven. Tot slot worden in hoofdstuk 10 de conclusies besproken.

Hoofdstuk 2

Bewegingsvergelijkingen

2.1 Incompressibele stromingen

De vergelijkingen, die stroming van een vloeibare medium beschrijven, zijn gebaseerd op de wetten van behoud van massa, impuls en energie (zie [2]). Als uitgangspunt beschouwen we een zogenaamd vloeistofelement met volume V. De positie van het vloeistofelement wordt vastgesteld in een Cartesisch assenstelsel (x_1, x_2, x_3). Het vloeistofelement zal zich door het medium bewegen met de snelheid u_i ter plaatse van de positie van het element. Verder zullen we gebruik maken van de Einstein-sommatie-conventie. Dit betekent dat in het geval van een herhaalde index (bijvoorbeeld $u_i u_i$) over alle richtingen (x_1, x_2, x_3) wordt gesommeerd: $u_i u_i = u_1 u_1 + u_2 u_2 + u_3 u_3$.

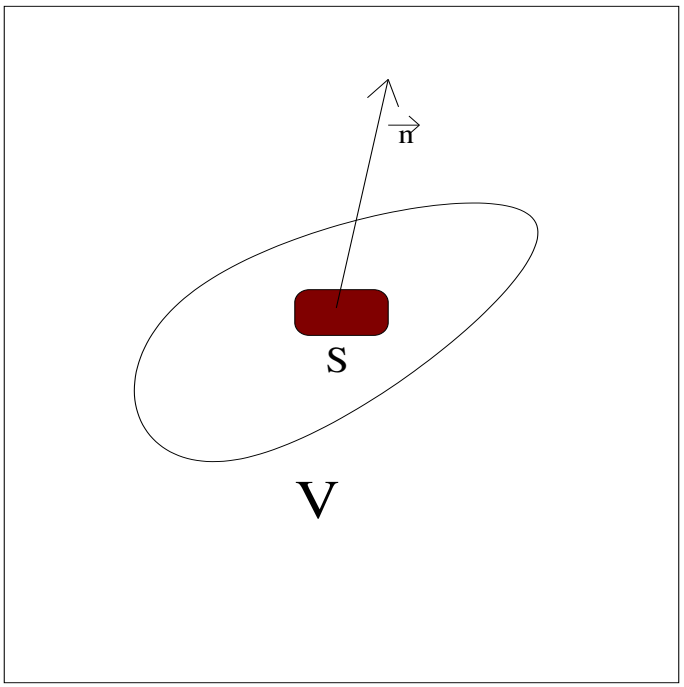
2.2 Continuiteitsvergelijking

Beschouw een vast (dat wil zeggen niet meebewegend met de stroming) volume V, dat omsloten wordt door een oppervlak S, waarop de buitennormaal n gedefinieerd is, zie figuur 2.1. Behoud van massa impliceert dat de verandering van de massa in V gelijk is aan de netto massaflux die door het oppervlak S stroomt, (mits er geen interne bronnen in V aanwezig zijn).
Een integraalvorm van de massabalans is:

$$\frac{\partial M}{\partial t} \equiv \frac{\partial}{\partial t} \int_V \rho dV = - \int_S \rho v \cdot n dS, \tag{2.1}$$

waarbij M is de totale massa in V en ρ is de dichtheid van het medium.
Toepassing van de divergentie stelling van Gauss op (2.1) geeft:

$$\int_V \frac{\partial \rho}{\partial t} dV = - \int_V div(\rho v) dV. \tag{2.2}$$



Figuur 2.1: Een vloeistofelement met volume V

Aangezien de vergelijking (2.2) voor alle V geldt, volgt :

$$\frac{\partial \rho}{\partial t} + div(\rho v) = 0. \tag{2.3}$$

De vergelijking (2.3) staat bekend als continuiteitsvergelijking.

2.3 Navier-Stokes vergelijkingen

Behoud van impuls voor een incompressibele stroming, ofwel een onsamendrukbare stroming (ρ is constant) en tevens voor een zogenaamd Newtons fluidum, wordt beschreven door de volgende vergelijkingen:

$$\frac{\partial(\rho u_i)}{\partial t} + \frac{\partial}{\partial x_j}(\rho u_i u_j) = -\frac{\partial p}{\partial x_i} - \frac{\partial}{\partial x_j}(\sigma_{ij}) + \rho g_i, \tag{2.4}$$

waarbij ρg_i de volumekracht is, met g_i de zwaartekrachtversnelling in de x_i-richting. De term σ_{ij} is spanningstensor. Deze tensor is gedefinieerd als

$$\sigma_{ij} = -\mu(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}) + \frac{2}{3}\mu\frac{\partial u_k}{\partial x_k}\delta_{ij}, \tag{2.5}$$

waarin $\delta_{ij} = 0$ als $i \neq j$ en $\delta_{ij} = 1$ als $i = j$. De term μ staat bekend als de dynamische viscositeit. Dat is een materiaalconstante, die dus alleen afhankelijk is van het medium.

De vergelijkingen (2.4) staan bekend als de Navier-Stokes vergelijkingen.

2.4 Behoud van energie

Behoud van energie in een vloeistof of gas wordt beschreven door, (zie [3])

$$\frac{\partial}{\partial t}(\rho E) + \frac{\partial}{\partial x_j}(\rho u_j E) = -\frac{\partial}{\partial x_j}(q_j) - \frac{\partial}{\partial x_j}(p u_j) + \rho u_i g_i - \frac{\partial}{\partial x_j}(\sigma_{ij} u_i) + S_{rad}. \tag{2.6}$$

waarin $E = U + \frac{1}{2} u_i u_i$, waarbij U is de inwendige energie per massaeenheid, de term $\frac{1}{2} u_i u_i$ is de kinetische energie, q_j is de j-component van de energieflux en S_{rad} is de stralingsbronterm.
Als wij vergelijking (2.6) in termen van enthalpie herschrijven ($H = U + \frac{p}{\rho}$) dan volgt:

$$\frac{\partial}{\partial t}(\rho H + \frac{1}{2}\rho u_i u_i - p) + \frac{\partial}{\partial x_j}(\rho u_j (H + \frac{1}{2} u_i u_i)) = -\frac{\partial}{\partial x_j}(q_j) + \rho u_i g_i - \frac{\partial}{\partial x_j}(\sigma_{ij} u_i) + S_{rad}. \tag{2.7}$$

In ons geval is de enthalpie de som van de voelbare enthalpie H_{sens} en de latente (chemische) enthalpie H_{lat}. Voor een ideaal gas (perfect gas) geldt $dH_{sens} = c_p dT$ waarbij T de temperatuur voorstelt en c_p is de specifieke warmte bij constante druk. Voor de energieflux geldt het volgende

$$q_j = -\frac{\lambda}{c_p}\frac{\partial H}{\partial x_j}, \tag{2.8}$$

waarbij λ de thermische conductiviteit voorstelt. De visceuse dissipatieterm en de gravitatieterm worden verwaarloosd. Tevens is de kinetische energie $\frac{1}{2} u_i u_i$ verwaarloosbaar ten opzichte van enthalpie H zolang het Mach getal klein is. Het Mach getal is volgt gedefinieerd :

$$Mach\ getal = \frac{snelheid\ van\ de\ stroming}{geluidsnelheid}$$

De vergelijking (2.7) wordt dan :

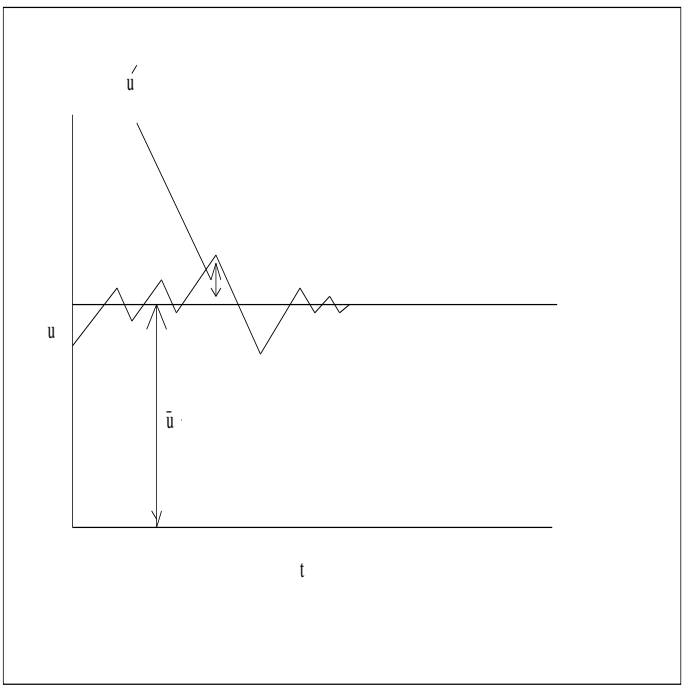
$$\frac{\partial}{\partial t}(\rho H - p) + \frac{\partial}{\partial x_j}(\rho u_j H) = -\frac{\partial}{\partial x_j}(\frac{\lambda}{c_p}\frac{\partial H}{\partial x_j}) + S_{rad}. \tag{2.9}$$

2.5 Turbulente stromingen

Binnen de stromingsleer zijn er twee typen stromingen te onderscheiden: laminair en turbulent. De karakteristieke eigenschappen van beide stomingstypen zijn:

- Laminair: gelaagd, glad en geordend,
- Turbulent: fluctuerend en chaotisch.

In een turbulente stroming, is men in het algemeen geinteresseerd in het gemiddelde van een variabele ϕ over een tijdsinterval [0,T]. Als uitgangspunt stellen we dat in een turbulente stroming een instantane variabele ϕ gesplitst kan worden in een gemiddelde plus een fluctuatie. Neem als voorbeeld de snelheidscomponent u dan geldt $u = \overline{u} + u'$ waarbij de middelingsoperator wordt aangegeven met een streep boven de variabele en een fluctuatie wordt aangegeven met een accent zie figuur 2.2. Deze splitsing wordt meestal de Reynoldsdecompositie genoemd. Deze middelingsoperator voldoet aan de volgende regels (zie [2]):



Figuur 2.2: Turbulente stroming

$$\overline{f+g} = \overline{f}+\overline{g}, \tag{2.10}$$
$$\overline{\alpha f} = \alpha\overline{f}, \tag{2.11}$$
$$\overline{\frac{\partial f}{\partial s}} = \frac{\partial \overline{f}}{\partial s}, \tag{2.12}$$
$$\overline{\overline{f}g} = \overline{f}\overline{g}. \tag{2.13}$$

Hierin zijn f en g fluctuerende grootheden en α is een constante. Op basis van (2.10-2.13) kunnen we de volgende relaties afleiden: $\overline{u'} = 0$, $\overline{\overline{u}} = \overline{u}$ en $\overline{\overline{f}\overline{g}} = \overline{f}\overline{g}$ Definitie van de Reynoldsdecompositie voor de variabele ϕ luidt

$$\phi(\underline{x},t) = \overline{\phi}(\underline{x}) + \phi'(\underline{x},t), \tag{2.14}$$

waarin het Reynoldsgemiddelde $\overline{\phi}(\underline{x})$ gegeven wordt door:

$$\overline{\phi}(\underline{x}) = \lim_{T\to\infty}\frac{1}{T}\int_0^T \phi(\underline{x},t)ds, \tag{2.15}$$

waarbij T de middelingstijd voorstelt.

2.6 Bewegingsvergelijkingen voor turbulente stromingen

Om de bewegingsvergelijkingen voor een turbulente stroming te vinden, definiëren we

$$u_i = \overline{u_i} + u_i', \tag{2.16}$$
$$\rho = \overline{\rho} + \rho'. \tag{2.17}$$

Als wij (2.16-2.17) in de bewegingsvergelijking (2.3-2.4) substitueren dan volgt

Behoud van massa

$$\frac{\partial}{\partial t}(\overline{\rho}+\rho') + \frac{\partial}{\partial x_j}((\overline{\rho}+\rho')(\overline{u_i}+u_i')) = 0. \tag{2.18}$$

Als we nu de fluctuaties van de dichtheid ρ' in (2.18) verwaarlozen dan volgt

$$\frac{\partial}{\partial t}(\overline{\rho}) + \frac{\partial}{\partial x_j}(\bar{\rho}\bar{u}_i) + \frac{\partial}{\partial x_j}(\overline{\rho}u_i') = 0. \tag{2.19}$$

Als we nu het Reynoldsgemiddelde van (2.19) nemen dan volgt de vergelijking voor massabehoud in een turbulente stroming

$$\frac{\partial}{\partial t}(\overline{\rho}) + \frac{\partial}{\partial x_j}(\overline{\rho}\bar{u}_i) = 0. \tag{2.20}$$

Navier-Stokes vergelijkingen

De Navier-Stokes vergelijkingen voor een turbulente stroming worden:

$$\frac{\partial}{\partial t}(\overline{\rho}\bar{u}_i) + \frac{\partial}{\partial x_j}(\overline{\rho}\bar{u}_i\bar{u}_j + \overline{\rho}\overline{u_i'u_j'}) = -\frac{\partial \overline{p}}{\partial x} + \frac{\partial}{\partial x_j}(\overline{\sigma_{ij}}) + \overline{\rho}g_i, \tag{2.21}$$

waarbij

$$\overline{\sigma_{ij}} = \mu(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i}) - \frac{2}{3}\mu\frac{\partial \overline{u_i}}{\partial x_j}\delta_{ij}. \tag{2.22}$$

2.7 $(k-\epsilon)$ model

Door het toepassen van de Reynoldsdecompositie hebben we in de turbulente vergelijking nieuwe onbekenden $(\overline{\rho}\overline{u_i'u_j'})$ gecreëerd, de zogenaamde Reynold stress termen. Nu zijn er meer onbekenden dan het aantal vergelijkingen. Om deze onbekenden te bepalen is een turbulentie model nodig. Welk turbulentie model gebruikt wordt hangt mede af van het probleem. Wij kiezen voor modellen waarbij tenminste de turbulente kinetische energie k en de de dissipatie ervan ϵ afgeschat worden. Voor de Reynoldstressterm gebruiken we de volgende uitdrukking (zie [3]):

$$\overline{\rho}\overline{u_i'u_j'} = -\mu_t(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_i}}{\partial x_j}) + \frac{2}{3}\overline{\rho}k\delta_{ij}, \tag{2.23}$$

waarbij $k = \frac{1}{2}\overline{u_i'u_j'}$ de turbulente kinetische energie voorstelt. De turbulente viscositeit is:

$$\mu_t = \frac{C_\mu \overline{\rho} k^2}{\epsilon}, \tag{2.24}$$

waarin C_μ een empirische constante is en ϵ is de dissipatie van turbulente kinetische energie. ϵ is gedefinieerd als:

$$\epsilon = \frac{\mu}{\overline{\rho}}(\overline{\frac{\partial u_i'}{\partial x_j}\frac{\partial u_i'}{\partial x_j}}). \tag{2.25}$$

Om de onbekende, Reynoldstermen te bepalen moeten k en ϵ bekend zijn. Hiervoor zijn twee extra vergelijkingen nodig:

$$\frac{\partial}{\partial t}(\overline{\rho}k) + \frac{\partial}{\partial x_j}(\overline{\rho}\bar{u_j}k) = \frac{\partial}{\partial x_j}((\mu + \frac{\mu_t}{\sigma_k})\frac{\partial k}{\partial x_j}) + P_k - \overline{\rho}\epsilon, \tag{2.26}$$

$$\frac{\partial}{\partial t}(\overline{\rho}\epsilon) + \frac{\partial}{\partial x_j}(\overline{\rho}\bar{u_j}\epsilon) = \frac{\partial}{\partial x_j}((\mu + \frac{\mu_t}{\sigma_\epsilon})\frac{\partial \epsilon}{\partial x_j}) + (C_{\epsilon 1}P_k - C_{\epsilon 2}\overline{\rho}\epsilon)\frac{\epsilon}{k}, \tag{2.27}$$

waarbij σ_k en σ_ϵ de turbulente Prandtlgetallen voorstellen, $C_{\epsilon 1}$ en $C_{\epsilon 2}$ zijn empirische constanten en P_k is de productie term en deze wordt gegeven door

$$P_k = (\mu_t(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i}) - \frac{2}{3}\overline{\rho}k\delta_{ij})\frac{\partial \overline{u_i}}{\partial x_j}. \tag{2.28}$$

Het blijkt dat als dichtheidscorrelaties niet verwaarloosbaar zijn in turbulente stromingen handiger is om de Favre decompositie te gebruiken. Er onstaan dan geen extra correlaties in de convectie termen. In X-stream is momenteel slechts de Reynoldsdecompositie toegepast.

2.7.1 Rand- en beginvoorwaarden

Voor het oplossen van de partiële differentiaal vergelijkingen dienen we voor de stationaire problemen over randvoorwaarden te beschikken. Voor de niet-stationaire problemen komen er beginvoorwaarden bij. Het opstellen van de juiste voorwaarden is belangrijk. Algemene regels zijn niet te geven, men kan zich het beste in het fysische probleem afvragen wat er nodig is, wil het probleem fysisch goed gesteld zijn. Een aantal veel voorkomende randvoorwaarden, die opgelegd kunnen worden voor een variabele ϕ zijn Dirichlet- en Neumann randvoorwaarden.

randvoorwaarden

- Dirichlet: $\phi = f$,
- Neumann: $(\Gamma_\phi \frac{\partial \phi}{\partial x_j})n_j = f$,
- Voor een niet meebewegend randoppervlak geldt: voor de tangentiële snelheid geldt $u_t = 0$ en is bekend als no-slip voorwaarde. Voor de normale snelheid geldt $u_n = 0$ en is bekend als no-flow voorwaarde.

beginvoorwaarden

Voor de beginvoorwaarden moeten we de waarde op tijd t_0 kennen ofwel $\phi(t_0) = \phi_0$.

2.7.2 Wandfunctie

De wandfunctie gebruiken we als randvoorwaarden voor de turbulente kinetische energie k en de dissipatie ϵ. Direct naast de wand in een zeer dunne laag, de grenslaag, zijn de aanames voor het turbulentie model in het buitengebied niet meer geldig. Daarnaast neemt de snelheid zeer snel af tot nul aan de wand. Voor de wandfunctie beschouwen we de stroming in het wandgebied waarin de parallele richting en de normaalrichting aangegeven zijn met x respectievelijk y. Hierbij kan de stroming opgevat worden als één-dimensionale Couette-stroming met snelheid U in de x-richting (zie [3]). In vergelijkingsvorm :

$$\frac{\partial}{\partial y}(\mu + \mu_t)\frac{\partial U}{\partial y} = 0 \qquad voor \;\; y > 0. \tag{2.29}$$

waarbij μ en μ_t de moleculaire dynamische, respectievelijk, de turbulente viscositeit zijn. We definieren de wandschuifspanning τ_ω door :

$$\tau_\omega = (\mu + \mu_t)\frac{\partial U}{\partial y} \qquad op \;\; y = 0. \tag{2.30}$$

Uit de vergelijking (2.30) volgt :

$$\frac{\partial U}{\partial y} = \frac{\tau_\omega}{\mu + \mu_t}. \tag{2.31}$$

We definiëren een zogenaamde wrijvingssnelheid U^*, een dimensieloze snelheid U^+ en een dimensieloze afstand tot de wand y^+ :

$$U^* = \sqrt{\frac{\tau_\omega}{\rho}}, \tag{2.32}$$

$$U^+ = \frac{U}{U^*}, \tag{2.33}$$

$$y^+ = \frac{\rho y U^*}{\mu}. \tag{2.34}$$

Substitutie van (2.33) en (2.34) in (2.31) resulteert in de volgende dimensieloze vergelijking :

$$\frac{\partial U^+}{\partial y^+} = \frac{\mu}{\mu + \mu_t}. \tag{2.35}$$

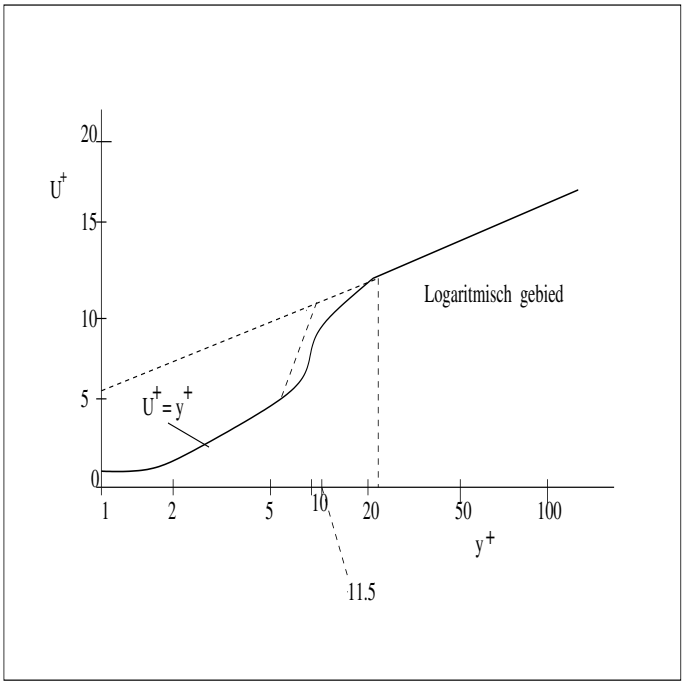
We kunnen μ_t schatten met behulp van de zogenaamde Prandtl-mengweg hypothese:

$$\mu_t = \rho l_m^2 |\frac{\partial U}{\partial y}|. \tag{2.36}$$

waarbij l_m de mengweg is en deze is gedefinieerd door $l_m = k_v y$ met $k_v = 0.42$ en is bekend als de Karman constante. We kunnen μ_t uitdrukken in termen van dimensieloze variabelen U^+ en y^+ door (2.34) en (2.34) te gebruiken. Het resultaat is :

$$\mu_t = \mu k_v^2 (y^+)^2 |\frac{\partial U^+}{\partial y^+}|. \tag{2.37}$$

Het wandgebied waarin vergelijking (2.35) geldig is, kan opgedeeld worden in twee sublagen, de logaritmische en de visceuze laag, zie figuur 2.3.



Figuur 2.3: De grenslaag

De logaritmische sublaag is het deel van het gebied waarin de moleculaire dynamische viscositeit μ te verwaarlozen is ten opzichte van de turbulente viscositeit μ_t. Dus de vergelijking (2.35) wordt:

$$\frac{\partial U^+}{\partial y^+} = \frac{\mu}{\mu_t}. \tag{2.38}$$

Substitutie van (2.37) in vergelijking (2.38) geeft :

$$k_v^2(y^+)^2|\frac{\partial U^+}{\partial y^+}|\frac{\partial U^+}{\partial y^+} = 1. \tag{2.39}$$

De vergelijking (2.39) heeft de volgende oplossing :

$$U^+ = \frac{1}{k_v} ln(Ey^+) \qquad \text{voor } \; y^+ \geq 11.5. \tag{2.40}$$

waarbij $E = 9.8$ een integratie constante is. De waarde van E is experimenteel bepaald. De viskeuze laag is het gebied vlakbij de wand waarin de turbulente viscositeit μ_t te verwaarlozen is ten opzichte van de viscositeit μ. In dit geval wordt de vergelijking (2.35) :

$$\frac{\partial U^+}{\partial y^+} = 1. \tag{2.41}$$

De vergelijking lossen we op met de randvoorwaarde $U^+ = 0$ voor $y^+ = 0$ en we krijgen

$$U^+ = y^+ \qquad\qquad voor \;\; y^+ \leq 11.5. \tag{2.42}$$

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 (2.26) 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 de vergelijking (2.26). Hieruit volgt :

$$P_k - \rho\epsilon = 0, \tag{2.43}$$

waarbij de produktie term P_k bepaald kan worden door vergelijking (2.28) te beschouwen voor een Couette-stroming:

$$P_k = \mu_t(\frac{\partial U}{\partial y})^2. \tag{2.44}$$

Als wij de vergelijking (2.44) en (2.37) in (2.43) invullen en vervolgens de dimensieloze variabelen U^+ en y^+, dan krijgen we :

$$\epsilon = (\frac{\rho}{\mu})(k_v y^+)^2 (U^*)^4 |\frac{\partial U^+}{\partial y^+}|(\frac{\partial U^+}{\partial y^+})^2. \tag{2.45}$$

Substitutie van (2.37) en (2.45) in (2.24) dan volgt :

$$k = (c_\mu)^{-\frac{1}{2}}(k_v y^+)^2 (U^*)^2 (\frac{\partial U^+}{\partial y^+})^2. \tag{2.46}$$

In het volgende hoofdstuk zullen we het discretiseren van de Navier-Stokes vergelijkingen beschrijven.

Hoofdstuk 3

Discretisatie van de Navier-Stokes vergelijkingen

3.1 Discretisatie

Vaak is het niet mogelijk de vergelijkingen (impulsbalans, massabalans, energiebalans enz.) zodanig te vereenvoudigen dat een analytische of benaderende oplossing gevonden kan worden. De vereenvoudiging is niet mogelijk omdat bijvoorbeeld alle termen in de drie-dimensionale transportvergelijkingen van belang zijn en bovendien zijn deze vergelijkingen gekoppeld door bijvoorbeeld een temperatuur afhankelijke stofeigenschap zoals viscositeit en dichtheid. Dus moet er naar een ander manier gezocht worden om tot een oplossing van het probleem te komen. Een mogelijkheid is om deze problemen numeriek op te lossen. Er zijn een aantal methoden bekend:

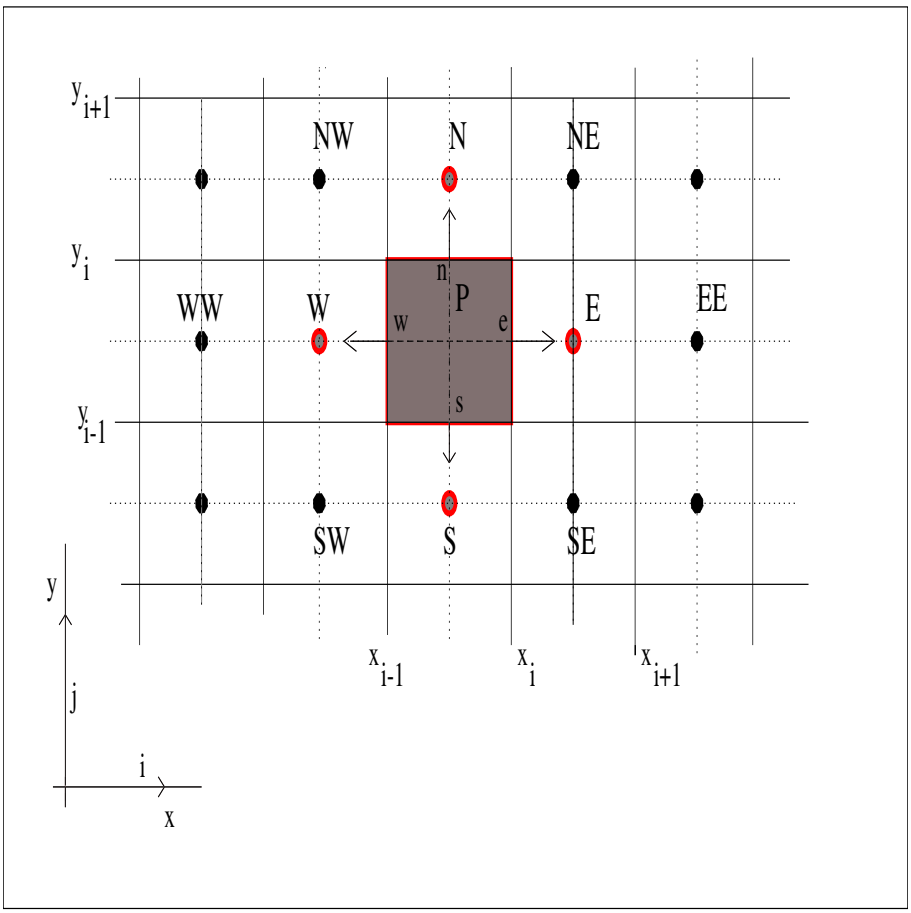
- Eindige Elementen Methode (EEM),
- Eindige Differentie Methode (EDM),
- Eindige Volume Methode (EVM).

In dit verslag wordt de EVM gehanteerd. Alleerst dient vastgesteld te worden wat de aard van het rooster moet zijn dat gebruikt zal gaan worden. Er zijn nogal wat mogelijkheden: cilindrisch, orthogonaal of niet orthogonaal, twee of drie dimensies. In X-stream wordt gebruik gemaakt van een drie-dimensionaal niet orthogonaal rooster. We beschouwen eerst het orthogonale rooster.

3.1.1 Orthogonaal rooster

Het drie-dimensionale gebied, waarop de oplossing van een probleem gevonden moet worden, wordt in een eindige aantal control volumes (Cv's) verdeeld zie figuur 3.1 voor een twee-dimensionaal gebied. Ieder Cv bevat één roosterpunt P in

het centrum. In dit punt worden de onbekenden ϕ (de snelheidscomponenten u, v, w, de druk p, de temperatuur T, de kinetische energie k en de dissipatie ϵ) bepaald. Hier wordt gebruik gemaakt van het zogenaamd 'colocated grid arrangement', (zie [4]). Elke Cv is omsloten door een aantal zijvlakken S_k, zes in $3D$ en vier in $2D$. Deze zijvlakken zijn aangegeven ten opzichte van het centrum van P met e(ast), w(est), n(orth), s(outh), t(op) en b(ottom) zie figuur 3.1. De algemene vorm van de vergelijkingen in hoofdstuk (2) is:



Figuur 3.1: Het orthogonale rekenrooster

$$\frac{\partial}{\partial t}(\rho\phi) + \nabla(\rho v\phi) - \nabla\cdot(\Gamma_\phi\nabla\phi) = S_\phi. \tag{3.1}$$

We beschouwen een control volume Cv met volume $V = \Delta x\Delta y\Delta z$. We integreren over deze dit Control Volume:

$$\int_V \frac{\partial \rho\phi}{\partial t} dV + \int_V \nabla(\rho v\phi) dV = \int_V \nabla \cdot (\Gamma_\phi \nabla\phi) dV + \int_V S_\phi dV. \tag{3.2}$$

Het divergentietheorema van Gauss toepassen op (3.2) geeft:

$$\int_V \frac{\partial \rho\phi}{\partial t} dV = -\int_S (\rho v \cdot \phi) dS + \int_S \Gamma_\phi \nabla\phi \cdot n dS + \int_V S_\phi dV, \tag{3.3}$$

waarbij S de rand van het Cv is en n is de eenheidnormaal op S. De netto flux door een Cv is te schrijven als de som van de totale flux die door de zijvlakken stroomt.

$$\int_S f dS = \sum_k \int_{S_k} f dS, \tag{3.4}$$

waarbij f een component is van de convectieterm $(\rho\phi v \cdot n)$ of een component van de diffusieterm $(\Gamma_\phi \nabla\phi \cdot n)$. We concentreren ons op zijvlak 'e' van één Cv. Om de integraal $\int_{S_e} f dS$ exact te bepalen, moeten we f overal op het zijvlak 'e' weten. Aangezien deze waarden niet bekend zijn zullen er interpolatie schema's gebruikt worden. De eenvoudigste benadering is de midpunt regel:

$$F_e = \int_{S_e} f dS \approx f_e \int_{S_e} dS. \tag{3.5}$$

Om de convectie term (f^c) en de diffusieterm (f^d) te berekenen, nemen we aan dat de snelheidscomponenten u, v, w en ρ en Γ bekend zijn. Er zullen twee interpolatie schema's beschreven worden, maar we definiëren eerst het rooster Peclet getal (Pe): $Pe = \frac{\rho_e u_e \Delta x_e}{\Gamma_\phi}$. Numerieke oscillaties kunnen optreden als $Pe > 2$ met het centrale differentieschema.

Upwind Differentie Schema (UDS) :

$\phi_e = \phi_P$ als $(v \cdot n)_e \geq 0$
en
$\phi_e = \phi_E$ als $(v \cdot n)_e < 0$

- voordeel is dat met dit schema oscillaties voorkomen worden, omdat er niet aan de voorwaarde voor het Peclet getal voldaan hoet te worden (zie [5]).
- nadeel is dat dit schema minder nauwkeurig is.

Central Differentie Schema (CDS) :

Een andere benadering van het zijvlak 'e' is lineaire interpolatie tussen de buren van het centrum P.

$$\phi_e = \phi_E \lambda_e + \phi_P (1 - \lambda_e). \tag{3.6}$$

met

$$\lambda_e = \frac{x_e - x_P}{x_E - x_P}. \tag{3.7}$$

- nadeel is dat met dit schema oscillaties gegenereerd kunnen worden, omdat voor grote stapgrootte Δx_e niet aan de voorwaarde $Pe \leq 2$ voldaan is.
- voordeel is dat dit schema nauwkeurig is ten opzichte van UDS.

Het is ook mogelijk om beide schema's (UDS en CDS) te combineren. Dit geeft:

$$\phi_e = \phi_e^{UDS} + \beta(\phi_e^{CDS} - \phi_e^{UDS})^{vorige}. \tag{3.8}$$

waarbij $0 \leq \beta \leq 1$ en β is de blending factor. De term met superscript 'vorige' is afkomstig van de vorige iteratie. Dit schema genereert bij een geschikte β geen oscillaties en is nauwkeuriger dan UDS. Dit schema staat bekend als deferred correctie schema en dit schema is eerst voorgesteld door Khosla en Rubin [12]. Het schema (3.8) is met de schema's UDS en CDS aan de hand van een één-dimensionale convectie-diffusie vergelijking bekeken. De resultaten zijn in appendix A weergegeven.

Voor de afgeleide in f^d wordt de benadering met CDS:

$$(\frac{\partial \phi}{\partial x})_e \approx \frac{\phi_E - \phi_P}{x_E - x_P}. \tag{3.9}$$

Voor de bronterm hebben we :

$$Q_p = \int_V S_\phi dV \approx (S_C + S_P \phi_P)\Delta V_P. \tag{3.10}$$

waarbij de bronterm gelineariseerd is.
We zullen deze interpolatie schema's voor een twee-dimensionale stationaire convectie-diffusie vergelijking (3.3) uitschrijven.

$$\int_S (\rho v \cdot \phi) dS = \int_S \Gamma_\phi \nabla \phi \cdot n dS + \int_V S_\phi dV. \tag{3.11}$$

De convectie term

De convectie term voor de variabele ϕ door het zijvlak 'e' kunnen we schrijven als:

$$F_e^c = \int_{S_e} \rho \phi v \cdot n dS \approx \dot{m}_e \phi_e. \tag{3.12}$$

waarbij $\dot{m}_e$ de massaflux die door het zijvlak 'e' stroomt en de variabele ϕ_e is de snelheidsvector op het zijvlak 'e'. We kunnen de flux door het zijvlak 'e' schrijven als: $\dot{m}_e = \int_{S_e} \rho v \cdot n dS = (\rho u)_e \Delta y$, waarbij u de snelheid is in de x-richting op zijvlak 'e'.

Benaderen van ϕ_e in (3.12) met UDS geeft:

$$F_e^c = max(\dot{m}_e, 0)\phi_P + min(\dot{m}_e, 0)\phi_E, \tag{3.13}$$

Op analoge wijze volgt voor de andere zijvlakken w(est), n(orth) en s(outh):

$$F_w^c = max(\dot{m}_w, 0)\phi_W + min(\dot{m}_w, 0)\phi_P, \tag{3.14}$$

$$F_n^c = max(\dot{m}_n, 0)\phi_P + min(\dot{m}_n, 0)\phi_N, \tag{3.15}$$

$$F_s^c = max(\dot{m}_s, 0)\phi_S + min(\dot{m}_s, 0)\phi_P. \tag{3.16}$$

Benaderen van ϕ_e in (3.12) met CDS geeft

$$F_e^c = \dot{m}_e(1 - \lambda_e)\phi_P + \dot{m}_e\lambda_e\phi_E, \tag{3.17}$$

waarbij λ_e gedefinieerd zoals in (3.7). Op analoge wijze volgt voor de andere randen (w(est), n(orth) en s(outh)):

$$F_w^c = \dot{m}_w(1 - \lambda_w)\phi_P + \dot{m}_w\lambda_w\phi_W\,, \tag{3.18}$$

$$F_n^c = \dot{m}_n(1 - \lambda_n)\phi_P + \dot{m}_n\lambda_n\phi_N, \tag{3.19}$$

$$F_s^c = \dot{m}_s(1 - \lambda_s)\phi_P + \dot{m}_s\lambda_s\phi_S\,. \tag{3.20}$$

De convectie term in (3.11) kunnen we schrijven als:

UDS interpolatie geeft de volgende coefficienten :

$$A_P^c\phi_P = A_E^c\phi_E + A_W^c\phi_W + A_N^c\phi_N + A_S^c\phi_S\,. \tag{3.21}$$

met

$$A_E^c = min(\dot{m}_e, 0), \tag{3.22}$$
$$A_W^c = min(\dot{m}_w, 0), \tag{3.23}$$
$$A_N^c = min(\dot{m}_n, 0), \tag{3.24}$$
$$A_S^c = min(\dot{m}_s, 0), \tag{3.25}$$
$$A_P^c = (A_E^c + A_W^c + A_N^c + A_S^c). \tag{3.26}$$

CDS interpolatie:
Er geldt:

$$F_e^c + F_w^c + F_s^c + F_n^c = (\sum_c \dot{m}_c)\phi_P - (\sum_c \dot{m}_c\lambda_c)\phi_P + \\ + (\lambda_e\dot{m}_e\phi_E + \lambda_w\dot{m}_w\phi_W + \lambda_n\dot{m}_n\phi_N + \lambda_s\dot{m}_s\phi_S), \tag{3.27}$$

waarbij $c = e, w, n, s$. In vergelijking (3.27) is de eerste term in het rechterlid nul en dat volgt uit de wet van massabehoud. De vergelijking (3.27) wordt korter geschreven als:

$$\sum_c F_c = -A_P^c\phi_P + \sum_c A_{nb}^c\phi_{nb}, \tag{3.28}$$

waarbij

$$A_E^c = \dot{m}_e\lambda_e, \tag{3.29}$$
$$A_W^c = \dot{m}_w\lambda_w, \tag{3.30}$$
$$A_N^c = \dot{m}_n\lambda_n, \tag{3.31}$$
$$A_S^c = \dot{m}_s\lambda_s, \tag{3.32}$$
$$A_P^c = (A_E^c + A_W^c + A_N^c + A_S^c), \tag{3.33}$$

De diffusie term

Voor de diffusie term krijgen we:

$$F_e^d = \int_{S_e} \Gamma_\phi grad\phi \cdot ndS \approx (\Gamma\frac{\partial\phi}{\partial x})_e\Delta y. \tag{3.34}$$

De afgeleide op de zijvlak 'e' bepalen we met CDS dan volgt:

$$F_e^d \approx \frac{\Gamma_\phi\Delta y}{x_E - x_P}(\phi_E - \phi_P), \tag{3.35}$$

De diffusie termen voor de andere zijvlakken gaat op analoge wijze:

$$F_w^d \approx \frac{\Gamma_\phi\Delta y}{x_P - x_W}(\phi_P - \phi_W), \tag{3.36}$$

$$F_n^d \approx \frac{\Gamma_\phi \Delta x}{x_N - x_P}(\phi_N - \phi_P), \tag{3.37}$$

$$F_s^d \approx \frac{\Gamma_\phi \Delta x}{x_P - x_S}(\phi_P - \phi_S). \tag{3.38}$$

De diffusie term in (3.11) kunnen we schrijven als:

$$A_P^d \phi_P = A_E^d \phi_E + A_W^d \phi_W + A_N^d \phi_N + A_S^d \phi_S \tag{3.39}$$

met

$$\begin{aligned}
A_E^d &= -\frac{\Gamma_\phi \Delta y}{x_E - x_P}, && (3.40)\\
A_W^d &= -\frac{\Gamma_\phi \Delta y}{x_P - x_W}, && (3.41)\\
A_N^d &= -\frac{\Gamma_\phi \Delta x}{x_N - x_P}, && (3.42)\\
A_S^d &= -\frac{\Gamma_\phi \Delta x}{x_P - x_S}, && (3.43)\\
A_P^d &= (A_E^d + A_W^d + A_N^d + A_S^d). && (3.44)
\end{aligned}$$

De bronterm

Voor de bronterm geldt:

$$Q_p = \int_S S_\phi dS \approx (S_C + S_P \phi_P)\Delta V_P = (S_C + S_P \phi_P)\Delta x \Delta y. \tag{3.45}$$

waarbij de bronterm gelineariseerd is.
De convectie-diffusie vergelijking in (3.11) kunnen we als volgt schrijven:

$$A_P \phi_P = A_E \phi_E + A_W \phi_W + A_N \phi_N + A_S \phi_S + b_P \tag{3.46}$$

met

$$\begin{aligned}
A_E &= A_E^c + A_E^d, && (3.47)\\
A_W &= A_W^c + A_W^d, && (3.48)\\
A_N &= A_N^c + A_N^d, && (3.49)\\
A_S &= A_S^c + A_S^d, && (3.50)\\
A_P &= A_P^c + A_P^d. && (3.51)
\end{aligned}$$

Het rechterlid b_P bevat de termen, die expliciet zijn bepaald. Het rechterlid b_P is dan te schrijven als:

$$b_P = S_C \Delta x \Delta y + b_e + b_w + b_s + b_n. \tag{3.52}$$

waarbij :

$$b_e = \beta(F_e^{CDS} - F_e^{UDS})^{vorige}, \tag{3.53}$$
$$b_w = \beta(F_w^{CDS} - F_w^{UDS})^{vorige}, \tag{3.54}$$
$$b_n = \beta(F_n^{CDS} - F_n^{UDS})^{vorige}, \tag{3.55}$$
$$b_s = \beta(F_s^{CDS} - F_s^{UDS})^{vorige}. \tag{3.56}$$

De vergelijking (3.46) wordt korter geschreven als :

$$A_P\phi_P = \sum_{nb} A_{nb}\phi_{nb} + b_P, \tag{3.57}$$

waarbij nb de buurpunten (E, W, S, N) van het punt P.

3.1.2 Niet-orthogonaal rooster

In het geval dat het rekendomein van de stroming gecompliceerd is, wordt vaak gebruik gemaakt van een niet-orthogonaal rooster. De voordelen van zo'n rooster zijn ondermeer:

- Het kan makkelijk aangepast worden aan elk rekendomein,
- Het inbedden van de randvoorwaarden in het rekendomein is makkelijk te implementeren.

Maar er zijn ook nadelen :

- De gediscretiseerde vergelijkingen, die wij uiteindelijk willen oplossen bevatten extra termen, die meer rekenwerk vergen en deze termen brengen implementatie moeilijkheden met zich mee,
- Een belangrijke nadeel is dat de effecten van een niet orthogonaal rooster numerieke oscillaties in de oplossing kunnen genereren (zie [4], blz. 224-228).

We zullen om oscillaties te elimineren, gebruik maken van deferred correctie. We beschouwen nu de tijdsafhankelijke convectie-diffusie vergelijking (3.1) in $3D$.

De diffusie term

De diffusie flux door de zijvlak 'e' wordt met midpuntregel als volgt benaderd:

$$F_e^d = \int_{S_e} \Gamma_\phi \nabla\phi \cdot n dS \approx (\Gamma_\phi \nabla\phi \cdot n)_e S_e. \tag{3.58}$$

De vergelijking (3.58) voor een niet-orthogonaal rooster kunnen we door gebruik te maken van deferred correctie als volgt schrijven:

$$F_e^d = \Gamma_\phi S_e (\frac{\partial \phi}{\partial \xi})_e + \Gamma_\phi S_e [\overline{(\frac{\partial \phi}{\partial n})_e} - \overline{(\frac{\partial \phi}{\partial \xi})_e})]^{vorige}. \tag{3.59}$$

waarbij n is de normaal vector op het oppervlak S_e en ξ is de richting van de vector PE, zie figuur 3.2. De eerste term van het rechterlid in (3.59) geeft de matrixcoefficienten. De tweede term, deferred correctie, aangegeven met superscript 'vorige' is afkomstig van de vorige iteratie en deze term wordt in het rechterlid b_P gestopt. De deferred correctie wordt met de geinterpoleerde gradiënt van de variabele ϕ op zijvlak 'e' bepaald, namelijk

$$\overline{(\frac{\partial \phi}{\partial n})_e} = \overline{(\nabla \phi)_e} \cdot n_e, \tag{3.60}$$

$$\overline{(\frac{\partial \phi}{\partial \xi})_e} = \overline{(\nabla \phi)_e} \cdot \frac{PE}{|PE|}, \tag{3.61}$$

De uiteindelijke uitdrukking voor de benadering van de diffusieflux, waarbij de eerste term in (3.59) met een centraal differentie schema wordt benaderd:

$$F_e^d = \Gamma_\phi S_e \frac{\phi_E - \phi_P}{|PE|} + \Gamma_\phi S_e [\overline{(\nabla \phi)_e} \cdot n_e - \overline{(\nabla \phi)_e} \cdot \frac{PE}{|PE|}]^{vorige}. \tag{3.62}$$

Op analoge wijze worden de fluxen voor de andere zijvlakken bepaald.

De convectie term

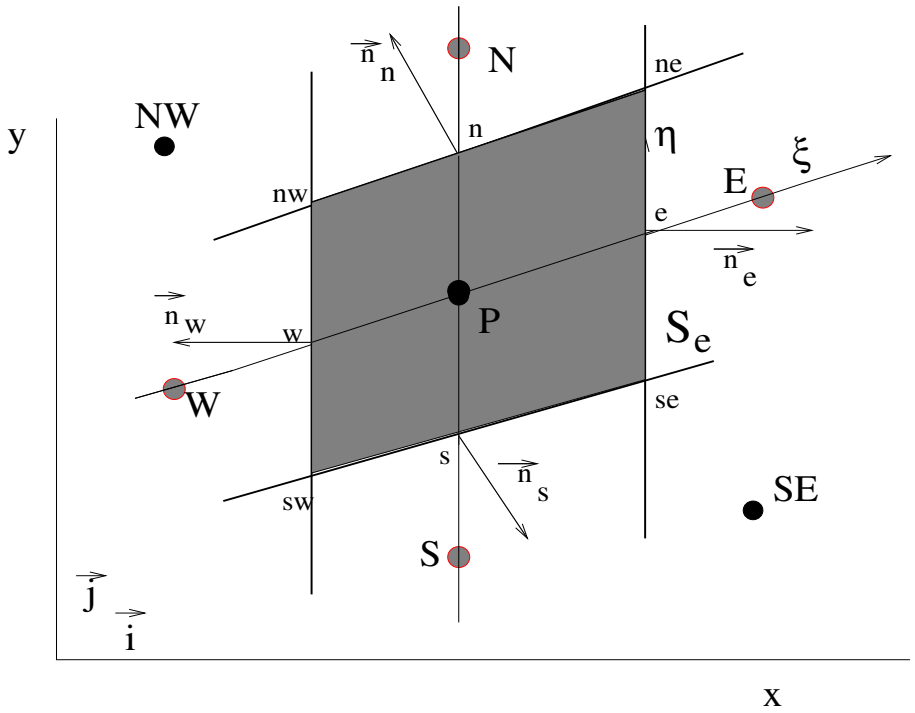
De convectie flux van een variabele ϕ door zijvlak 'e', zie figuur 3.2 is te schrijven als:

$$F_e^c = \int_{S_e} \rho \phi v \cdot n dS \approx \dot{m}_e \phi_e. \tag{3.63}$$

waarbij $\dot{m}_e$ de massaflux door zijvlak 'e' en ϕ_e is het gemiddelde waarde van ϕ op zijvlak 'e'. De massaflux is als volgt gedefinieerd :

$$\dot{m}_e = \int_{S_e} \rho v \cdot n dS \approx (\rho v \cdot n)_e S_e. \tag{3.64}$$

waarbij



Figuur 3.2: Een niet-orthogonaal controle volume

$$n_e S_e = S_e^x i + S_e^y j. \tag{3.65}$$

De oppervlakten S_e^x en S_e^y zijn als volgt gedefinieerd :

$$S_e^x = (y_n - y_s)_e, \tag{3.66}$$
$$S_e^y = -(x_n - x_s)_e. \tag{3.67}$$

Het oppervlak S_e wordt:

$$S_e = \sqrt{(S_e^x)^2 + (S_e^y)^2}. \tag{3.68}$$

De massaflux door zijvlak 'e' wordt dan :

$$\dot{m}_e = \rho_e (u S_e^x + v S_e^y)_e. \tag{3.69}$$

waarbij u en v de snelheidscomponenten zijn in x-richting respectievelijk y-richting.
We benaderen de variabele ϕ_e door:

$$\phi_e = \phi_e^{UDS} + \beta(\phi_e^{CDS} - \phi_e^{UDS})^{vorige}. \tag{3.70}$$

Op analoge wijze worden de fluxen door de andere zijvlakken benadert.

De bronterm

De bronterm wordt :

$$Q_p = \int_S S_\phi dS \approx (S_C + S_P \phi_P) \Delta V_P. \tag{3.71}$$

waarbij ΔV_P de volume van een Cv en deze volume is voor $2D$
als volgt gedefinieerd:

$$\Delta V = \frac{1}{2}[(r_{ne} - r_{sw}) \times (r_{nw} - r_{se})] = \frac{1}{2}[(x_{ne} - x_{sw})(y_{nw} - y_{se}) - (y_{ne} - y_{sw})(x_{ne} - x_{sw})]. \tag{3.72}$$

waarbij bijvoorbeeld r_{ne} de afstandsvector is van de oorsprong tot het punt ne.

Tijdsdiscretiseren

Voor de tijdsdiscretiseren wordt het volledige impliciete tijdsschema gebruikt.

$$\int_V \frac{\partial \rho \phi}{\partial t} dV = (\phi_P - \phi_P^{t-\Delta t}) \frac{\Delta V}{\Delta t}. \tag{3.73}$$

waarbij Δt de tijdstap is en $(t - \Delta t)$ duidt het vorige tijdsniveau aan. Na het samenvoegen van de diffusie, convectie, bronterm en de tijdsafhankelijke term krijgen we een algebraische vergelijking

$$A_P \phi_P = \sum_{nb} A_{nb} \phi_{nb} + b_P \tag{3.74}$$

waarij de coefficienten A_P, A_{nb} en het rechterlid b_P zijn dan als volgt gedefinieerd :

$$A_E = \Gamma_\phi \frac{S_e}{|PE|} + min(\dot{m}_e, 0), \tag{3.75}$$

$$A_W = \Gamma_\phi \frac{S_w}{|PW|} + min(\dot{m}_w, 0), \tag{3.76}$$

$$A_N = \Gamma_\phi \frac{S_n}{|PN|} + min(\dot{m}_n, 0), \tag{3.77}$$

$$A_S = \Gamma_\phi \frac{S_s}{|PS|} + min(\dot{m}_s, 0), \tag{3.78}$$

$$A_T = \Gamma_\phi \frac{S_t}{|PT|} + min(\dot{m}_t, 0), \tag{3.79}$$

$$A_B = \Gamma_\phi \frac{S_b}{|PB|} + min(\dot{m}_b, 0), \tag{3.80}$$

$$A_P = A_E + A_W + A_N + A_S + A_T + A_B - S_P \Delta V_P + \frac{\Delta V}{\Delta t}. \tag{3.81}$$

Voor het rechterlid b_P bevat de termen die uit de vorige iteratie zijn bepaald en de gelineariseerde term van de bronterm ofwel :

$$b_P = S_C \Delta v_P + \frac{\Delta V}{\Delta t} \phi_P^{t-\Delta t} + b_e + b_w + b_n + b_s + b_t + b_b. \tag{3.82}$$

waarbij :

$$b_e = \Gamma_\phi S_e [\overline{(\nabla\phi)_e} \cdot n_e - \overline{(\nabla\phi)_e} \cdot \frac{PE}{|PE|}] + \beta(F_e^{CDS} - F_e^{UDS}), \tag{3.83}$$

$$b_w = \Gamma_\phi S_w [\overline{(\nabla\phi)_w} \cdot n_w - \overline{(\nabla\phi)_w} \cdot \frac{PW}{|PW|}] + \beta(F_w^{CDS} - F_w^{UDS}), \tag{3.84}$$

$$b_n = \Gamma_\phi S_n [\overline{(\nabla\phi)_n} \cdot n_n - \overline{(\nabla\phi)_n} \cdot \frac{PN}{|PN|}] + \beta(F_n^{CDS} - F_n^{UDS}), \tag{3.85}$$

$$b_s = \Gamma_\phi S_s [\overline{(\nabla\phi)_s} \cdot n_s - \overline{(\nabla\phi)_s} \cdot \frac{PS}{|PS|}] + \beta(F_s^{CDS} - F_s^{UDS}), \tag{3.86}$$

$$b_t = \Gamma_\phi S_t [\overline{(\nabla\phi)_t} \cdot n_t - \overline{(\nabla\phi)_t} \cdot \frac{PT}{|PT|}] + \beta(F_t^{CDS} - F_t^{UDS}), \tag{3.87}$$

$$b_b = \Gamma_\phi S_b [\overline{(\nabla\phi)_b} \cdot n_b - \overline{(\nabla\phi)_b} \cdot \frac{PB}{|PB|}] + \beta(F_e^{CDS} - F_e^{UDS}). \tag{3.88}$$

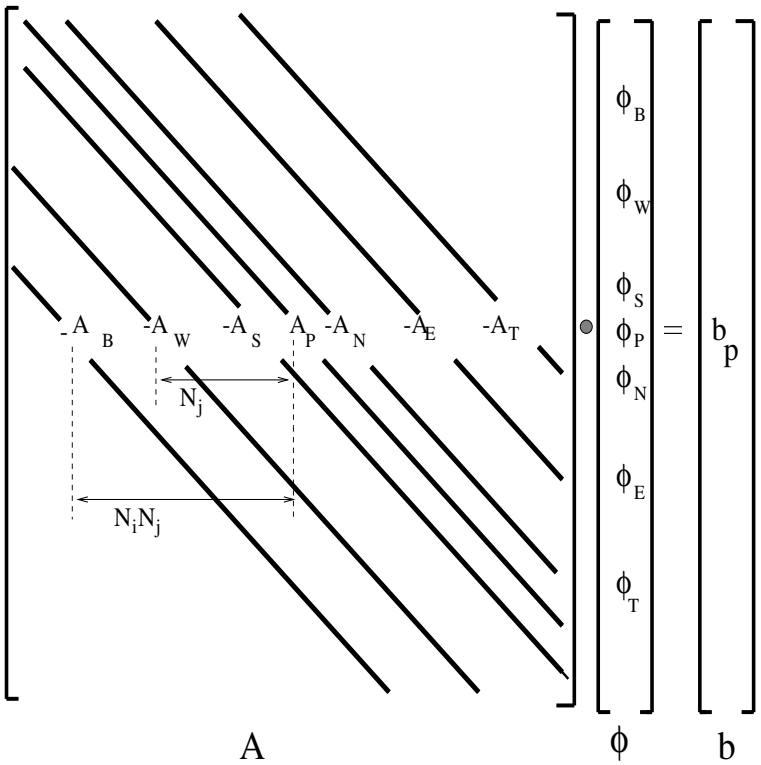
We kunnen (3.74) in matrix vector notatie schrijven. Stel dat het roosterpunt ϕ_P met lokale referentie overkomt met ϕ_l dan wordt de vergelijking (3.74)

$$\begin{aligned} &A_{l,l-N_iN_j}\phi_{l-N_iN_j} + A_{l,l-N_j}\phi_{l-N_j} + A_{l,l-1}\phi_{l-1} + A_{l,l}\phi_l + \\ &\quad + A_{l,l+1}\phi_{l+1} + A_{l,l+N_j}\phi_{l+N_j} + A_{l,l+N_iN_j}\phi_{l+N_iN_j} = b_l. \end{aligned} \tag{3.89}$$

roosterpunt	*notatie*	*geheugenlokatie*
i, j, k	P	$l = (k-1)N_jN_i + (i-1)N_j + j$
$i-1, j, k$	W	$l - N_j$
$i, j-1, k$	S	$l - 1$
$i, j+1, k$	N	$l + 1$
$i+1, j, k$	E	$l + N_j$
$i, j, k-1$	B	$l - N_iN_j$
$i, j, k+1$	T	$l + N_iN_j$

Tabel 3.1: index tabel

waarbij l de index is van één-dimensionale array, waarin de elementen van de vector ϕ zijn opgeslagen. Aan de hand van de onderstaande tabel kunnen we door middel van de roosterpunten (i, j, k) de index l achterhalen. Op deze manier, dus het opslaan van vijf vectoren in $2D$ of zeven vectoren in $3D$ van matrix A in een één-dimensionale array, besparen we op de geheugenruimte in de computer.



Figuur 3.3: Het stelsel vergelijkingen

De vergelijking (3.89) kunnen we voor alle roosterpunten schrijven als $A\phi = b_P$, waarbij A een matrix is en ϕ is een vector die de onbekenden bevat, zie figuur 3.3.

Hoofdstuk 4

Numerieke methoden

4.1 De SIP solver

In hoofdstuk 3 hebben we het stelsel lineaire vergelijkingen verkregen, dat ontstaat is na het discretiseren van de Navier-Stokes vergelijkingen:

$$A\phi = b, \tag{4.1}$$

waarbij de matrix A ijl is, hetgeen inhoudt dat slechts enkele elementen van A niet nul zijn. Numerieke oplosmethoden voor (4.1) zullen deze eigenschappen moeten uitbuiten, daar de stelsels die in de praktijk voorkomen vaak zeer groot zijn. Voor de numerieke oplossing zijn verschillende directe methoden en iteratieve methoden beschikbaar. Bij een directe methode is het uitbuiten van ijlheid niet eenvoudig. Iteratieve methoden hebben de voorkeur bij zeer grote aantal onbekenden. De structuur van A biedt ons de mogelijkheid om iteratieve methoden efficïent te gebruiken.
Na n iteraties krijgen we een benaderende oplossing ϕ^n voor ϕ in (4.1). Invullen van ϕ^n in (4.1) houden we het residu r^n over :

$$A\phi^n = b - r^n. \tag{4.2}$$

De doelstelling is om met een iteratieve proces het residu r^n naar nul te drijven. Een veel gebruikte methode in CFD (Computational Fluid Dynamics) is de SIP solver (Strongly Implictit Procedure, zie [6]).
Een algemenere iteratieve methode voor (4.1) is :

$$M\phi^{n+1} = N\phi^n + b, \tag{4.3}$$

met $A = M - N$ en n is de n^{de} iteratie en $n = 0, 1, 2$.....
Hier zijn M en N ook ijle matrices. Als wij $M\phi^n$ van beide kanten van (4.3) aftrekken dan volgt :

$$M(\phi^{n+1} - \phi^n) = A\phi - A\phi^n = b - (M - N)\phi^n = r^n, \tag{4.4}$$

of

$$M\delta^{n+1} = r^n, \tag{4.5}$$

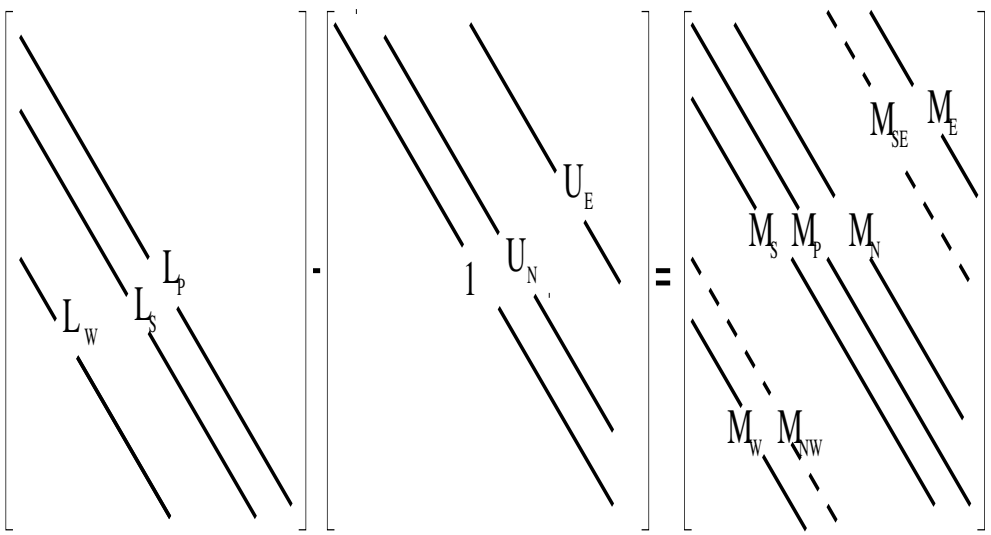
met

$$\delta^{n+1} = \phi^{n+1} - \phi^n, \tag{4.6}$$

De vergelijking (4.6) wordt de correctie of update genoemd. De term r^n is het residu en is voor de duidelijkheid gedefinieerd door:

$$r^n = b - (M - N)\phi^n. \tag{4.7}$$

We ontbinden de matrix M in L en U in een twee-dimensionaal probleem, zie figuur 4.1, waarbij de hoofddiagonaal van U enen bevat. Het produkt LU geeft twee extra diagonalen ten opzichte van de matrix A. Dit produkt leidt tot :



Figuur 4.1: LU-onbinding van matrix M

$$M_W^l = L_W^l, \tag{4.8}$$
$$M_{NW}^l = L_W^l U_N^{l-N_j}, \tag{4.9}$$
$$M_S^l = L_S^l, \tag{4.10}$$
$$M_P^l = L_W^l U_E^{l-N_j} + L_S^l U_N^{l-1} + L_P^l, \tag{4.11}$$
$$M_N^l = U_N^l L_P^l, \tag{4.12}$$
$$M_{SE} = L_S^l U_E^{l-1}, \tag{4.13}$$
$$M_E^l = U_E^l L_P^l, \tag{4.14}$$

waarbij l de index van de geheugenlokatie van de elementen van M in een één-dimensionale array. Om convergentie van (4.3) te versnellen, willen we L en U zodanig selecteren dat M erg veel op A lijkt. Er geldt :

$$(M\phi)_P = \sum_{nb} M_{nb}\phi_{nb} + M_{NW}\phi_{NW} + M_{SE}\phi_{SE}, \tag{4.15}$$

De laatste twee termen van (4.15) zijn de extra diagonalen. Matrix N bevat zeven niet nul diagonalen en heeft de zelfde struktuur als M. Matrix N bevat ook de twee extra diagonalen M_{NW} en M_{SE}. Om matrix A goed te benaderen met M, willen we de bijdrage van N in (4.3) bijna nul hebben, liefst nul, met andere woorden:

$$\sum_{nb} N_{nb}\phi_{nb} + M_{NW}\phi_{NW} + M_{SE}\phi_{SE} \approx 0. \tag{4.16}$$

Dit is mogelijk als de bijdrage van de twee extra diagonalen in (4.16) door de andere diagonalen geëlimineerd wordt. We schrijven nu (4.16) als:

$$M_{NW}(\phi_{NW} - \phi_{NW}^*) + M_{SE}(\phi_{SE} - \phi_{SE}^*) \approx 0, \tag{4.17}$$

waarbij ϕ_{NW}^* en ϕ_{SE}^* benaderigen zijn van ϕ_{NW} en ϕ_{SE}. Om (4.17) te realiseren moeten we ϕ_{NW}^* en ϕ_{SE}^* uitdrukken in de variabelen ϕ_{nb} met nb=W, E, S, N en P. Een mogelijke benadering voorgestelt door Stone is als volgt:

$$\phi_{NW}^* \approx \alpha(\phi_W + \phi_N - \phi_P), \tag{4.18}$$

$$\phi_{SE}^* \approx \alpha(\phi_S + \phi_E - \phi_P), \tag{4.19}$$

Hier is de parameter α een relexatiefactor en wordt om stabiliteit redenen $\alpha \leq 1$ gekozen. Voor $\alpha = 1$ zijn de benaderingen (4.18) en (4.19) lineair interpolaties in

$2D$. Het kiezen van optimale α is probleem afhankelijk en deze parameter wordt in de stromingsproblemen vaak 0.92 gekozen. Als wij (4.18) en (4.19) in (4.17) vullen dan volgt :

$$M_{NW}\phi_{NW} - \alpha M_{NW}(\phi_W + \phi_N - \phi_P) + M_{SE}\phi_{SE} - \alpha M_{SE}(\phi_S + \phi_E - \phi_P) \approx 0. \tag{4.20}$$

We stellen (4.16) gelijk aan (4.20) stellen dan volgt :

$$\sum_{nb} N_{nb}\phi_{nb} \approx -\alpha M_{NW}(\phi_W + \phi_N - \phi_P) - \alpha M_{SE}(\phi_S + + \phi_E - \phi_P) + M_{NW}\phi_{NW} + M_{SE}\phi_{SE}. \tag{4.21}$$

We kunnen nu met de relatie $M\phi = A\phi + N\phi$ de elementen van L en U bepalen. Er geldt:

$$\sum_{nb} M_{Nb}\phi_{nb} = \sum_{nb} A_{nb}\phi_{nb} + \alpha M_{NW}(\phi_W + \phi_N - \phi_P) + + \alpha M_{SE}(\phi_S + \phi_E - \phi_P), \tag{4.22}$$

De elementen van de matrices L en U behoeven we slechts een keer te bepalen en deze zijn door gebruik te maken van (4.8 - 4.14):

$$L_W = A_W - \alpha M_{NW} = A_W - \alpha L_W U_N, \tag{4.23}$$

Uit (4.23) volgt de uitdrukking voor L_W, namelijk:

$$L_W^l = -\frac{A_W^l}{1 + \alpha U_N^{l-N_j}}, \tag{4.24}$$

Op analoge wijze volgen de andere termen :

$$L_S^l = -\frac{A_S^l}{1 + \alpha U_E^{l-1}}, \tag{4.25}$$

$$L_P^l = A_P^l + \alpha(L_W^l U_N^{l-N_j} + L_S^l U_E^{l-1}) - L_W^l U_E^{l-N_j} - L_S^l U_N^{l-1}, \tag{4.26}$$

$$U_N^l = \frac{A_N^l - \alpha L_W^l U_N^{l-N_j}}{L_P^l}, \tag{4.27}$$

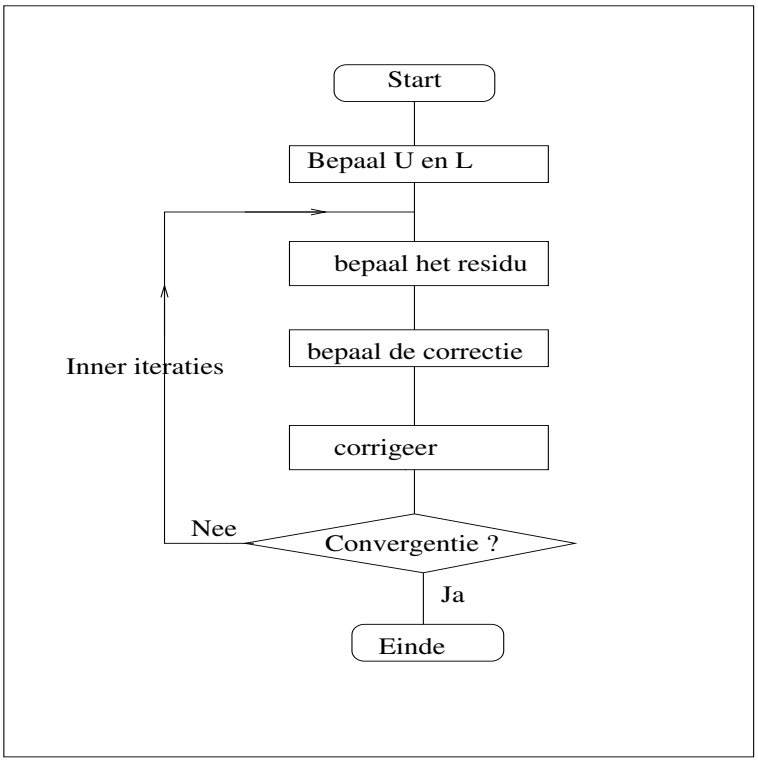
$$U_E^l = \frac{A_E^l - \alpha L_S^l U_E^{l-1}}{L_P^l}. \tag{4.28}$$

Nu kunnen we (4.5) op lossen.

$$U\delta^{n+1} = L^{-1} r^n = z^n, \tag{4.29}$$

waarbij z^n een hulpvector is.
De elementen van z^n in (4.29) zijn te bepalen door :

$$z^l = (r^l - L_S^l z^{l-1} - L_W^l z^{l-N_j}) \frac{1}{L_P^l}. \tag{4.30}$$



Figuur 4.2: Flowdiagram van de SIP solver

Na het bepalen van z in (4.30), kunnen we de correctie bepalen door $\delta^{n+1} = U^{-1} z^n$ op te lossen. Er volgt:

$$\delta^l = (z^l - U_N^l \delta^{l+1} - U_E^l \delta^{l+N_j}). \tag{4.31}$$

In figuur 4.2 is aan de hand van een flowdiagram de SIP solver schematisch weergegeven.

Hoofdstuk 5

Oplosmethode voor de Navier-Stokes vergelijkingen

Om de Navier-Stokes vergelijkingen op te lossen, zijn er twee mogelijke aanpakken.

- Alle variabelen in de Navier-Stokes vergelijkingen worden tegelijkertijd opgelost, dat wil zeggen dat de momentenvergelijkingen en de continuïteitsvergelijking beschouwd worden als één geheel systeem. De corresponderende matrix A bestaat uit een aantal blokmatrices.
- In de tweede aanpak nemen we aan dat een aantal variabelen bekend zijn en worden hiermee de gelineariseerde stelsel vergelijkingen sequentieel, iteratief opgelost. De meeste gebruikte oplosmethode is de $SIMPLE$ methode ($\underline{S}$emi $\underline{I}$mplicit $\underline{M}$ethod for $\underline{P}$ressure $\underline{L}$inked $\underline{E}$quations).

We zullen gebruik maken van de tweede aanpak, (zie [4]).

5.1 De SIMPLE methode

De impulsvergelijkingen kunnen slechts opgelost worden, indien het drukveld gegeven is. Indien slechts een schatting van het drukveld wordt gebruikt, zal het resulterende snelheidsveld na oplossen van de impulsvergelijkingen niet voldoen aan de continuïteitsvergelijking. Vervolgens kan deze continuïteitsvergelijking gebruikt worden het bijpassende drukveld te vinden. Dit is weer een nieuwe schatting van het drukveld, waarmee de impulsvergelijkingen weer kunnen worden opgelost. Op deze manier kan iteratief een snelheidsveld gevonden worden, dat zowel aan de impulsvergelijkingen als aan de continuïteitsvergelijking voldoet. Zaak is dus met behulp van de continuïteitsvergelijking een vergelijking voor de druk af te leiden. Gemakshalve gebruiken we hier Cartesische coordinaten. De gediscretiseerde impulsvergelijking is te schrijven als :

$$A_P^{u_i} U_{i,P}^{m*} = \tilde{b}_{i,P}^{m-1} - \sum A_l^{u_i} U_{i,l}^{*} - \Delta V_P (\frac{\partial p^{m-1}}{\partial x_i})_P, \tag{5.1}$$

waarbij m is de m^{de} iteratie is, ΔV_P is het volume van een cel met centrum P. Hier is de drukgradiënt in het rechterlid b_P uitgehaald en wordt expliciet getoond. De term met (*) in (5.1) geeft aan dat de snelheid nog niet aan de continuïteitsvergelijking voldoet. Deze snelheid moet nog gecorrigeerd worden. In (5.1) delen we door de term $A_P^{u_i}$ en vervolgens kunnen we de eerste term van het rechterlid van de verkregen vergelijking met $\tilde{U}_{i,P}^{m*}$ aangeven. In deze term zit geen bijdrage van de drukgradiënt. De vergelijking (5.1) wordt dan:

$$U_{i,P}^{m*} = \tilde{U}_{i,P}^{m*} - \frac{\Delta V_P}{A_P^{u_i}} (\frac{\partial p^{m-1}}{\partial x_i})_P. \tag{5.2}$$

Als wij de snelheid en druk in P corrigeren dan schrijven we (5.2) als:

$$U_{i,P}^{m} = \tilde{U}_{i,P}^{m*} - \frac{\Delta V_P}{A_P^{u_i}} (\frac{\partial p^{m}}{\partial x_i})_P, \tag{5.3}$$

De snelheid en druk in (5.2) zijn gecorrigeerd met:

$$U_i^m = U_i^{m*} + U_i', \tag{5.4}$$

$$p^m = p^{m-1} + p', \tag{5.5}$$

waarbij U' en p' zijn de snelheid- en respectievelijk de drukcorrectie. Als wij de vergelijkingen (5.4) en (5.5) in (5.3) vullen dan volgt een relatie tussen de snelheids- en de drukcorrectie.
De relatie tussen U' en p' wordt :

$$U_{i,P}' = -\frac{\Delta V_P}{A_P^{u_i}} (\frac{\partial p'}{\partial x_i})_P. \tag{5.6}$$

Om de massaflux te bepalen, hebben we de snelheid op de zijvlakken nodig. We concentreren ons op zijvlak 'e'. Deze snelheid wordt met interpolatie bepaald. Om de oscillaties van de druk te elimeneren wordt de geinterpoleerde snelheid gecorrigeerd met het verschil van de drukgradiënt en de geinterpoleerde drukgradiënt. In formule :

$$U_{i,e}^{m*} = \overline{(U_i^{m*})}_e - \Delta V_e \overline{(\frac{1}{A_P^{u_i}})}_e ((\frac{\partial p}{\partial x_i})_e - \overline{(\frac{\partial p}{\partial x_i})}_e)^{m-1}, \tag{5.7}$$

waarbij de geinterpoleerde waarden met een streep zijn aangegeven. Het volume ΔV_e is gedefinieerd door $(PE \cdot S_e n_e) = (PE \cdot n_e) S_e$. Er blijkt dat slechts de snelheidsnormaal een bijdrage levert aan de massaflux door zijvlak 'e'. Voor de normaal snelheidscomponent $v_n = v \cdot n$ op zijvlak 'e' wordt vergelijking (5.7) dan:

$$v_{n,e}^{m*} = \overline{(v_n^{m*})}_e - \Delta V_e \overline{(\frac{1}{A_P^{v_n}})}_e ((\frac{\partial p}{\partial n})_e - \overline{(\frac{\partial p}{\partial n})_e})^{m-1}, \tag{5.8}$$

De drukgradiënt in (5.8) benaderen we met CDS, zie figuur 5.1. Er geldt:

$$(\frac{\partial p}{\partial n})_e \approx \frac{p_{E'} - p_{P'}}{|r_{E'} - r_{P'}|} = \frac{p_{E'} - p_{P'}}{(r_{E'} - r_{P'}) \cdot n}, \tag{5.9}$$

met

$$r_{P'} = r_e - [(r_e - r_P) \cdot n] n, \tag{5.10}$$

$$r_{E'} = r_e - [(r_e - r_E) \cdot n] n, \tag{5.11}$$

De druk in E' en P' benaderen we als volgt:

$$p_{P'} \approx p_P + (gradp)_P \cdot (r_{P'} - r_P), \tag{5.12}$$

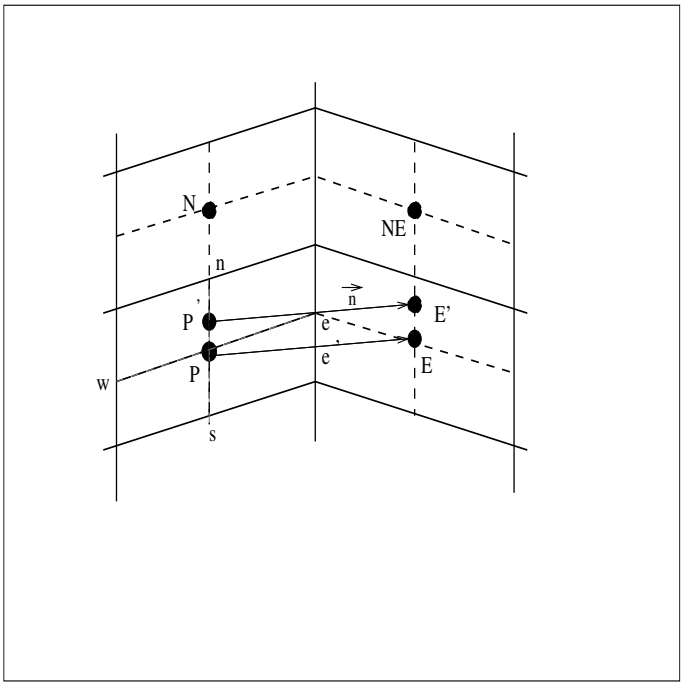
$$p_{E'} \approx p_E + (gradp)_E \cdot (r_{E'} - r_E), \tag{5.13}$$

We kunnen nu voor $(\frac{\partial p}{\partial n})_e$ schrijven:

$$(\frac{\partial p}{\partial n})_e \approx \frac{p_E - p_P}{(r_E - r_P) \cdot n} + \frac{(gradp)_E \cdot (r_{E'} - r_E) - (gradp)_P \cdot (r_{P'} - r_P)}{(r_E - r_P) \cdot n}, \tag{5.14}$$

De tweede term in het rechterlid wordt nul als het rooster orthogonaal is, want dan vallen de punten zowel E en E' als P en P' samen, zie figuur 5.1. De vergelijking (5.8) wordt dan:

$$\begin{aligned} v_{n,e}^{m*} = & \overline{(v_n^{m*})}_e \\ & - \frac{\Delta V_e}{(r_E - r_P) \cdot n} \overline{(\frac{1}{A_P^{v_n}})}_e [(p_E - p_P) + (gradp)_E \cdot (r_{E'} - r_E) \\ & - (gradp)_P \cdot (r_{P'} - r_P) - \overline{(gradp)_e} \cdot (r_E - r_P)]. \end{aligned} \tag{5.15}$$



Figuur 5.1: Non-orthogonaal cellen

Het bepalen vanaf de derde term in het rechterlid van (5.15) vergt meer rekenwerk en deze termen worden in de praktijk vaak verwaarloosd (zie [4]).
De snelheid in (5.15) gebruiken we om de massflux te bepalen, namelijk

$$\dot{m}_e^{m*} = \rho_e v_{n,e}^{m*} S_e. \tag{5.16}$$

Op analoge wijze vinden we de massaflux door de andere zijvlakken. De som van massflux voldoet niet aan de behoudswet van massa :

$$\sum_c \dot{m}_c^{m*} = \Delta\dot{m} \qquad c = e, w, n, s, \cdots \tag{5.17}$$

De som in (5.17) moet nul zijn om aan het behoud van massa te voldoen, en zal daarom gecorrigeerd worden met:

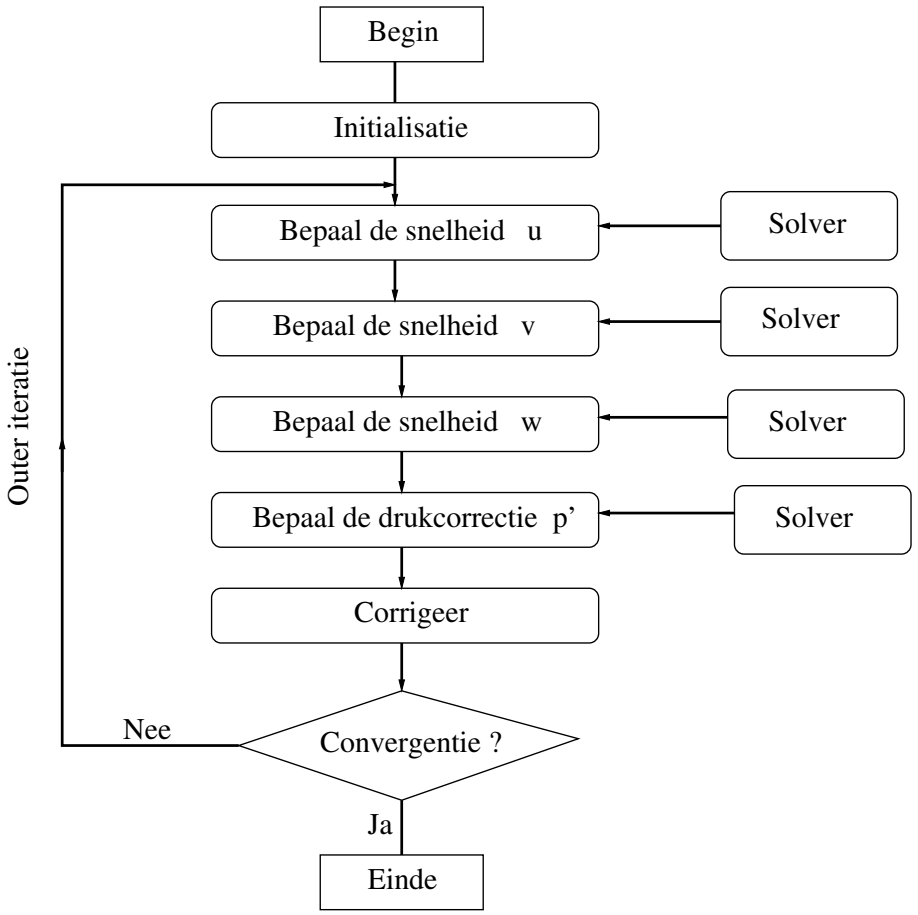
$$\begin{aligned}\dot{m}'_e = \rho_e v'_{n,e} S_e \approx -(\rho\Delta V S)_e \overline{(\frac{1}{A_P^{v_n}})}_e \, (\frac{\partial p'}{\partial n})_e \approx \\ - (\rho\Delta V S)_e \overline{(\frac{1}{A_P^{v_n}})}_e \, [(p'_E - p'_P) + (gradp')_E \cdot (r_{E'} - r_E) \\ (gradp')_P \cdot (r_{P'} - r_P) - \overline{(gradp')_e} \cdot (r_E - r_P)].\end{aligned} \tag{5.18}$$

Op analoge wijze vinden we de massacorrectie voor de andere zijvlakken, en dat de gecorrigeerde massaflux wel aan de behoud van massa voldoet, namelijk:

$$\sum_c \dot{m}'_c + \Delta\dot{m} = 0 \qquad c = e, w, n, s, \cdots \tag{5.19}$$

Op deze manier krijgen we het stelsel vergelijkingen voor de drukcorrectie met het het tweede term in (5.19) als het rechterlid ofwel

$$A_P p'_P = \sum_{nb} A_{nb} p'_{nb} + b. \tag{5.20}$$



Figuur 5.2: Flowdiagram van SIMPLE

De termen $gradp'$ in (5.18) leidt tot veel rekenwerk. Als het effect van de niet-orthogonale term niet al te groot is, wordt deze term verwaarloosd. Tijdens één outer iteratie van SIMPLE worden het stelsel vergelijkingen voor de snelheden en

de drukcorrectie iteratief opgelost. In X-stream wordt de SIP solver toegepast. Met deze oplossingen worden de druk en het snelheidsveld gecorrigeerd. In figuur 5.2 is aan de hand van een flowdiagram de outerloop van SIMPLE weergegeven. In sectie 8.5 worden nadere details van De SIMPLE methode gegeven.

Hoofdstuk 6

Multigrid methode

Multigrid (MGM) is een efficiënte iteratieve methode om een stelsel vergelijkingen op te lossen. Het stelsel vergelijkingen dat na discretisatie is onstaan, zie hoofdstuk 3, geven we aan met:

$$A^K \phi^K = b^K, \tag{6.1}$$

waarbij de index K het fijnste rooster aangeeft.
MGM maakt gebruik van de klassieke iteratieve methoden (Gauss-Seidel, Jacobi, de SIP solver, enzovoort) ook wel smoothers genoemd. Het basisidee van MGM is onstaan doordat de fout aan het begin van het iteratieproces vaak een sterk fluctuerend karakter heeft, waarbij de fluctuaties optreden over afstanden, die van de orde van de fijne maaswijdte zijn. Het blijkt dat de iteratieve methoden deze fluctuaties vaak na enkele iteraties (sweeps) wegdempen, in de zin dat alleen nog de fluctuaties van de orde van enkele maaswijdte aanwijzig zijn. Vanaf dit moment gaat het iteratieproces langzamer verlopen en wordt een overstap naar een grof rooster gemaakt. Op dit grove rooster treden er nu alleen fluctuaties op over afstanden van de orde van de grove maaswijdte, die weer door enkele slagen met een iteratieve methode weggedempt zullen worden. Dit proces zal herhaald worden tot dat er weinig roosterpunten over zijn, dan wordt het het verkregen stelsel $A^1\phi^1 = b^1$ met een directe methode of een iterative methode opgelost. Een andere eigenschap van een klassieke iteratieve methode is dat deze snel naar de exacte oplossing van (6.1) convergeert, mits er met een goede startvector begonnen wordt. Een manier om een goede startvector te krijgen is om op een grof rooster te beginnen, dus de logische gedachte is om op het grofste rooster te beginnen. Combinatie van deze twee constateringen over iteratieve methoden:

- De smoothing eigenschap na enkele iteratie slagen,
- Kies een startvector op het grofste rooster,

leidt tot het V-cycle algoritme. Bij MGM wordt in het geval van vertex discretisatie gebruik gemaakt van de rekenroosters:

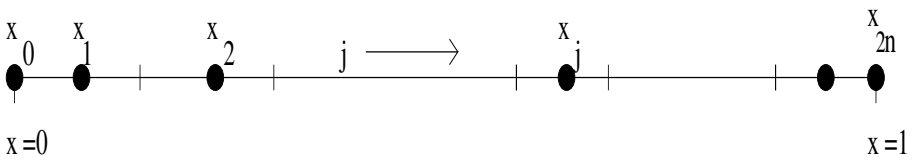
$G^k \quad k = 1, 2 \ldots\ldots, K \qquad$ waarbij $G^1 \subset G^2 \subset G^3 \ldots \cdots \subset G^K$

met G^K als het fijnste rooster waarop de PDV opgelost zal worden. In het geval van Cell-Centred discretisatie is het rekenrooster G gedefinieerd als:

$G = \{x \in \mathbf{R}^d : x = (j-s)h; j = (j_1, j_2, \ldots, j_d; s = \frac{(1,\ldots,1)}{2}; h = (h_1, \ldots, h_d); j_\alpha = 1, 2, \ldots n_\alpha; h_\alpha = \frac{1}{n_\alpha}, \alpha = 1, 2, \ldots, d\}$,

waarbij d het aantal dimensies voorstelt. Voor $d = 1$ is G, zie figuur 6.1:

$$G = \{x_j \in \mathbf{R} : x_j = (j - \frac{1}{2})h, j = 1, 2, ...2n, h = \frac{1}{2n}\}, \tag{6.2}$$



Figuur 6.1: Cell-centred discretisatie

De discretisatie van de PDV op het fijnste rooster G^K leidt tot

$$A^K \phi^K = b^K. \tag{6.3}$$

We geven de verzameling van de roosterfuncties op G^k aan met U^k. In MGM zullen we vaak van een fijn rooster naar een grof rooster over moeten stappen en ook andersom. Om informatie tussen de roosters G^k te kunnen uitwisselen, hebben we nodig:
- De restrictie operator $R^k : U^k \longrightarrow U^{k-1}$ en
- De prolongatie operator $P^k : U^{k-1} \longrightarrow U^k$.
Tussen deze twee operatoren kan een relatie bestaan: $R^k = (P^k)^*$. Hier is R^k de geadjungeerde van P^k ofwel

$$(R^k u^k, v^{k-1})_{k-1} = (u^k, P^k v^{k-1})_k, \tag{6.4}$$

$\forall v^{k-1} \in U^{k-1}$ en $\forall u^k \in U^k$.

Op G^K is het stelsel (6.3) gegeven. Uit dit stelsel zullen we met de operatoren P en R het nieuwe stelsel vinden, dat geldig is op het grove rooster. Dit nieuwe stelsel bepalen we met de Galerkin benadering (Galerkin Coarse grid approximation, GCA). Het stelsel op het k^{de} rooster noteren we met :

$$A^k \phi^k = b^k. \tag{6.5}$$

Stel dat ϕ^k een benadering is (verkregen na enkele relaxatieslagen) van de oplossing ϕ^k_{ex} van (6.3). We definïeren de correctie:

$$e^k = \phi^k_{ex} - \phi^k. \tag{6.6}$$

De correctie voldoet aan:

$$A^k e^k = r^k, \tag{6.7}$$

waarbij $r^k = b^k - A^k \phi^k$. We zullen (6.7) bij benadering oplossen via de overgang op het grove rooster. Het oplossen van (6.7) is equivalent met het bepalen van $e^k \in U^k$ zodanig dat :

$$(A^k e^k, v^k)_k = (r^k, v^k)_k \qquad \forall v^k \in U^k$$

De Galerking benadering leidt tot het stelsel:

$$A^{k-1} e^{k-1} = r^{k-1}. \tag{6.8}$$

op rooster G^{k-1} met $A^{k-1} = R^k A^k P^k$ en $b^{k-1} = R^k b^k$. We zullen MGM aan de hand van een één-dimensionaal probleem illustreren. We nemen het volgende model:

$$-\frac{d^2u}{dx^2} = f(x) \qquad \Omega = (0,1) \tag{6.9}$$

Voor de randen hebben we u(0)=$\frac{du}{dx}(1) = 0$ gekozen. We zullen de vergelijking (6.9) met een EVM discretiseren. We beschouwen een geschikt gekozen deeldomein (Cv) rond elke roosterpunt x_i. We integreren (6.9) over dit deeldomein. Voor ons model kiezen we als deeldomein bij x_i het segment $\omega_i = [x_i - \frac{h}{2}, x_i + \frac{h}{2}]$. Integratie van (6.9) over ω_i levert:

$$\int_{\omega_i} -\frac{d^2u}{dx^2} dx - \int_{\omega_i} f(x) dx = -\frac{1}{h}(\frac{du}{dx}(x_i + 0.5h) - \frac{du}{dx}(x_i - 0.5h)) - hf(x_i), \tag{6.10}$$

Beide afgeleiden worden met centrale differentie methode benaderd en dan volgt de volgende differentie formule:

$$-u_{i-1} + 2u_i - u_{i+1} = h^2 f_i, \qquad i = 2, 3, 4 \cdots\cdots, 2n-1 \tag{6.11}$$

Voor de randen:

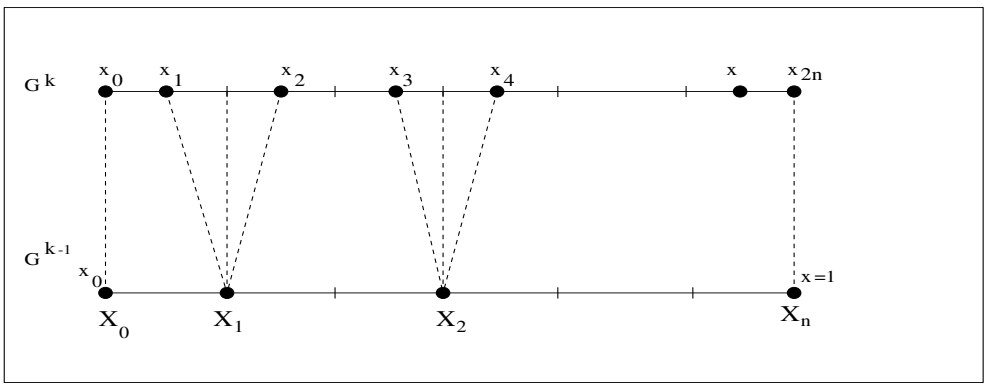
$$3u_1 - u_2 = h^2 f_1 \qquad op \;\; x = 0, \tag{6.12}$$

$$-u_{2n-1} + u_{2n} = h^2 f_{2n} \qquad op \;\; x = 1 \tag{6.13}$$

Het stelsel $A^K \phi^K = b^K$ wordt dan:

$$A^K = \begin{pmatrix} 3 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{pmatrix}.$$

en de kolomvector $b^K = (h^2 f_1, \cdots\cdots, h^2 f_{2n})$, waarbij A^K een $(2n \times 2n) - matrix$ is en b^K is een $(2n \times 1) - vector$ is. Om nieuwe stelsels op de grove roosters te vinden zullen we de operatoren R en P definiëren. In figuur 6.2 zien we dat de vereniging van twee cellen op het fijnste rooster één cel op grove rooster G^{K-1} levert.



Figuur 6.2: Overgang van fijn naar grof rooster

De restrictie operator R wordt:

$$(Ru)_j = \frac{1}{2}(u_{2j-1} + u_{2j}), \tag{6.14}$$

De prolongatie operator P wordt:

$$(P\bar{u})_{2j} = \frac{3}{4}\bar{u}_j + \frac{1}{4}\bar{u}_{j+1}, \tag{6.15}$$

$$(P\bar{u})_{2j+1} = \frac{1}{4}\bar{u}_j + \frac{3}{4}\bar{u}_{j+1}. \tag{6.16}$$

Het nieuwe stelsel $A^{k-1}\phi^{k-1} = b^{k-1}$ wordt verkregen door:
$A^{k-1} = R^k A^k P^k$ en $b^{k-1} = R^k b^k$. Met (6.11-6.16) en de randen volgt de differentie formule voor het rooster $(k-1)$:

$$-\bar{u}_{i-1} + 2\bar{u}_i - \bar{u}_{i+1} = (Rf)_i = \bar{h}^2 \bar{f}_i \qquad i = 2, 3, \cdots, n-1 \tag{6.17}$$

In matrixvorm is A^{K-1} dan:

$$A^{K-1} = \begin{pmatrix} 4 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & -1 & 2 & -1 & \\ & & & -1 & 1 & \end{pmatrix}.$$

waarbij A^{K-1} een $(n \times n) - matrix$ is en b^{K-1} is een $(n \times 1) - vector$ is. Het stelsel A^{k-1} en het rechterlid b^{k-1} kunnen we ook door middel van discretisatie op rooster $k-1$ krijgen.

6.1 Multigrid algoritmen

Er zijn drie problemen te onderscheiden:

- Lineaire problemen,
- Niet-lineaire problemen,
- Tijdsafhankelijk lineaire of niet-lineaire problemen.

Alle multigrid algoritmen zijn varianten van het basis multigrid algoritme. Dit algoritme is voor niet-lineaire problemen, maar bevat het multigrid algoritme voor lineaire problemen als een special geval. Aangezien nu binnen de programmeertaal Fortran 90, (zie [7]) de mogelijkheid bestaat om recursieve procedures te implementeren, zullen we naast de iteratieve multigrid ook de recursieve versie bekijken, (zie [14]).
Zij $G^K : k = 0, 1, 2, \cdots, K$ een verzameling van roosters, waarbij G^K het fijnste rooster is. Zij U^k een verzameling van roosterfuncties $G^K \longrightarrow \Re$. Gegeven zijn de transfer operatoren $P^k : U^{k-1} \longrightarrow U^k$ (prolongatie) en $R^k : U^k \longrightarrow U^{k-1}$ (restrictie). We noteren het stelsel dat opgelost moet worden op rooster G^k als :

$$L^k(u^k) = b^k. \tag{6.18}$$

De operator L^k kan lineair zijn of niet-lineair zijn. Voor het gladstrijken van de fouten op ieder rooster G^k, hebben we een solver nodig, die met $S(u, v, f, \nu, k)$ aangeduid wordt. Hierin wordt met startvector u^k een verbeterde oplossing v^k verkregen. Het rechterlid is f^k, ν_k is het aantal relaxatieslagen en k is het roosterniveau. Op het grofste rooster is het nodig om de exacte oplossing te bepalen. We doen dit door het verkregen stelsel iteratief op te lossen met een hoge nauwkeurigheid.

Multigrid algoritme voor een lineair probleem

We nemen nu slechts twee roosters k en $k-1$. Voor lineaire problemen is L^k in (6.18) lineair, we nemen aan dat L^{k-1} ook lineair is. (Om L^{k-1} te bepalen kunnen we de methode van Galerkin toepassen dus $L^{k-1} = R^k L^k P^k$ bepalen of gewoon opnieuw op rooster $k-1$ discretiseren). Het eerste algoritme luidt:

Het Two-grid algoritme.

Subroutine TG($\tilde{u}$,u,f,k)
Begin

S($\tilde{u}$,u,f,ν,k)	**(1)**
$r^k = f^k - L^k(u^k)$	**(2)**
$\tilde{u}^{k-1} = 0$	**(3)**
$f^{k-1} = R^k r^k$	**(4)**
S($\tilde{u}$,u,f,$\overline{\nu}$,k-1)	**(5)**
$u^k = u^k + P^k u^{k-1}$	**(6)**
S($\tilde{u}$,u,f,μ,k)	**(7)**

End TG

In (1) wordt de solver aangeroepen met als input de startvector $\tilde{u}$, het rechterlid f, ν is het relaxatieslagen en k is het roosterniveau. De solver produceert een verbeterde oplossing u. In (2) wordt met behulp van de verbeterde oplossing u het residu r bepaald. In (3) wordt de nieuwe startvector $\tilde{u}^{k-1}$ voor $L^{k-1}(u^{k-1}) = f^{k-1}$ gekozen. Hier kunnen we $\tilde{u}^{k-1}$ ook bepalen met $\tilde{u}^{k-1} = R^k \tilde{u}^k$. In (4) wordt de solver opnieuw aangeropen, waarbij nu meer relaxatieslagen $\overline{\nu}$ nemen, zodat we heel dichter bij de exacte oplossing zitten. Deze oplossing zullen we als startvector gebruiken om het stelsel op G^k op te lossen. In (5) wordt de oplossing u^k gecorrigeerd met $P^k u^{k-1}$. In (6) wordt de solver met u^k als startvector opnieuw aangeroepen. Stappen (1) en (6) worden ook wel de pre-smoothing respectievelijk post-smoothing genoemd. Voor de recursieve versie van TG definiereen we γ_k als het aantal V-cycle van k niveau, zie figuur 6.3 voor $\gamma_k = 1$ in het geval van $k = 4$.

De recursieve versie van multigrid algoritme

Subroutine $RTG(\tilde{u}, u, f, k)$
Begin
if (k=1) then
$S(\tilde{u}, u, f, \overline{\nu}, k-1)$ **(1)**
else
$S(\tilde{u}, u, f, \nu, k)$ **(2)**
$r^k = f^k - L^k(u^k)$ **(3)**
$f^{k-1} = R^k r^k$ **(4)**
$\tilde{u}^{k-1} = 0$ **(5)**
for i=1 (step 1) until γ_k **do**
$RTG(\tilde{u}, u, f, k-1)$ **(6)**
$\tilde{u}^{k-1} = u^{k-1}$ **(7)**
endfor
$u^k = u^k + P^k u^{k-1}$ **(8)**
S($\tilde{u}$**,u,f,**μ**,k)** **(9)**
End RTG

Binnen deze procedure wordt onderscheid gemaakt tussen $k = 1$ en k groter dan 1. Als $k = 1$ (het grofste rooster) wordt u^1 bijna exact bepaalt. Voor k groter dan 1 zijn de stappen (2-5) het zelfde als de stappen (1-3) van het algoritme TG. Stap (6) wordt de procedure RTG opnieuw aangeroepen, maar nu voor rooster $k - 1$. Stappen (8-9) zijn het zelfde als (5-6) in TG.

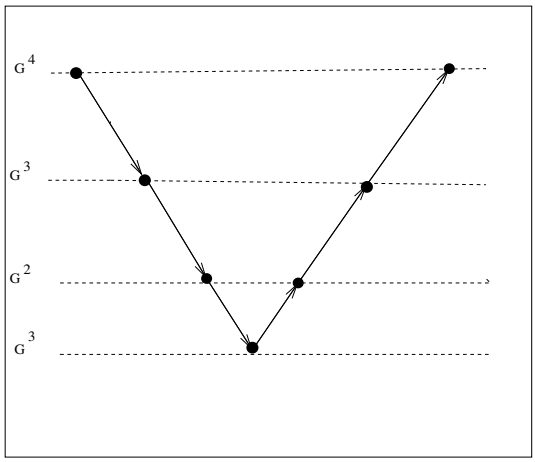
Multigrid algoritme voor een niet-lineair probleem

In het geval van een niet-lineair probleem (incompressibele Navier-Stokes vergelijkingen), hebben we te maken met niet lineaire termen, die eerst door linearisatie technieken (Picard, Newton) lineair gemaakt moeten worden. Het niet-lineaire Two grid algoritme luidt:

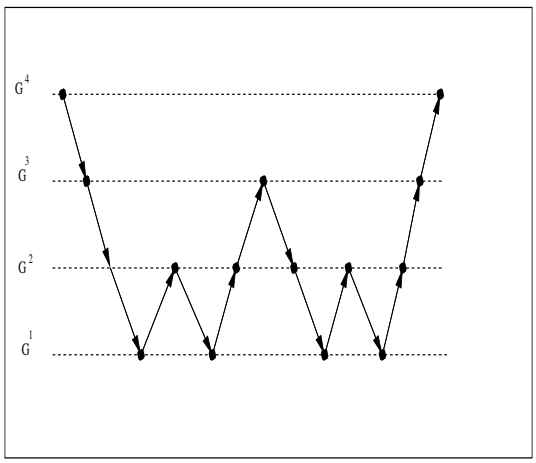
Subroutine $NMG(\tilde{u}, u, f, k)$
Begin
$S(\tilde{u}, u, f, \nu, k)$ **(1)**
$r^k = f^k - L^k(u^k)$ **(2)**
kies $\tilde{u}^{k-1}$**,** s_{k-1} **(3)**
$f^{k-1} = L^{k-1}(\tilde{u}^{k-1}) + s_{k-1} R^{k-1} r^k$ **(4)**
S ($\tilde{u}$**,u,f,**$\overline{\nu}$**,k-1)** **(5)**
$u^k = u^k + \frac{1}{s_{k-1}} P^k (u^{k-1} - \tilde{u}^{k-1})$ **(6)**
S($\tilde{u}$**,u,f,**μ**,k)** **(7)**
End NMG

voor $s_{k-1} = 1$ en $\tilde{u}^{k-1} = 0$ hebben we TG.

In stap (3) worden $\tilde{u}^{k-1}$ en s_{k-1} gekozen. Een mogelijkheid om $\tilde{u}^{k-1}$ te kiezen is $\tilde{u}^{k-1} = \tilde{R}^{k-1}u^k$. Hier hoeft $\tilde{R}^{k-1}$ niet identiek te zijn aan de restrictie operator R. Een andere mogelijkheid is op het grove rooster te beginnen. Voor het parameter $s_{k-1} = 1$ krijgen we de niet-lineaire algoritme FAS (Full Approximation Storage). De route voor het bezoek van roosters G^k wordt de multigrid cycle genoemd. In figuren 6.3-6.4 is er een aantal routes gegeven. Figuur (6.3): $\gamma_k = 1$ is de V-cycle en figuur 6.4: $\gamma_k = 2$ is de W-cycle voor k=1,2 $\cdots$, $K-1$, in het geval van $k = 4$. Als we de V-cycles en de W-cycles combineren dan krijgen we de F-cycle.



Figuur 6.3: V-cycle



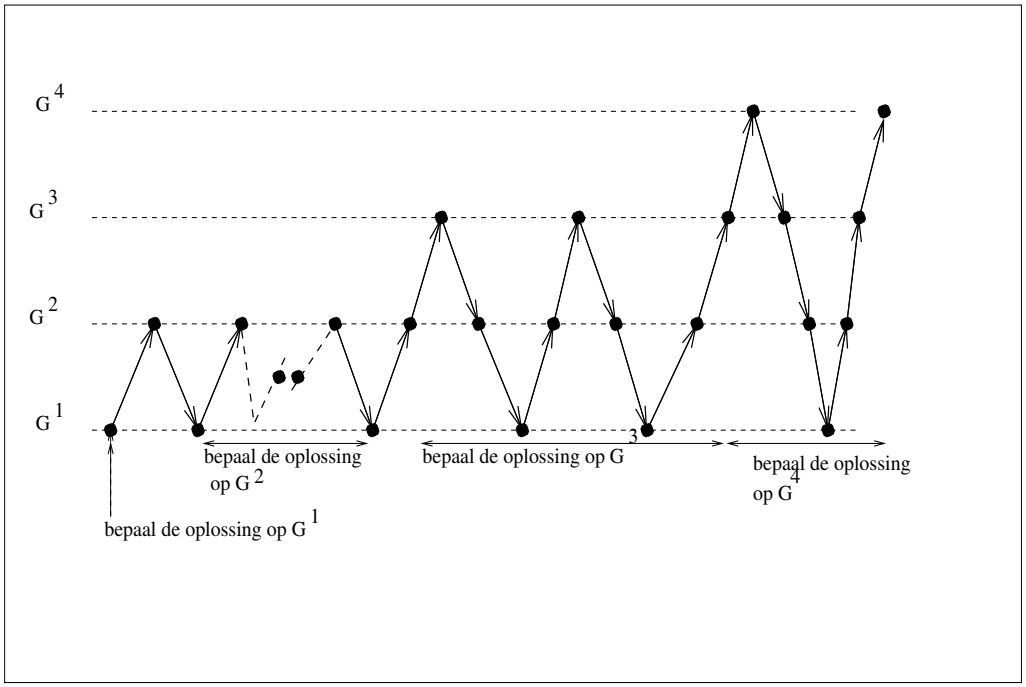
Figuur 6.4: W-cycle

Het rekenwerk met V-cycle, W-cycle en F-cycle kan gereduceerd worden als wij met een goede startvector beginnen. Dit leidt tot het full multigrid algoritme

(FMG). FMG begint om een goede startvector te krijgen op het grofste rooster. Het is zinloos om op het fijnste rooster te beginnen als wij niet over een goede startvector beschikken. In het geval van niet-lineaire problemen kan een willekeurige startvector zelfs tot divergentie leiden. Dus om deze reden zullen we op rooster $k = 1$ beginnen. Het FMG-algoritme luidt:

kies $\tilde{u}^1$ **(1)**
$S(\tilde{u}, u, f, \nu, k)$ **(2)**
for k=2 (step 1) until K do
$\tilde{u}^k = \tilde{P}^k \tilde{u}^{k-1}$ **(3)**
for i=1 (step 1) until $\tilde{\gamma}_k$ **do**
MG($\tilde{u}$**,u,f,k)** **(4)**
$\tilde{u}^k = u^k$ **(5)**
endfor
endfor

Hier kan $\tilde{\gamma}_k$ verschillend zijn dan γ_k in (4). Een mogelijk route voor FMG is in figuur 6.5 weergegeven.



Figuur 6.5: nested iteratie

6.1.1 Multigrid-SIMPLE algoritme

Het nadeel van SIMPLE is dat deze methode steeds minder efficiënt wordt als het rekenrooster steeds verfijnd wordt. De reden van de trage convergentie is ondermeer de zwakke koppeling tussen de variabelen. Om deze koppeling tussen de variabelen te versterken, wordt SIMPLE met multigrid methode versneld. Deze versnelling met multigrid methode kan op twee manieren:

- a) De binnen iteraties van SIMPLE versnellen. In dit geval wordt de SIP solver versneld.
- b) De outer iteraties van SIMPLE versnellen.

Ons aandacht gaat meer naar de versnelling van de outer iteraties van SIMPLE met niet-lineaire multigrid methode als hiervoor gegeven.
Op rooster k voldoet de exacte oplossing ϕ^k aan de volgende vergelijking:

$$L^k(\phi^k) = b^k. \tag{6.19}$$

met L^k een niet-lineaire discrete operator. Na m outer iteraties met SIMPLE op het fijnste rooster, wordt een tussenoplossing $\phi^{k,m}$ (u, v, w en p) verkregen. Als wij de tussenoplossing $\phi^{k,m}$ in (6.19) vullen dan krijgen we:

$$L^k(\phi^{k,m}) = b^k - r^{k,m}, \tag{6.20}$$

waarbij $r^{k,m}$ het residu is.
De tussenoplossing $\phi^{k,m}$ en het residu $r^{k,m}$ worden geinterpoleerd naar rooster $(k-1)$ en we geven het resultaat aan met $\tilde{\phi}^{k,m}$ en $\tilde{r}^{k,m}$. Op rooster (k-1) wordt de discrete operator gegeven door discretisatie. Nu kunnen we de vergelijking opstellen voor rooster $(k-1)$, namelijk:

$$Ł^{k-1}(\hat{\phi}^{k-1}) = L^{k-1}(\tilde{\phi}^{k-1}) + \tilde{r}^{k-1}. \tag{6.21}$$

We kunnen de gelineariseerde versie van vergelijking (6.21) schrijven als:

$$\hat{A}^{k-1}\hat{\phi}^{k-1} = \hat{b}^{k-1} + \underline{\tilde{A}^{k-1}\tilde{\phi}^{k-1} - \tilde{b}^{k-1} + \tilde{r}^{k-1}} \tag{6.22}$$

Tijdens de outer iteraties op rooster $(k-1)$ worden nu alleen de termen met een dakje veranderd. De term met een streep eronder in (6.22) verandert niet tijdens de outer iteraties van SIMPLE op rooster $(k-1)$. Alleen de eerste outer iteratie zijn de matrices $\hat{A}^{k-1}$ en $\tilde{A}^{k-1}$ gelijk en dat geldt ook voor de bijbehorende rechterleden, ofwel

$$L^{k-1}(\hat{\phi}^{k-1}) = L^{k-1}(\tilde{\phi}^{k-1}). \tag{6.23}$$

Als het residu $\tilde{r}^{k-1}$ in (6.21) nul is, dan is geen correctie nodig voor de eerder verkregen tussenoplossing $\phi^{k,m}$. In het geval dat $\tilde{r}^{k-1} \neq 0$ wordt de correctie op rooster $(k-1)$ bepaald :

$$\phi' = \hat{\phi}^{k-1} - \tilde{\phi}^{k-1}. \tag{6.24}$$

We extrapoleren de correctie ϕ' naar rooster k, vervolgens tellen we de correctie op bij de eerder verkregen tussenoplossing $\phi^{k,m}$. De gecorrigeerde oplossing wordt als nieuwe startoplossing voor SIMPLE gebruikt. Hier hebben we slechts één V-cycle beschreven.

6.1.2 Interpolatie

Doordat we voor discretisatie EVM methode met 'colocated arrangement' hebben gebruikt, kunnen we de massafluxen en de residuen voor rooster $k-1$ eenvoudig bepalen, namelijk door middel van de berekende massafluxen en de residuen op rooster k . In figuur 6.6, is één Cv op grof rooster verdeeld in vier Cv's $(2D)$ en in acht Cv's $(3D)$ verdeeld. Dit geeft de mogelijkheid om de massaflux door de zijvlakken van een grof Cv te schrijven als de som van de massafluxen door de zijvlakken van Cv's op het fijne rooster, Ofwel voor de zijvlak 'e' in $2D$ zie figuur 6.6 schrijven we :

$$\dot{m}_{e,k-1}^{I,J} = \dot{m}_{e,k}^{i+1,j} + \dot{m}_{e,k}^{i+1,j+1} \tag{6.25}$$

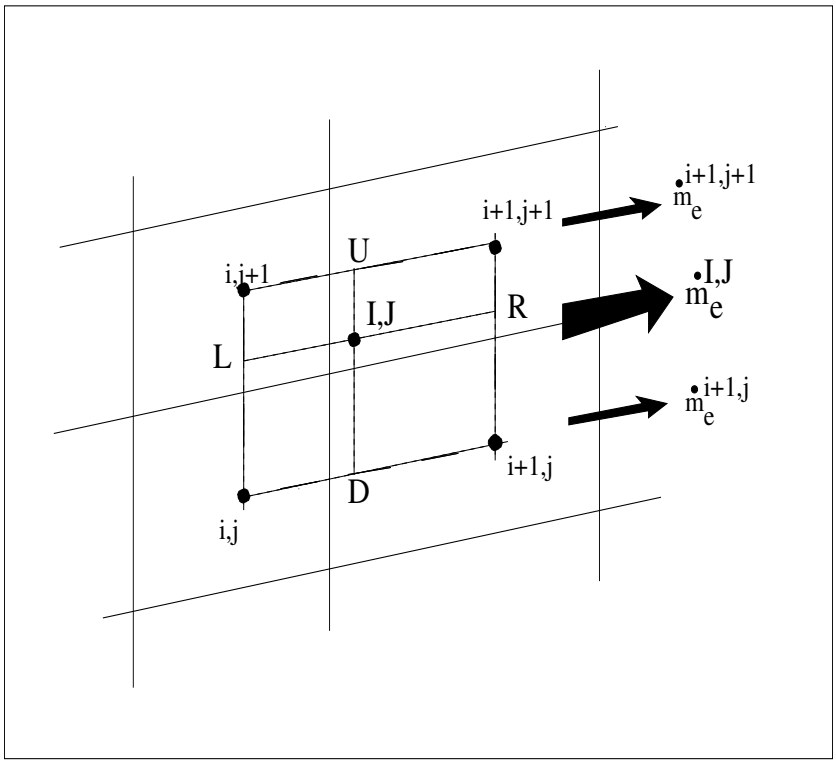
Aangezien massabehoud op het fijne rooster k geldt, blijft deze wet ook gelden op het grove rooster $k-1$.
Voor het residu r op rooster $k-1$ geldt, zie figuur 6.6:

$$\tilde{r}_{k-1}^{I,J} = r_k^{i,j} + r_k^{i,j+1} + r_k^{i+1,j} + r_k^{i+1,j+1}. \tag{6.26}$$

Voor de massaflux en het residu voor rooster $k-1$ hebben we dus geen interpolatie technieken nodig. Voor de interpolatie van variabele ϕ van rooster k naar $k-1$ gebruiken we de volgende interpolatie:

$$\phi_C = \frac{1}{N_f} \sum_{i=1}^{N_f} [\phi_{F_i} + (grad\phi)_{F_i} \cdot (r_C - r_{F_i})], \tag{6.27}$$

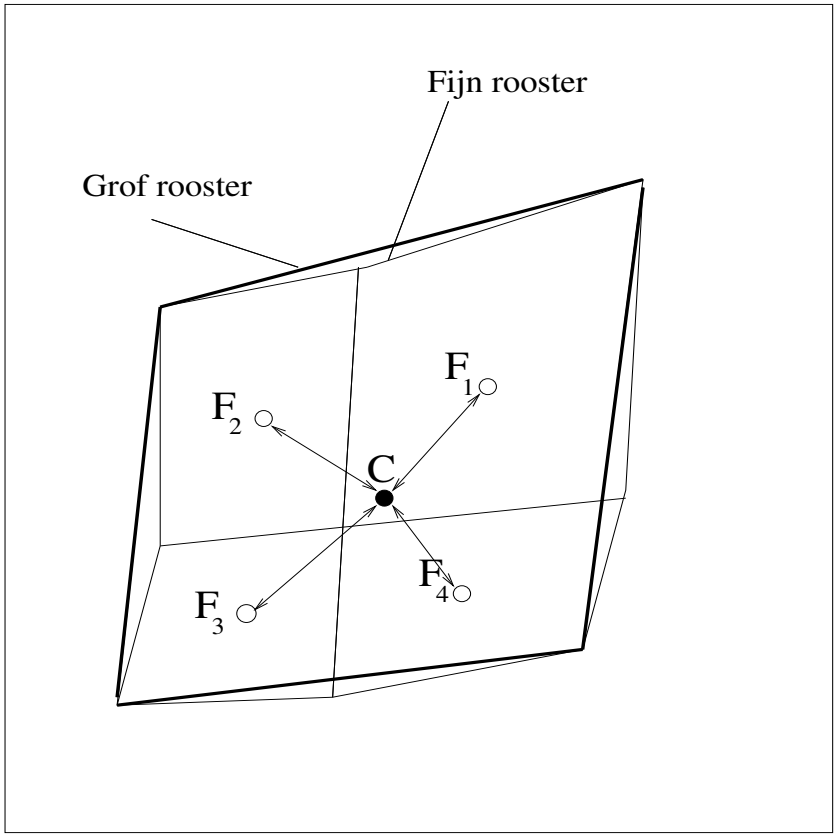
waarbij N_f het aantal fijne rooster Cv's in één grof Cv (vier in $2D$ en acht in $3D$), F_i zijn de middelpunten van 4 Cv's in één grof Cv met C als het middelpunt, r_C en r_{F_i} zijn de afstanden van het oorsprong tot het middelpunt C respectievelijk de middelpunten F_i, zie figuur 6.7 voor $2D$.



Figuur 6.6: massafluxen door zijvlak 'e'

Voor de extrapolatie van de correctie ϕ' van rooster $k-1$ naar rooster k gebruiken we de volgende vergelijking:

$$\phi'_{F_i} = \phi'_C + (grad\phi')_C \cdot (r_{F_i} - r_C) \tag{6.28}$$



Figuur 6.7: Interpolatie van de variabele ϕ van fijn naar grof rooster en andersom

Hoofdstuk 7

Multigrid en 2D Laplace vergelijking

In het vorige hoofdstuk zijn een aantal belangrijke componenten van de theorie van Multigrid methode (MGM) behandeld. Daarin zijn onder ander de volgende onderdelen behandeld :

- Algoritmen
  - Correction scheme (CS)
  - Full approximation scheme (FAS)
- Cycles
  - V-cycle
  - W-cycle
  - Full multigrid method (FMG)
- Smoothers
  - Gauss-Seidel
  - De SIP solver
  - SIMPLE

Binnen MGM procedure zijn de interpolatie technieken ook van groot belang. Deze zijn nodig om de informatie van het ene grid naar het andere te brengen. Wij zullen in dit hoofdstuk ook bilineaire interpolatie beschrijven. In dit hoofdstuk komen de volgende punten aan de orde:

- We zullen door gebruik te maken van twee iteratieve methoden : Gauss-Seidel en de SIP solver een twee dimensionale Laplace vergelijking oplossen,
- Beide iteratieve methoden zullen we als smoothers gebruiken voor MGM om dezelfde Laplace vergelijking op te lossen,

Om de efficientie van MGM t.o.v. beide iteratieve methoden te laten zien is MGM met beide iteratieve methoden aan de hand van rekentijd vergeleken. Alle oplosmethoden zijn in de programeertaal Fortran 90 geimplementeerd, zie appendix B voor de implementatiecode en de toelichting. De berekeningen zullen we uitvoeren op een werkstation HP9000 serie, model 735/99.

7.1 De exacte oplossing

We beschouwen de volgende Laplace vergelijking :

$$\frac{\partial^2\phi(x,y)}{\partial x^2} + \frac{\partial^2\phi(x,y)}{\partial y^2} = 0 \qquad (0,1)\times(0,1), \tag{7.1}$$

met de volgende Dirichlet randvoorwaarden :

$$\begin{aligned}\phi(0,y) &= 0\\ \phi(x,0) &= 0\\ \phi(x,1) &= 0\\ \phi(1,y) &= sinh(\pi)sin(\pi y)\end{aligned}$$

De Laplace vergelijking (7.1) kunnen we exact oplossen, namelijk:

$$\phi(x,y) = \sinh(\pi x)\sin(\pi y) \tag{7.2}$$

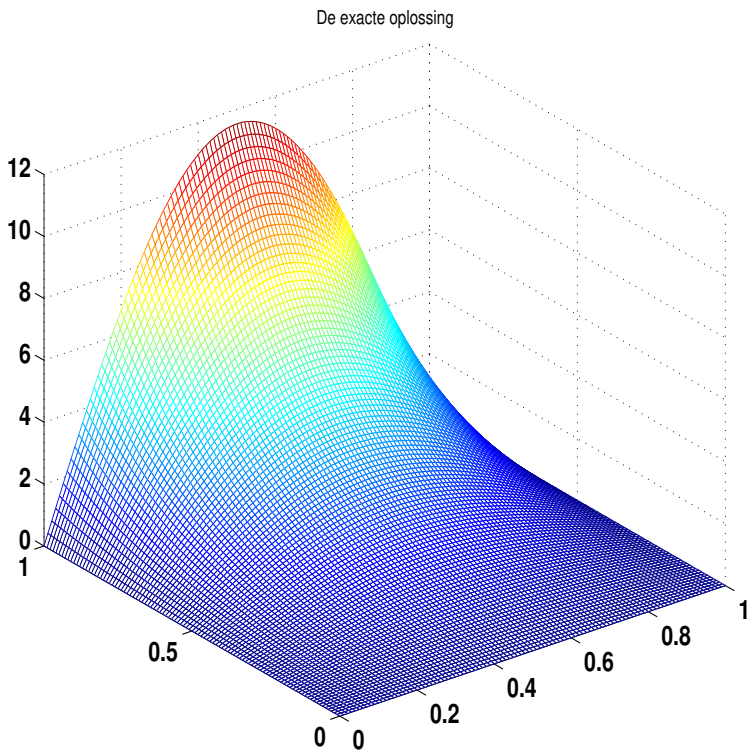
We zullen de exacte oplossing gebruiken om een vergelijking te kunnen maken met de numerieke oplossing. Met name om de discretisatiefout te bepalen en indien nodig locale gridverfijning toe te passen. Een hoge discretisatiefout treedt op in het rekendomein waarin grote gradienten van de oplossing zich voordoen. In figuur 7.1 is de exacte oplossing weergegeven.

7.2 De numerieke oplossing

De Laplace vergelijking (7.1) is met de Eindige Volume Methode (EVM) gediscretiseerd. Het rekenrooster is in figuur 7.2 weergegeven. De stapgrootte voor het discretiseren zijn h_1 in x-richting en h_2 in y-richting. We integreren over een controle volume Cv, zie figuur 7.2, vervolgens gebruiken we het divergentietheorema van Gauss en dat geeft :

$$\int_V \nabla\cdot(\nabla\phi)dV = \int_S \nabla\phi\cdot n dS, \tag{7.3}$$

waarbij S een zijvlak is van een Cv. We zullen de integraal (7.3) voor het zijvlak east 'e', zie figuur 7.2 uitschrijven. Er volgt :



Figuur 7.1: De exacte oplossing

$$\int_{y_{j-1/2}}^{y_{j+1/2}} \frac{\partial \phi}{\partial x}(x_{i+1/2}, y)dy \approx (y_{j+1/2} - y_{j-1/2})\frac{\partial \phi}{\partial x}(x_{i+1/2}, y_j)$$
$$= (y_{j+1/2} - y_{j-1/2})\frac{\phi(x_{i+1}, y_j) - \phi(x_i, y_j)}{x_{i+1} - x_i} \tag{7.4}$$

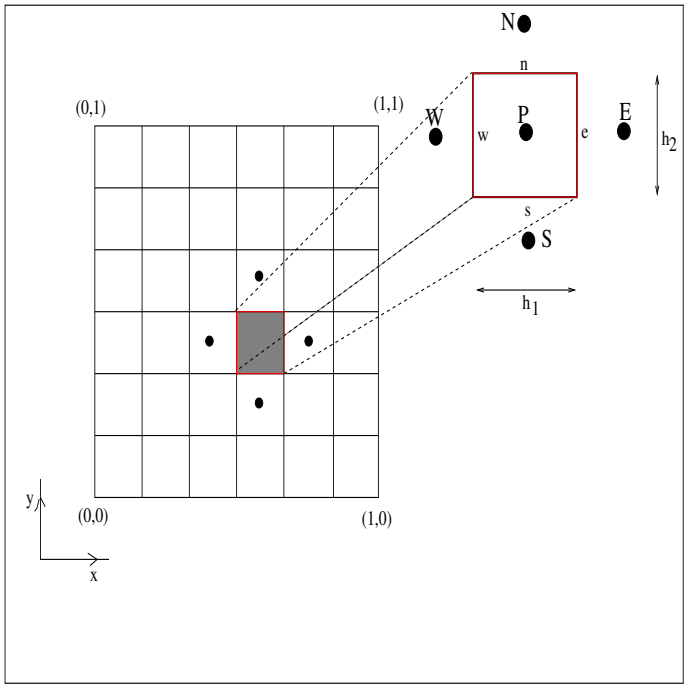
Als wij de naamgeving van hoofdstuk 3 aanhouden dan kunnen we het rechterlid van vergelijking (7.4) schrijven als:

$$\frac{y_n - y_s}{x_E - x_P}(\phi_E - \phi_P) \tag{7.5}$$

Hieruit volgt de matrix coefficient A_E:

$$A_E = \frac{y_n - y_s}{x_E - x_P} \tag{7.6}$$

Op analoge wijze kunnen we met de andere zijvlakken (west, north en south) te werk gaan. Dan volgen de matrix coefficienten :



Figuur 7.2: Het rekenrooster

$$A_W = \frac{y_n - y_s}{x_P - x_W} \tag{7.7}$$

$$A_N = \frac{x_e - x_w}{y_N - y_P} \tag{7.8}$$

$$A_S = \frac{x_e - x_w}{y_P - y_S} \tag{7.9}$$

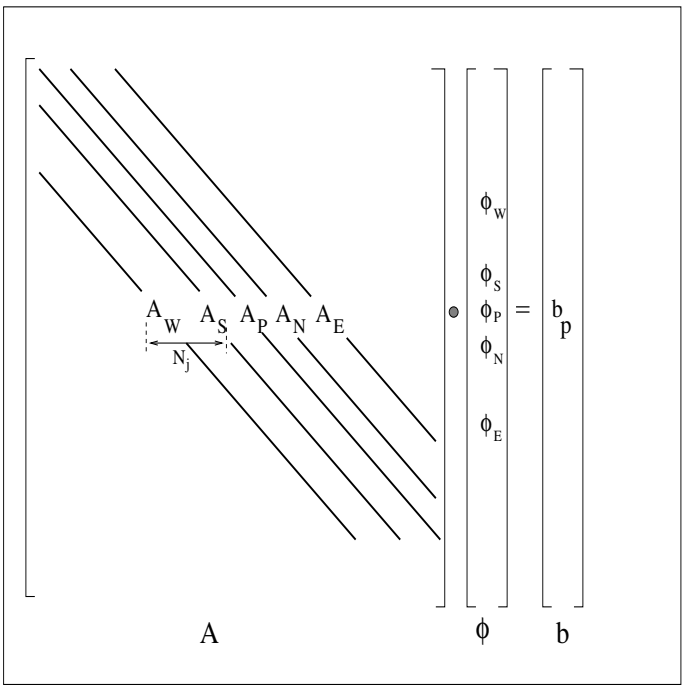
$$A_P = -\sum_{nb} A_{nb} \tag{7.10}$$

$$b_P = 0 \tag{7.11}$$

waarbij nb de buurpunten zijn van het centrum P van een Cv.
Het resultaat van het discretisatie proces leidt tot een algebraische vergelijking, die als volgt te schrijven is :

$$A_P \phi_P = \sum_{nb} A_{nb} \phi_{nb} + b_P. \tag{7.12}$$

De matrixcoefficienten zijn elk in een één-dimensionale array opgeslagen (zie tabel in hoofdstuk 3, blz.27), dat zijn zes arrays in totaal, vijf voor de matrix coefficienten en één voor het rechterlid. Het stelsel vergelijkingen (7.12) in matrix-vector vorm is in figuur 7.3 weergegeven, waarbij matrix A symmetrisch en ijl is, hetgeen inhoudt dat slechts enkele elementen per rij van A niet nul zijn.



Figuur 7.3: Het stelsel vergelijkingen

7.2.1 Implementatie van Gauss-Seidel

De structuur van matrix A, zie figuur 7.3, biedt ons de mogelijkheid om iteratieve methoden efficient te gebruiken. Gauss-Seidel is zo'n iteratieve methode. Gauss-Seidel wordt vaak, wegens het makkelijk te implementeren algoritme, gebruikt om

- Het stelsel vergelijkingen (7.12) op te lossen,
- als smoother te fungeren voor MGM algoritmen.

In het geval dat $h_1 = h_2$, kunnen we het iteratieve algoritme van GS als volgt opschrijven :

$$\phi_P^{(n+1)} = 0.25(\phi_E^{(n)} + \phi_W^{(n+1)} + \phi_N^{(n)} + \phi_S^{(n+1)} - b_P) \tag{7.13}$$

waarbij ϕ_P de oplossing in punt P en ϕ_P is de gezochte oplossing in het punt P. Verder geeft de bovenindex de iteratie aan. Hierbij doorlopen we de punten in een lexicografische manier en starten in het linker onder hoekpunt.

7.2.2 Convergentie

Om na te gaan of een iteratieproces convergeert en de gewenste oplossing van het gediscretiseerde vergelijking is verkregen, wordt gebruik gemaakt van de volgende convergentie eis:

$$\sum R_P^{(n)} \leq \delta \tag{7.14}$$

waarbij de sommatie over alle interne roosterpunten gaat, δ is van de orde van 10^{-1} tot en met 10^{-7} en R_P^n is het genormaliseerde residu en is als volgt gedefinieerd :

$$R_P^{(n)} = \frac{|\sum A_{nb}\phi_{nb}^{(n)} + b_P - A_P\phi_P^{(n)}|}{|R_P^{(0)}|} \tag{7.15}$$

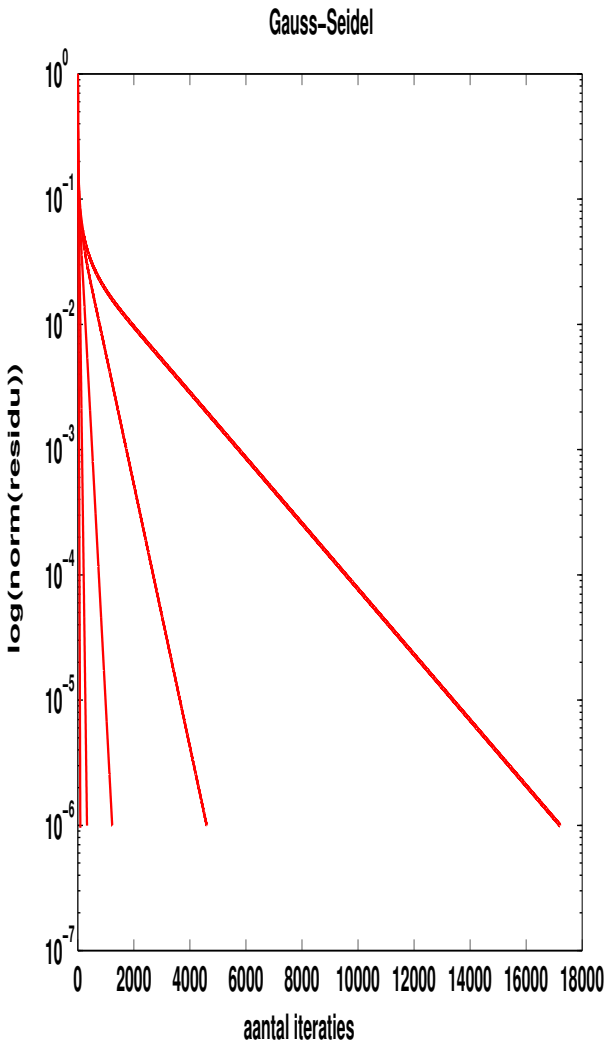
waarbij $R_P^{(0)}$ het residu is na de eerste iteratie.

7.2.3 Resultaten Gauss-Seidel

In figuur 7.4 is de norm van het residu bij verschillende roosters als functie van van het aantal iteraties weergegeven. In figuur 7.4 zijn de grafieken van links naar rechts de resultaten verkregen met GS voor de rekenroosters 8×8, 16×16, 32×32, 64×64 en 128×128. De startvector voor Gauss-Seidel is nul gekozen. In tabel 7.1 zijn de iteraties van GS bij verschillende roosters en het convergentie criterium $\delta = 10^{-6}$ weergegeven.

$rekenrooster$	$iteratie$
4×4	24
8×8	87
16×16	326
32×32	1226
64×64	4599
128×128	17213
256×256	–

Tabel 7.1: Iteraties van Gauss-Seidel



Figuur 7.4: Het verloop van het residu

7.2.4 Conclusie

Uit tabel 7.1 en de figuur 7.4, blijkt dat het GS algoritme erg langzaam convergeert naarmate het rekenrooster fijner wordt. We zien dat GS 4599 iteraties nodig heeft om een oplossing te vinden op het rekenrooster van 64 $\times 64$ controle volumes. Het verfijnen van dit rooster geeft een rekenrooster van 128 $\times$ 128 controle volumes en hiervoor heeft GS 17213 iteraties nodig. Dat is bijna vier keer zoveel t.o.v. het rekenrooster van 64 $\times$ 64.

7.3 Implementatie van de SIP solver

Een andere iteratieve methode die wij in dit hoofdstuk zullen bekijken is de SIP solver. Met deze methode kunnen we een stelsel lineaire vergelijkingen $A\phi = b$ oplossen, waarbij A zowel een symmetrische als niet symmetrische matrix mag zijn. Deze methode is erg populair binnen CFD. Met name binnen de procedure van SIMPLE hebben we zo'n solver nodig, want tijdens een outer iteratie van SIMPLE lossen we $A^{u_i}\phi = b$ op waarbij de matrices A^{u_i} voor de snelheden u, v en w niet symmetrisch zijn en vervolgens lossen we $A^d p' = b$ op voor de drukcorrectie, waarbij de matrix A^d symmetrisch is. Een andere toepssing van de SIP solver is binnen multigrid methode. De sterke smoothingseigenschappen van de eerste iteraties van de SIP solver vormen in combinatie met multigrid een efficiente oplosmethode om een stelsel vergelijkingen op te lossen.
We zullen de volgende punten bekijken:

- We zullen de SIP solver implementeren om een twee-dimensionale Laplace vergelijking op te lossen,
- Binnen SIP solver speelt de relaxatieparameter α een belangrijke rol, daarom zullen we de invloeden van α op het aantal iteraties van de SIP solver bekijken,
- We zullen MGM implementeren met de SIP solver als smoother.

We zullen het algoritme van de SIP solver toepassen om (7.12) op te lossen. Hiervoor zullen we het algoritme van de SIP solver in het kort beschrijven, zie hoofdsuk 4 voor meer details.

In de SIP solver willen we oplossen :

$$M\delta^{n+1} = r^n, \tag{7.16}$$

met

$$\delta^{n+1} = \phi^{n+1} - \phi^n, \tag{7.17}$$

De vergelijking (7.17) wordt de correctie of update genoemd. De term r^n is het residu. We willen de inverse van matrix M zo goedkoop mogelijk bepalen. We ontbinden de matrix M in L en U, waarbij de hoofddiagonaal van U enen bevat.
In de SIP solver worden de volgende elementen van L en U gebruikt :

$$L_W^l = -\frac{A_W^l}{1 + \alpha U_N^{l-N_j}}, \tag{7.18}$$

$$L_S^l = -\frac{A_S^l}{1 + \alpha U_E^{l-1}}, \tag{7.19}$$

$$L_P^l = A_P^l + \alpha(L_W^l U_N^{l-N_j} + L_S^l U_E^{l-1}) - L_W^l U_E^{l-N_j} - L_S^l U_N^{l-1}, \tag{7.20}$$

$$U_N^l = \frac{A_N^l - \alpha L_W^l U_N^{l-N_j}}{L_P^l}, \tag{7.21}$$

$$U_E^l = \frac{A_E^l - \alpha L_S^l U_E^{l-1}}{L_P^l}. \tag{7.22}$$

Nu kunnen we (7.16) op lossen.

$$U\delta^{n+1} = L^{-1} r^n = z^n, \tag{7.23}$$

waarbij z^n een hulpvector is. De elementen van z^n in (7.23) zijn te bepalen door :

$$z^l = (r^l - L_S^l z^{l-1} - L_W^l z^{l-N_j}) \frac{1}{L_P^l}. \tag{7.24}$$

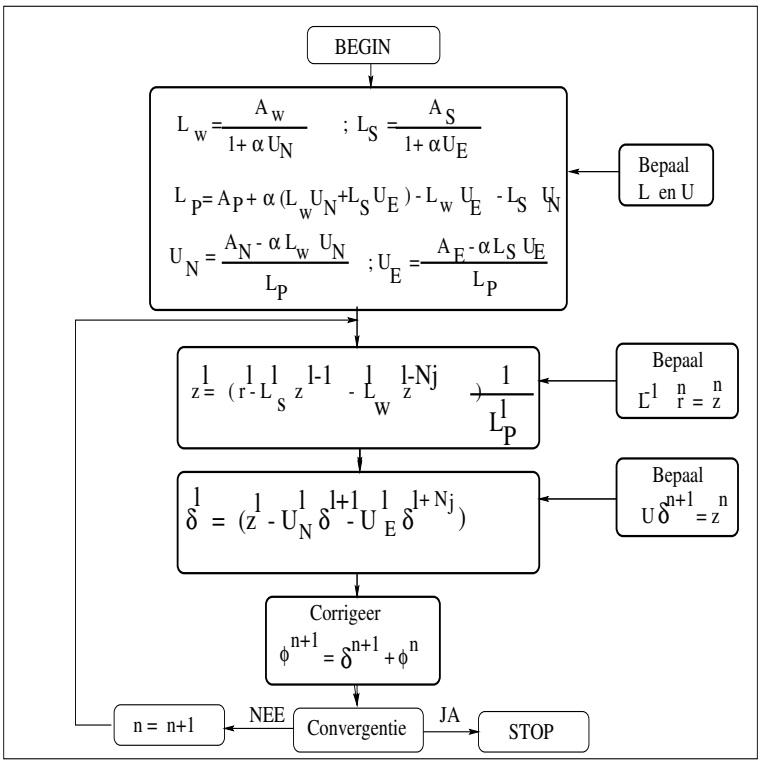
Na het bepalen van z in (7.24), kunnen we de correctie bepalen door $\delta^{n+1} = U^{-1} z^n$ op te lossen. Er volgt:

$$\delta^l = (z^l - U_N^l \delta^{l+1} - U_E^l \delta^{l+N_j}). \tag{7.25}$$

Hier is l de index van de geheugenlokatie van de elementen in een één-dimensionale array. In figuur 7.5 is het algoritme van de SIP solver aan de hand van een schema samengevat. We zullen dit schema gebruiken om de implementatie van de SIP solver te realiseren. De matrices L en U worden slechts één keer bepaald. In dit schema blijven we de variabele ϕ corrigeren tot dat aan het convergentie criterium voldaan wordt, anders wordt de volgende iteratie van de SIP solver uitgevoerd ($n = n + 1$).

7.3.1 Resultaten van de SIP solver

In figuur 7.6 is de norm van het residu bij verschillende rekenroosters als functie van het aantal iteraties weergegeven. In deze figuur zijn de grafieken van links naar rechts de resultaten voor de rekenroosters 8×8, 16×16, 32×32, 64×64, 128×128 en 256×256. In tabel 7.2 zijn de iteraties van de SIP solver bij

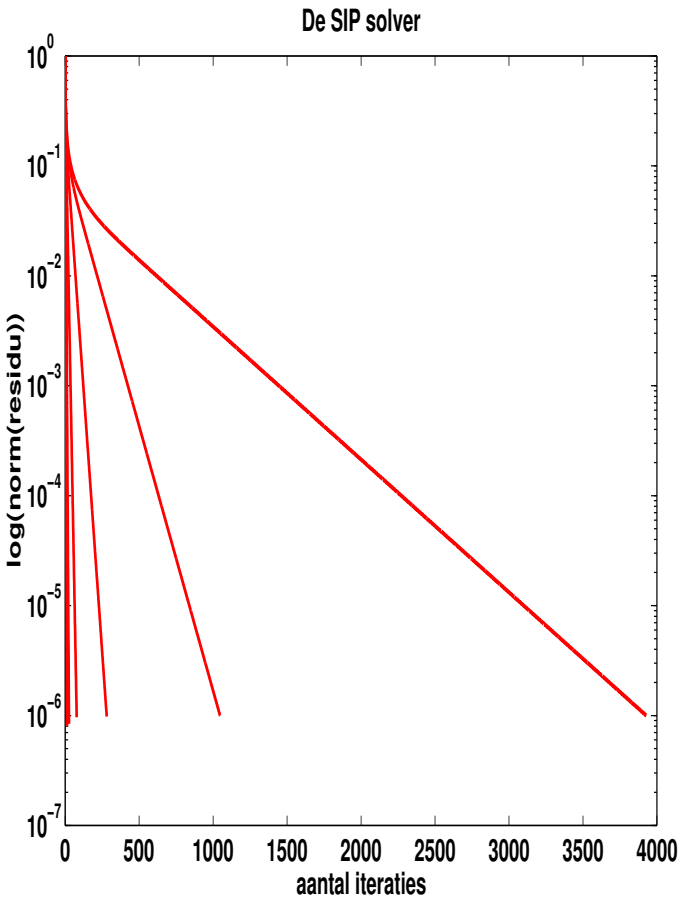


Figuur 7.5: Oplosschema voor de SIP solver

rekenrooster	*iteratie*
4×4	12
8×8	13
16×16	26
32×32	78
64×64	281
128×128	1047
256×256	3920
512×512	–

Tabel 7.2: Iteraties van de SIP solver

verschillende rekenroosters weergegeven. Deze resultaten zijn verkregen met een nul startoplossing, een stopcriterium van 10^{-6} en de relexatieparameter $\alpha = 0.92$. We hebben de relaxatie parameter α gevarieerd om na te gaan welke invloed deze parameter heeft op het aantal iteraties. Deze resultaten zijn in tabel 7.3 samengevat. Deze resultaten zijn verkregen met het rooster 64×64 en $\delta = 10^{-6}$.
We zien in tabel 7.3 dat α wel duidelijk invloed heeft op het aantal iteraties van de



Figuur 7.6: De norm van het residu bij verschillende roosters

SIP solver. Zo zien we dat de optimale waarde van α gelijk aan 0.9359 is. Bij deze waarde zijn slechts 236 iteraties van de SIP solver nodig. De SIP solver heeft de meeste iteraties nodig voor α is nul. Hiervoor heeft SIP solver 1425 iteraties nodig. De waarde α is nul geeft precies de LU ontbinding van de Incomplete Cholesky decompositie. Bij α groter 0.94 divergeert de SIP solver.
In het geval van 64 hebben we veel $\alpha's$ getest op een optimaal waarde van 0.9359 te vinden en divergentie voor α groter 0.94, maar zijn deze waarden voor alle rekenroosters constant of moeten we voor elk rooster nagaan welk α geschikt is. Om dit na te gaan hebben we de optimale α voor drie rekenrooster bepaald. De resultaten zijn tabel 7.4 weergegeven. In deze tabel is het interval van de optimale waarde van α met het aantal SIP solver iteraties weergegeven. Uit deze tabel blijkt dat de optimale α niet voor alle rekenroosters hetzelfde is, α hangt wel degelijk af van het rekenroosters. Het blijkt dat voor grove grids α zelfs één mag zijn en de SIP solver niet divergeert.

α	$iteratie$
0.	1425
0.1	1353
0.2	1274
0.3	1188
0.4	1093
0.5	985
0.6	863
0.7	721
0.8	550
0.9	333
0.9355	238
0.9356	237
0.9357	237
0.9358	237
0.9359	236
0.94	divergentie
1.0	divergentie

Tabel 7.3: Iteraties van de SIP solver

$rekenrooster$	α	$iteratie$
8×8	$0.90 \leq \alpha \leq 0.93$	13
16×16	$0.92 \leq \alpha \leq 0.921$	26
32×32	$0.93 \leq \alpha \leq 0.939$	70

Tabel 7.4: Iteraties van de SIP solver

7.3.2 Conclusie

- Uit tabel 7.2 en figuur 7.6 bijkt dat de SIP solver een redelijk goede solver is.
- Uit tabel 7.3 blijkt dat de relexatieparameter α wel duidelijk invloed heeft op het aantal iteraties van de SIP solver. Zo blijkt dat de optimale waarde van α 0.9359 is en het werken met deze waarde reduceert het aantal iteraties van de SIP solver aanzienlijk. Verder blijkt dat bij α is nul SIP solver de meeste iteraties nodig heeft. Het kiezen van α groter dan 0.94 leidt tot divergentie.
- Het blijkt uit tabel 7.4 dat de optimale α afhankelijk is van het aantal Cv's van het rekenrooster.

7.3.3 Gauss-Seidel t.o.v. SIP solver

Tot slot hebben we de resultaten verkregen met beide iteratieve methoden naast elkaar gezet. Deze resultaten zijn in de onderstaande tabel weergegeven. Uit deze tabel blijkt dat de SIP solver bij $\alpha = 0.92$ veel minder iteraties nodig heeft dan Gauss-Seidel. Zo heeft SIP solver slechts 1047 iteraties nodig om een oplossing te vinden voor het rekenrooster 128×128, terwijl GS 17213 iteraties nodig heeft en dat is beduidend meer. Het blijkt dat SIP en GS traag convergeren naarmate het rekenrooster fijner wordt en het rekentijd neemt voor beide solvers dan alleen maar toe. De oorzaak van het slechte convergentie gedrag is dat naarmate het aantal onbekenden toeneemt, neemt het conditie getal van matrix A in (7.12) ook toe, waarbij het convergentie gedrag van een iteratieve methode steeds slechter wordt. Het blijkt dat het aantal iteraties van GS evenredig is met $\frac{1}{h^2}$ en dat geldt ook voor de SIP solver. Om deze reden zullen we multigrid toepassen om snelle convergentie te realiseren.

$rekenrooster$	GS	$SIPsolver$
8×8	87	13
16×16	326	26
32×32	1226	78
64×64	4599	281
128×128	17213	1047
256×256	–	3920

7.4 Multigrid methode

Voor het implementeren van MGM hebben wij de volgende keuzes moeten maken:

- 1. Aangezien de Laplace vergelijking een lineair probleem is, zullen we het correction scheme (CS) algoritme gebruiken. In deze methode is ϕ slechts op de fijnste grids de echte oplossing, terwijl ϕ op de grove grids de correctie is. De stappen in de CS algoritme voor twee grids zijn :

  1 - Op het fijne grid $k+1$, doe een aantal iteraties met GS of SIP solver,
  2 - Bepaal het residu op het grid $k+1$,
  3 - Interpoleer het residu van grid $k+1$ naar grid k,
  4 - Doe een aantal iteraties met GS of SIP solver op de correctie vergelijking,
  5 - Interpoleer de correctie van grid k naar grid $k+1$,
  6 - Corrigeer de oplossing op het grid $k+1$,
  - Herhaal de stappen 1-6 tot dat de convergentie is gerealiseerd.

  Het Two-grid algoritme.

(1)	**Subroutine TG($\tilde{u}$,u,f,k)**
(2)	Begin
(3)	S($\tilde{u}$,u,f,ν,k)
(4)	$r^k = f^k - L^k(u^k)$
(5)	$\tilde{u}^{k-1} = 0$
(6)	$f^{k-1} = R^k r^k$
(7)	S($\tilde{u}$,u,f,$\overline{\nu}$,k-1)
(8)	$u^k = u^k + P^k u^{k-1}$
(9)	S($\tilde{u}$,u,f,μ,k)
(10)	End TG

Zie blz. 46 voor de beschrijving van dit algoritme.

- 2. Voor de cycles is de V-cycle gebruikt: We beginnen het stelsel op het fijnste grid op te lossen, vervolgens bezoeken we de grove grids tot het grofste bereikt is, dan keren we terug totdat weer het fijnste grid is bereikt, zie (blz. 48).

- 3. De correctie bepaald op grid k, die nodig is om de oplossing op grid $k+1$ te corrigeren, wordt met bilineaire interpolatie naar rooster $k+1$ gebracht.

- 4. Het residu in een grove Cv op grid k is bepaald door de som van de residuen van vier fijne roosterpunten van grid $k+1$, zie vergelijking (6.26) blz.51

- 5. De Dirichlet randvoorwaarden zijn slechts op het fijnste grid gespecificeerd. Op grover grids zijn de vier randen nul genomen, omdat we op de grovere grids de correctie vergelijking oplossen en daarom hoeven we de randen op de grove grids niet te corrigeren.

- 6. De matrices en de rechterleden worden voor elk grid k bepaald.

- 7. De matrix coefficienten $(A_W(k), A_E(k), A_N(k), A_S(k), A_P(k))$, het rechterlid $b_P(k)$ en de oplossing $\phi(k)$ zijn elke voor alle grids in een één-dimensionale array opgeslagen. Tevens zijn de roosterpunten $(X(k), Y(k))$ en de middelpunten $(XC(k), YC(k))$ van een CV voor alle grids in een één-dimensionale array opgeslagen. Deze elementen hebben we nodig om de matrix coefficienten op elk grid te bepalen. Hieronder wordt duidelijk hoe deze variabelen opgeslagen worden.

- 8. Voor het gladstrijken van de fouten zijn GS en SIP solver toegepast. De startvector op de grove grids van de V-cycle is nul en op het fijnste grid mogen we een willekeurige startvector nemen, maar hier wordt ook nul gekozen.

- 9. De convergentie check wordt slechts op het fijnste grid van elke V-cycle gedaan, immers op het fijnste grid is ϕ de gezochte oplossing voor de Laplace vergelijking.

7.4.1 Datastructuur

Binnen de MGM procedure dienen er per gridniveau veel variabelen opgeslagen te worden. Om efficient met geheugen om te gaan is het opstellen van een efficiente datastructuur noodzakelijk. Het is mogelijk om de oplossing ϕ, de coefficienten en het rechterlid van elk grid k apart in een één-dimensionale array op te slaan. Echter deze aanpak vereist veel arrays van verschillende lengtes. Om deze reden zijn de oplossingen ϕ, de matrixcoefficienten en het rechterlid voor alle grids in een één-dimensionale array opgeslagen. Hiervoor hebben we een index nodig, die de locatie van een element voor een willekeurig gridpunt en voor een willekeurig grid k in het één-dimensionale array kan weergeven. Deze opslagtechniek zullen we aan de hand van twee grids duidelijk maken. We beschouwen de twee roosters in figuur 7.7. De grove gridpunten zijn met kruisjes aangegeven en de cirkels geven de gridpunten van het fijne grid aan. In figuur 7.7 zijn de gridpunten genummerd. Deze nummers geven de geheugenlocatie van het één-dimensionale array aan. Voor een willekeurig gridpunt met coordinaat (IG, JG) in grid k, zie figuur 7.7 is deze index als volgt gedefinieerd :

$$INPG = (IG - 1)NJG + JG \tag{7.26}$$

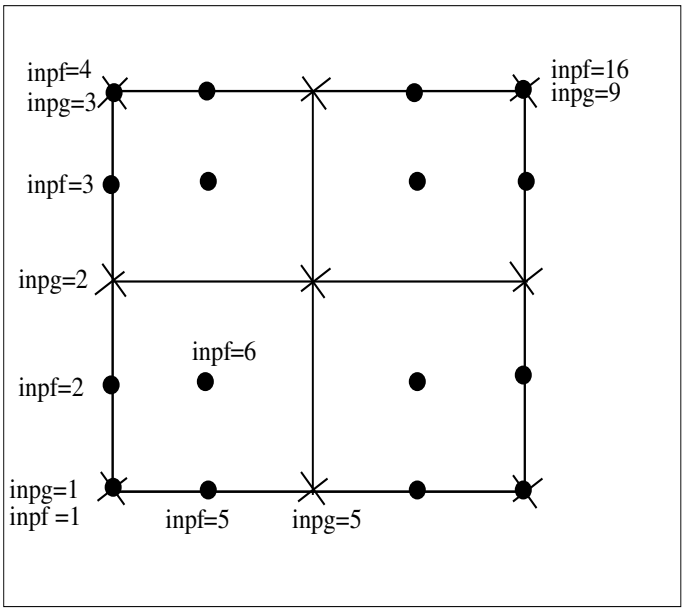
waarbij de $NJG - 1$ het aantal controle volumes is in y-richting. Voor een willekeurig gridpunt met coordinaat (IF, JF) in grid $k + 1$, zie figuur 7.7 is deze index als volgt gedefinieerd :

$$INPF = (IF - 1)NJF + JF \tag{7.27}$$

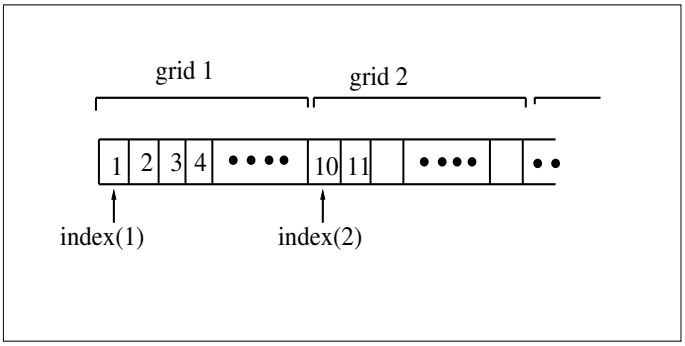
waarbij de $NJF - 1$ het aantal controle volumes is in y-richting van het fijne grid. Tussen de coordinaten (IG, JG) en (IF, JF) van beide grids bestaat de volgende relatie :

$$IF = 2(IG - 1), \tag{7.28}$$
$$JF = 2(JG - 1). \tag{7.29}$$

Het opslaan van een variabele ϕ van beide grids in een één-dimensionale array gaat als volgt:
Bepaal het aantal geheugenlocaties die nodig zijn voor de variabele ϕ op het fijne grid $k + 1$. In figuur 7.7 zien we dat dit aantal 16 is. Deze 16 elementen slaan we op in een één-dimensionale array vanaf index 10. De overige locaties (1 tot en met 9) zijn gereserveerd voor het grove grid k. In figuur 7.8 is de opslag van het data array van beide grids schematisch weergegeven. In deze figuur zijn de INDEX(1)



Figuur 7.7: Rekenroosters k en $k+1$



Figuur 7.8: Opslaan array

en INDEX(2) de startindices van de elementen voor het eerste grid, respectievelijk het tweede grid. Er geldt:

$$INDEX(1) = 0$$
$$INDEX(2) = INDEX(1) + NIGRID(2)\, NJGRID(2)$$
$$INDEX(3) = INDEX(2) + NIGRID(3)\, NJGRID(3)$$
$$INDEX(4) = INDEX(3) + NIGRID(4)\, NJGRID(4)$$

Op deze manier vinden we een algemene formule voor een willekeurige grid k ,

namelijk :

$$INDEX(k+1) = INDEX(k) + NIGRID(k+1) * NJGRID(k+1) \tag{7.30}$$

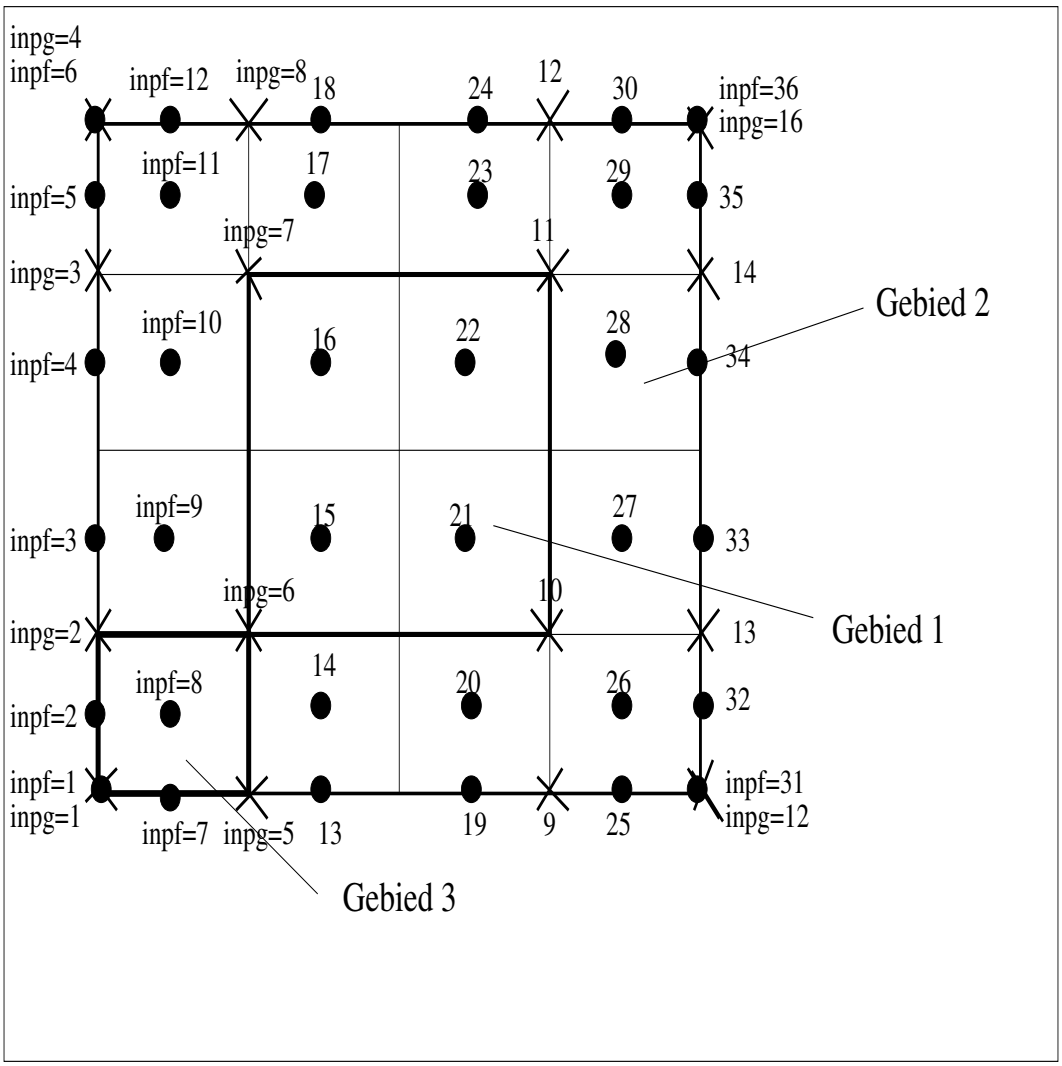
waarbij $NIGRID(k+1)$ en $NJGRID(k+1)$ het aantal roosterpunten is in x-richting respectievelijk y-richting voor grid $k+1$.

7.4.2 Bilineaire interpolatie

Voor het interpoleren van de correctie van grid k naar $k+1$ is bilineaire interpolatie gebruikt. Hieronder zullen we deze interpolatie methode beschrijven. Hiervoor beschouwen we het rekenrooster in figuur 7.9.
Het rekenrooster is in 16 fijne Cv's verdeeld. De middelpunten van de fijne Cv's zijn in figuur 7.9 met cirkeltjes aangegeven. Als wij het rekenrooster vergroven, dan krijgen we 4 grove Cv's. De middelpunten van de grove Cv's zijn in figuur 7.9 met kruisjes aangegeven. Op het grove rooster k is de oplossing ϕ bekend m.a.w. de waarden in roosterpunten aangegeven met kruisjes zijn bekend. In figuur 7.9 hebben we de index van Cv's van het fijne rooster met de index inpf en de Cv' van het grove rooster de index inpg aangegeven. Het is de bedoeling om de waarden op de fijne Cv's met interpolatie te benaderen. Dat doen we door het fijne rekenrooster in drie gebieden te splitsen:

- Gebied 1:
  De interne roosterpunten, dat zijn er in figuur 7.9 vier in totaal, aangegeven met cirkeltjes. Deze punten worden omringd door vier grove roosterpunten (kruisjes). Het gaat hier om de gridpunten met de volgende indices:
  - de fijne gridpunten: inpf= 15, 16, 21 en 22 zijn omringd door de grove gridpunten: inpg= 6, 7, 10 en 11.

- Gebied 2:
  De fijne Cv's aan de randen, zie figuur 7.9. Aan elke rand zijn er twee fijne Cv's en deze worden omringd door vier grove roosterpunten. Het gaat hier om de gridpunten met de volgende indices:
  - de fijne gridpunten aan de rand oost inpf=27 en 28 zijn omringd door de grove gridpunten inpg= 10, 11, 13 en 14,
  - de fijne gridpunten aan de rand west inpf=9 en 10 zijn omringd door de grove gridpunten inpg= 2, 3, 6 en 7 ,
  - de fijne gridpunten aan de rand noord inpf= 17 en 23 zijn omringd door de grove gridpunten inpg= 7, 8, 11 en 12,
  - de fijne gridpunten aan de rand zuid inpf= 14 en 20 zijn omringd door de grove gridpunten inpg= 5, 6, 9 en 10.



Figuur 7.9: Bilineair interpolatie

- Gebied 3:
  De vier fijne Cv's in de hoekpunten van het rekenrooster. Elk punt is omringd door vier grove roosterpunten. Het gaat hier om de gridpunten met de volgende indices:
  - het fijne gridpunt inpf= 8 is omringd door de grove punten inpg=1, 2, 5, 6,
  - het fijne gridpunt inpf= 11 is omringd door de punten inpg= 3, 4, 7,8,
  - het fijne gridpunt inpf= 26 is omringd door de punten inpg= 9, 10, 12,13,
  - het fijne gridpunt inpf= 29 is omringd door de punten inpg= 11, 12, 14, 16.

Nu zullen we deze informatie gebruiken om de waarde van ϕ op het fijne rooster bepalen:

Gebied 1:
De waarde van ϕ in dit gebied zijn als volgt te benaderen:

$$t = 0.75\phi(inpg) + 0.25\phi(inpg + 1) \tag{7.31}$$
$$u = 0.25\phi(inpg) + 0.75\phi(inpg + 1) \tag{7.32}$$

$$v = 0.75\phi(inpg + NJGRID(k)) + 0.25\phi(inpg + NJGRID(k) + 1) \tag{7.33}$$
$$w = 0.25\phi(inpg + NJGRID(k)) + 0.75\phi(inpg + NJGRID(k) + 1) \tag{7.34}$$

waarbij $NJGRID(k)$ het aantal gridpunten in y-richting van het grove grid k.

$$\phi(inpf) = 0.75t + 0.25v \tag{7.35}$$
$$\phi(inpf + 1) = 0.75u + 0.25w \tag{7.36}$$

$$\phi(inpf + NJGRID(k + 1)) = 0.25t + 0.75v \tag{7.37}$$
$$\phi(inpf + NJGRID(k + 1) + 1) = 0.25u + 0.75w \tag{7.38}$$

Gebied 2:
We beschouwen nu de twee cellen aan de oost rand. De waarde van ϕ in deze cellen zijn als volgt benaderd:

$$\phi(inpf) = 0.375\phi(inpg) + 0.125\phi(inpg + 1) + 0.375\phi(inpg + NJGRID(k)) + 0.125\phi(inpg + NJGRID(k) + 1) \tag{7.39}$$

$$\phi(inpf + 1) = 0.125\phi(inpg) + 0.375\phi(inpg + 1) + 0.125\phi(inpg + NJGRID(k)) + 0.375\phi(inpg + NJGRID(k) + 1) \tag{7.40}$$

Hier zijn de waarden van ϕ op de randen nul, omdat op grove grid zijn correctie aan de randen nul. In het geval van het roosterpunt aan de rand oost met index inpf=27 volgt het volgende:

$$\phi(inpf = 27) = 0.375\phi(inpg = 10) + 0.125\phi(inpg = 11) + 0.375\phi(inpg = 13) + 0.125\phi(inpg = 14)$$

waarbij de correcties $\phi(inpg = 13)$ en $\phi(inpg = 14)$ nul zijn. Dus $\phi(inpf = 27) = 0.375\phi(inpg = 10) + 0.125\phi(inpg = 10)$. Op analoge wijze kunnen we de waarden van de cellen aan de andere randen benaderen.

Gebied 3:

De waarde van ϕ van de cellen in de hoekpunten van het rekenrooster zijn als volgt te benaderen:

$$\phi(inpf) = 0.25(\phi(inp) + \phi(inp+1) + \phi(inp+NJGRID(k)) + \phi(inp+NJGRID(k)+1)) \quad (7.41)$$

Op analoge wijze kunnen de waarden van ϕ in de andere hoekpunten van het rekenrooster.
Hier zijn de de waarden aan de randen nul, omdat op de grove grid variabele ϕ bevat de correctie en deze is op de randen nul. In het geval van de hoekpunt met inpf=8 volgt het volgende :

$$\phi(inpf=8) = 0.25*(\phi(inpg=1)+\phi(inpg=2)+\phi(inpg=5)+\phi(inpg=6))$$

waarbij de waarde in de hoekpunt $\phi(inpg=1) = 0$ en de waarden op de randen $\phi(inpg=2)$ en $\phi(inpg=5)$ zijn nul. Dus $\phi(inpf=8) = 0.25\phi(inpg=6)$.

7.4.3 Resultaten MGM

- Smoothing met Gauss-Seidel

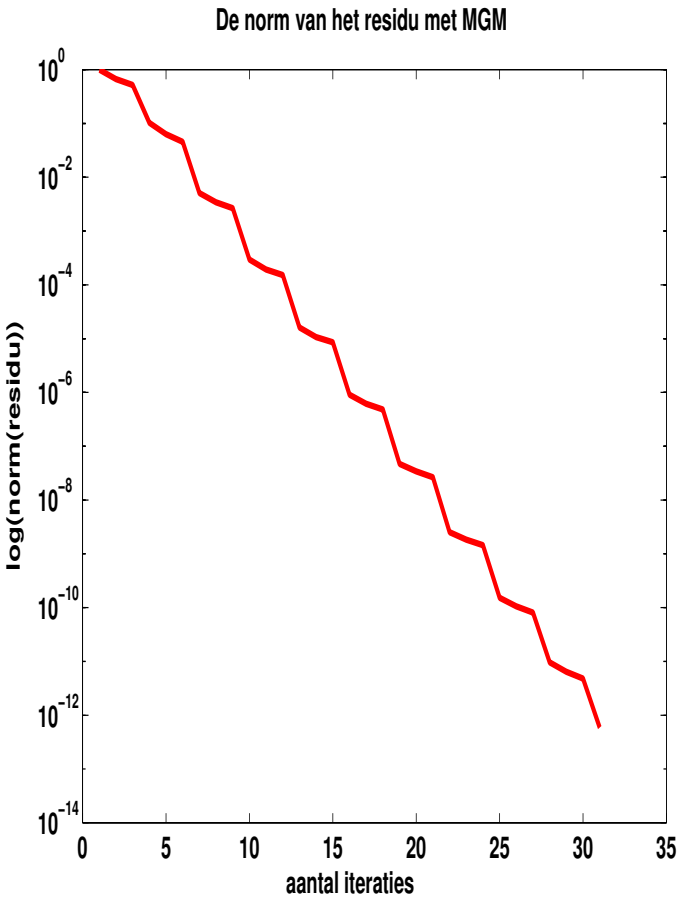
  In figuur 7.10 is het residu verloop als functie van het aantal iteraties op het fijnste grid van de V-cycle. Deze resultaten zijn verkregen door de volgende gegevens:
  - Het rekenrooster is 512×512,
  - Er zijn 9 gridniveau's gebruikt,
  - Per gridniveau zijn 3 GS-iteraties gebruikt,
  - Het convergentie criterium is $\delta = 10^{-12}$.

  Na 10 V-cycles is de norm van het residu onder $\delta = 10^{-12}$ gekomen. In figuur 7.10 zien we dat de afname van het residu naar nul erg snel gaat, dat is een groot verschil met GS en de SIP solver. In totaal zijn er 31 relaxatieslagen nodig op het fijnste grid.

  We hebben het aantal V-cyles bepaald, die nodig zijn om de oplossing van de Laplace vergelijking bij verschillende rekenroosters en bij verschillende convergentie criteria δ. Deze resultaten zijn verkregen door het maximale gridniveau per rekenrooster te nemen. Bij 8×8 en 1024×1024 zijn 3, respectievelijk 10 gridniveaus gebruikt. De resultaten zijn in tabel 7.5 weergegeven. In deze tabel zien we dat het aantal V-cycles constant blijft bij het verfijnen van het rekenrooster.

  In tabel 7.6 zijn de rekentijden die GS en MGM nodig hebben om de oplossingen behorend bij verschillende roosters te geven. Deze resultaten zijn verkregen met maximaal gridniveau en $\delta = 10^{-6}$.

  Uit deze tabel blijkt dat de rekentijd van GS bij fijne roosters heel snel toeneemt, terwijl MGM heel snel convergeert. Zo heeft MGM slechts 4.97 sec



Figuur 7.10: Het residu verloop op de fijnste grids van een V-cycle met Gauss-Seidel

	8×8	16×16	32×32	64×64	128×128	256×256	512×512
10^{-6}	6	6	5	5	5	5	5
10^{-5}	5	5	5	4	4	4	4
10^{-4}	4	4	4	4	4	4	4
10^{-3}	3	3	3	3	3	3	3
10^{-2}	2	2	2	2	2	2	2
10^{-1}	1	1	1	1	1	1	1

Tabel 7.5: Aantal V-cycle van MGM

nodig om een oplossing voor rooster 128×128 te bepalen. We zien dat MGM ook snel convergeert voor een erg fijn grid. Zo zijn er slechts 3 minuten en 25 seconden nodig om een oplossing te vinden voor het rekenrooster

$rekenrooster$	GS	MGM
8×8	0.06 sec	0.05 sec
16×16	0.46 sec	0.10 sec
32×32	5.98 sec	0.32 sec
64×64	88.78 sec	1.21 sec
128×128	1563.83 sec	4.97 sec
256×256	–	20.13 sec
512×512	–	81.83 sec
1024×1024	–	3 min 25 sec

Tabel 7.6: Rekentijden van Gauss-Seidel en Multigrid

van 1024×1024. Hieruit kunnen we concluderen dat MGM niet alleen sneller convergeert dan GS, maar dat MGM ook weinig tijd nodig heeft om een oplossing te vinden op erg verfijnde grids.

- Smoothing met SIP solver

  Voor het implementeren van MGM met SIP solver als smoother, hebben we ook de keuzes 1-9 gemaakt. We hebben het aantal V-cycles bepaald, die nodig zijn om de Laplace vergelijking bij verschillende rekenroosters en bij verschillende convergentie criteria. Deze resultaten zijn verkregen door maximale gridniveaus per rooster te nemen. Bij 8×8 en 1024×1024 zijn 3, respectievelijk 10 gridniveaus gebruikt. De resultaten zijn in tabel 7.7 weergegeven. Deze resultaten zijn met de volgende gegevens verkregen:

  - Per gridniveau zijn 3 SIP solver iteraties gebruikt,
  - De relaxatieparameter α is 0.92,
  - De startoplossing op het fijnste grid is nul gekozen.

	8×8	16×16	32×32	64×64	128×128	256×256	512×512
10^{-6}	3	5	5	4	3	3	3
10^{-5}	3	4	4	3	3	2	2
10^{-4}	2	3	3	2	2	2	2
10^{-3}	2	2	2	2	2	2	2
10^{-2}	1	1	1	1	1	1	1
10^{-1}	1	1	1	1	1	1	1

Tabel 7.7: Aantal V-cycle van MGM met $\alpha = 0.92$

In deze tabel zien we dat het aantal V-cycles constant blijft bij het verfijnen van het rekenrooster.

We hebben ook Incomplete Cholesky decompositie als smoother gebruikt voor multigrid. Incomplete Cholesky decompositie is equivalent met de SIP solver α=0. Ook hier hebben we het aantal V-cycles bij verschillende roosters en verschillende convergentie criteria bepaald. Deze resultaten zijn in tabel 7.8 weergegeven. Uit deze tabel blijkt dat er geen groot verschil is in het aantal V-cycles met SIP solver met $\alpha = 0.92$.

	8×8	16×16	32×32	64×64	128×128	256×256	512×512
10^{-6}	4	4	4	4	4	4	4
10^{-5}	3	3	3	3	3	3	3
10^{-4}	3	3	3	3	2	2	2
10^{-3}	2	2	2	2	2	2	2
10^{-2}	1	1	1	1	1	1	1
10^{-1}	1	1	1	1	1	1	1

Tabel 7.8: Aantal V-cycle van MGM met $\alpha = 0$.

In tabel 7.9 zijn de rekentijden die de SIP solver en MGM nodig hebben om de oplossingen behorend bij verschillende roosters te geven. Deze resultaten zijn voor MGM met maximale gridniveaus verkregen. De startvector is voor beide oplosmethoden nul gekozen, δ is 10^{-6} en $\alpha = 0.92$.

rekenrooster	*SIP solver*	*MGM*
8×8	0.04 sec	0.05 sec
16×16	0.10 sec	0.12 sec
32×32	0.66 sec	0.40 sec
64×64	8.77 sec	1.44 sec
128×128	131.83 sec	5.32 sec
256×256	1949.28 sec	20.94 sec
512×512	–	87.99 sec
1024×1024	–	3 min 19 sec

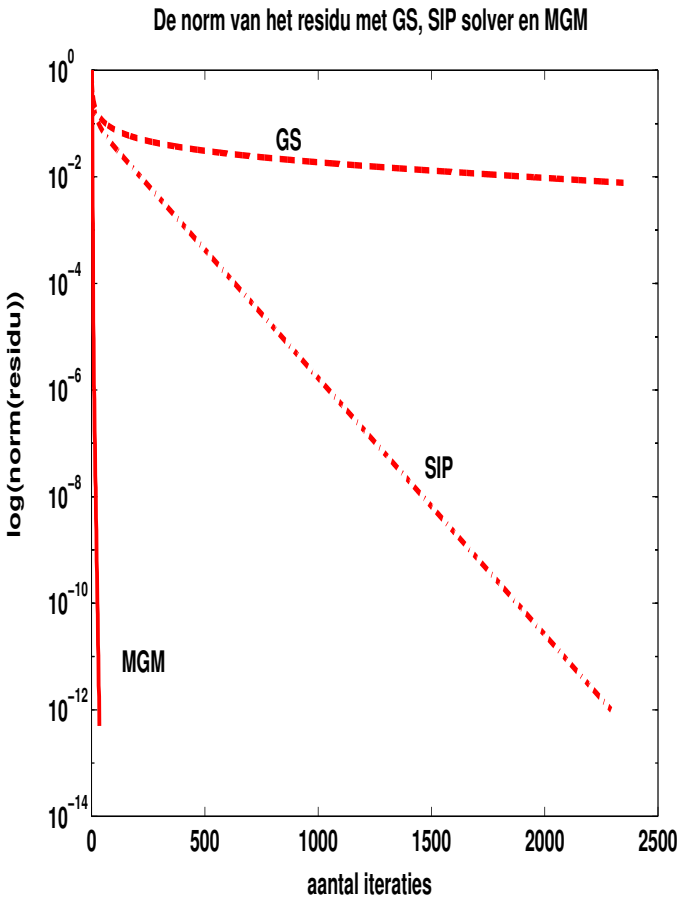
Tabel 7.9: Rekentijden van de SIP solver en Multigrid

Uit deze tabel blijkt dat de rekentijd van de SIP solver bij fijne rekenrooster snel toeneemt. De rekentijden van MGM zijn heel goed. Zo heeft MGM slechts 3 min en 19 sec nodig om de oplossing te vinden voor de rekenrooster van 1024×1024, dat zijn ongeveer één miljoen Cv's. In tabel 7.10 zijn het aantal iteraties van beide smoothers op fijnste grid weergegeven. Uit deze tabel blijkt duidelijk dat de SIP solver veel beter smoother dan GS naarmate het grid verfijnd wordt.
Tot slot hebben we de norm van het residu als functie van het aantal iteraties verkregen met Gauss-Seidel, de SIP solver en Multigrid bekijken. De resulaten zijn

$rekenrooster$	$MGM\ met\ GS$	$MGM\ met\ SIP$
8×8	19	12
16×16	19	16
32×32	17	17
64×64	17	16
128×128	16	13
256×256	16	10
512×512	16	10

Tabel 7.10: Aantal iteraties op het fijnste grid

in figuur 7.11 weergegeven. Deze resultaten zijn verkregen met een grid van 128^2 controle volumes en het convergentie criterium $\delta = 10^{-12}$. De iteraties van Multigrid zijn de iteraties op het fijnste grid van een V-cycles.



Figuur 7.11: Het residu verloop van GS, SIP solver en MGM

In deze figuur zien we dat Multigrid na enkele iteraties naar δ=10^{-12} convergeert, terwijl de andere solvers heel veel moeite hebben om te convergeren.
Beter convergentie gedrag dan die van Multigrid kunnen we niet wensen.

Hoofdstuk 8

De X-stream code

CFD pakketen met EEM (Eindige Elementen Methode) en EVM (Eindige Volume Methode) verschillen vooral in door het type rekenrooster dat gebruikt werd. De elementen bij EEM lieten toe dat elke willekeurige geometrie gediscretiseerd kon worden, terwijl dat bij de (multiblok) gestructureerde rekenroosters van de oudere EVM programma's vaak onmogelijk was. Echter vandaag de dag zijn er EVM pakketten die dezelfde cel vormvrijheid hebben als EEM. Deze CFD pakketten met EVM zijn weer te onderscheiden door :

- Het type rekenrooster dat gebruikt wordt; cartesisch, (multi) block structured of unstructured, niet-orthogonaal enzovoort.
- De cel-vormen : hexaheders, tetraeders, prisma's enzovoort.
- Waar de stromingsvariabelen berekend worden: " staggered" versus "colocated".
- De tijdsintegratie voor tijdsafhankelijk problemen: expliciet, impliciet of een tussenvorm.
- Hoe de koppeling tussen druk en de snelheid tot stand komt : bekende algoritmes zijn SIMPLE, SIMPLER, SIMPLEC, PISO enzovoort.

X-stream is een EVM CFD pakket en is gebaseerd op de volgende onderdelen:

- Het type rekenrooster dat gebruikt wordt: (multi) block structured, niet-orthogonaal
- De cel-vormen : hexaheders,
- waar de stromingsvariabelen berekend worden: "colocated".
- Hoe de koppeling tussen druk en de snelheid tot stand komt : SIMPLE,

Het model dat voor simulatie van laminaire en turbulente stromingen gebruikt wordt, bestaat uit een stelsel niet-lineaire gekoppelde differentiaal vergelijkingen (Navier-Stokes vergelijkingen). De meest populaire sequentiel algoritme om deze vergelijkingen op te lossen is zondermeer het SIMPLE algoritme. Om het convergentiegedrag van SIMPLE te verbeteren zijn een aantal varianten van SIMPLE ontwikkeld . Deze varianten zijn SIMPLER, SIMPLEC enzovoort.

X-stream is verbonden met pre- en postprocessing. Tussen de pre- en de postprocessing zit het hoofdprogramma die gebruik maakt van de SIMPLE solver.

Preprocessing

De preprocessing bestaat uit twee inputfiles. De eerste inputfile bevat de informatie over het stromingsprobleem. Deze informatie varieert van de keuze tussen het simuleren van een laminaire dan wel van een turbulente stroming tot het kiezen van het discretisatie schema voor de convectieve term. Hieronder zullen we een beschrijving van de inputparameters geven.

- TITLE : Het specificeren van de naam van het testcase.
- FILGRD : Het specificeren van de naam van het gridfile. Deze file bevat de gegevens over het grofste grid.
- LREAD : Dat is een logische variabele. TRUE geeft aan dat de berekeningen hervat kunnen worden met de opgeslagen resultaten als startwaarden. In X-stream is het mogelijk om de berekeningen te stoppen om bijvoorbeeld de relexatieparameters bij te stellen.
- LONGEO : TRUE geeft aan dat slechts de verfijning van het rekenrooster aangemaakt wordt. Deze grid data wordt in een file opgeslagen om gevisualiseerd te worden. Er worden geen berekeningen uitgevoerd.
- LTIME : TRUE geeft aan dat de gebruiker een tijdsafhankelijk probleem wil simuleren.
- ALFA : De relaxatie parameter voor de SIP solver.
- SLARGE : In het geval dat het residu groter is dan SLARGE dan spreken we van divergentie van SIMPLE en de berekeningen worden gestopt.
- DT : Tijdstap in het geval van tijdsintegratie voor een tijdsafhankelijk probleem.
- GAMT : Blendingfactor voor tijdsdiscretisatie. GAMT=0 : impliciet eerste orde Euler, GAMT=1 : impliciet tweede orde Euler, GAMT tussen 0 en 1 : blending tussen beide schema's.

- ITURB : TRUE geeft aan dat de gebruiker een turbulente stroming wil simuleren.
- CMU, C1, C2, CAPPA, LOG : Constanten die in het standaard $k - \epsilon$ model gebruikt worden.
- UIN, VIN, WIN : Waarden voor het snelheidsveld om de berekeningen van SIMPLE te starten.
- DENSIT, VISCOS, PRANDTL : Dichtheid van het fluidum, moleculair dynamische viscositeit, moleculair Prandtl getal.
- UINLET, VINLET, WINLET : Constante waarden van de snelheden aan de inlaat rand. In X-stream is het nu ook mogelijk een varierende inlaat snelheid te specificeren.
- KRES : Het gridlevel waar de berekeningen hervat worden, met name wanneer LREAD TRUE is.
- SORMAX Convergentie critirium voor SIMPLE, zie paragraaf 8.2
- LSG : Maximaal aantal outer iteraties van SIMPLE.
- NSW : Maximaal aantal inner iteraties van de SIP solver.
- URF : Onderrelaxatie factor voor de snelheden en de druk.
- SOR convergentie criterium voor inner iteraties van de SIP solver.
- GDS : Blendingfactor in het deferred discretisatie schema. GDS=0 : UDS, GDS=1 : CDS, en GDS tussen 0 en 1 geeft een mix tussen beide schema's.

In appendix C is een voorbeeld van een inputfile in X-stream weergegeven. Een andere inputfile die de gebruiker moet specificeren is de gridfile. X-stream vraagt in de inputfile om een gridfile. X-stream is nog niet instaat om de griddata van het grofste grid te creeren, maar beschikt wel over een procedure om een grof grid een aantal keer te verfijnen. Dit aantal wordt door de gebruiker gespecificeerd. Dus andere gridgeneratoren moeten gebruikt worden om de griddata van het grofste grid te verschaffen. In paragraaf 8.3 is het genereren van zo'n grid en het verfijnen in X-stream kort beschreven.
Deze gridfile bevat onder andere :
- Aantal blokken.
- Het aantal controle volumes in x-, y- en z-richting per blok.
- Aantal gridverfijning.
- Het type randvoorwaarde.

Hieronder geven we een onderdeel van de gridfile.
5 Number of blocks
1 1 2 4 6 Block number,Refinement level,NICV,NJCV,NKCV (at level 1 !)

2 1 2 2 6 Block number,Refinement level,NICV,NJCV,NKCV (at level 1 !)
3 1 2 2 2 Block number,Refinement level,NICV,NJCV,NKCV (at level 1 !)
4 2 2 2 2 Block number,Refinement level,NICV,NJCV,NKCV (at level 1 !)
5 3 4 8 6 Block number,Refinement level,NICV,NJCV,NKCV (at level 1 !)
Hier is het rekendomein in 5 blokken verdeeld, waarbij het aantal Cv's voor elke blok is gespecificeerd. Het aantal keer dat een blok verfijnd wordt is ook gegeven.

Solver

Nadat de gebruiker beide inputfiles klaar heeft, dan kan X-stream met de berekeningen starten. Het rekenen gebeurt in het hoofdprogramma, die uit het SIMPLE algoritme bestaat.
In hoofdstuk 4 hebben we het SIMPLE algoritme uitgebreid beschreven, daarom zullen we hier deze methode slechts samenvatten. De gediscretiseerde impulsvergelijking is als volgt te schrijven:

$$A_P^{u_i} U_{i,P}^* = \tilde{b}_{i,P} - \sum A_l^{u_i} U_{i,l}^* - \Delta V_P (\frac{\partial p^*}{\partial x_i})_P. \tag{8.1}$$

De snelheden bepaald met (8.1) voldoen niet aan de continuiteitsvergelijking. Om aan de continuiteitsvergelijking te kunnen voldoen, worden deze snelheden gecorrigeerd met snelheidcorrectie U_i'. De druk p^* wordt ook gecorrigeerd met drukcorrectie p', ofwel

$$U_i = U_i^* + U_i', \tag{8.2}$$
$$p = p^* + p'. \tag{8.3}$$

Door (8.2-8.3) te substitueren in (8.1) volgt een relatie tussen de drukcorrectie en de snelheidscorrectie, namelijk :

$$U_{i,P}' = -\frac{\Delta V_P}{A_P^{u_i}} (\frac{\partial p'}{\partial x_i})_P. \tag{8.4}$$

Om de massafluxen in de continuiteisvergelijking te bepalen, hebben we de snelheid op de zijvlakken nodig. Rie en Chow hebben om de druk oscillaties in colocated grid arrangement te voorkomen de volgende snelheid voorgesteld, voor het zijvlak east bijvoorbeeld:

$$U_{i,e}^* = \overline{(U_i^*)}_e - \Delta V_e \overline{(\frac{1}{A_P^{u_i}})}_e ((\frac{\partial p^*}{\partial x_i})_e - \overline{(\frac{\partial p^*}{\partial x_i})_e}) \tag{8.5}$$

De afleiding van de formule (8.5) is in paragraaf 8.5 beschreven. De massaflux berekend met (8.5) voldoet niet aan de continuiteitsvergelijking en wordt gecorrigeerd zodat :

$$\sum_c (\dot{m}_c' + \dot{m}_c) = 0, \quad c = e, w, n, s, t, b \tag{8.6}$$

waarbij $\dot{m}'$ de massaflux correctie is, die berekend is met de snelheidcorrectie (8.4). Deze correctie wordt voor het zijvlak east als volgt:

$$U'_{i,e} = -\frac{\overline{\Delta V_P}}{A_P^{u_i}}(\frac{\partial p'}{\partial x_i})_e \tag{8.7}$$

Invullen van de snelheidcorrecties in (8.6) geeft de volgende vergelijking, ook wel de drukcorrectie vergelijking genoemd:

$$A_P p'_P = \sum_{nb} A_{nb} p'_{nb} + \sum_c \dot{m}_c \tag{8.8}$$

Na het oplossen van deze vergelijking gebruiken we de oplossing p' om de snelheden en de druk te corrigeren door gebruik te maken van (8.2-8.3). Het beschreven algoritme vatten we als volgt samen:

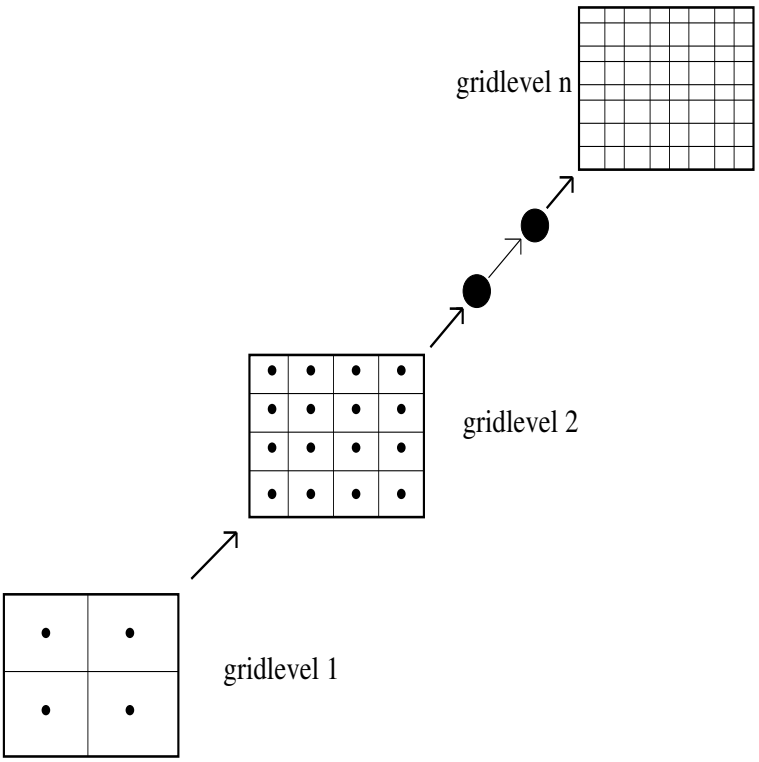
- 1. Kies start waarden voor de snelheden u, v, w en de druk p;
- 2. Los op de impulsvergelijkingen voor u^*, v^*, w^* en p^*;
- 3. Los op de drukcorrectie vergelijking voor p';
- 4. Update het snelheidsveld en het drukveld gebaseerd op p';
- Herhaal stappen 2, 3 en 4 totdat de residu onder een gegeven waarde komt.

SIMPLE is een iteratieve procedure en men heeft om een iteratie proces te starten een startoplossing nodig. Het kiezen van een goede startoplossing is noodzakelijk, want in dit geval, dus niet lineaire vergelijkingen kan een willekeurige startoplossing tot divergentie leiden. In X-stream wordt de startoplossing als volgt verkregen:

- Door het gebrek aan een goede startoplossing om SIMPLE op het fijnste grid te starten, wordt eerst een oplossing met SIMPLE op het grofste grid $k = 1$ gekregen, waarbij SIMPLE vanwege weinig gridpunten redelijk snel convergeert. De verkregen oplossing wordt naar grid $k + 1$ geinterpoleerd om daar als een startoplossing te dienen voor grid $k+1$. Na convergentie van SIMPLE op dit grid wordt de oplossing weer naar grid $k + 2$ geinterpoleerd. Dit proces wordt herhaald totdat het fijnste grid is bereikt. In figuur 8.1 is deze oplossingsprocedure schematisch weergegeven.

Postprocessing

Na de convergentie van SIMPLE worden de resultaten naar een outputfile geschreven. Deze resulaten kunnen gevisualiseerd worden.



Figuur 8.1: Startoplossing voor SIMPLE

8.1 Onderrelaxatie

Vaak wil het iteratieproces niet monotoon convergeren doordat oscillaties in de oplossing optreden. Deze oscillaties vertragen de convergentie en kunnen zelfs tot divergentie van het iteratieproces leiden. Om deze reden introduceren we een onderrelaxatie van de oplossing:

$$\tilde{\phi}_P^m = \alpha\phi_P^m + (1-\alpha)\phi_P^{m-1}, \tag{8.9}$$

waarbij m de outeriteratie teller van SIMPLE is en α is de relaxatiefactor. De waarde is α is tussen nul en een. Als wij de vergelijking (8.9) in de volgende vergelijking substitueren,

$$A_P\phi_P = \sum_{nb} A_{nb}\phi_{nb} + b_P, \tag{8.10}$$

dan volgt een stelsel vergelijkingen zoals (8.10), maar met A_P wordt vervangen door A_P/α en b_P vervangen door $b_P + (1/\alpha - 1)A_P\phi_P^{m-1}$. Deze veranderingen leidt tot het verhogen van dominatie van het hoofddiagonaal van de coefficient matrix, terwijl de andere diagonalen ongewijzigd zijn gebleven en dus hiermee krijgen we een meer robuste solver.

8.2 Convergentie cretiria

Om na te gaan of een iteratie proces geconvergeerd is naar het gewenste resultaat van van de gediscretiseerde vergelijking wordt in X-stream het volgende convergentiecriterium toegepast:

$$\sum R_P^{(n)} \leq \delta, \tag{8.11}$$

waarbij de som over alle interne grid cellen gaat. δ is van de orde 10^{-5} en $R_P^{(n)}$ is het genormaliseerde absolute residu en is als volgt gedefinieerd:

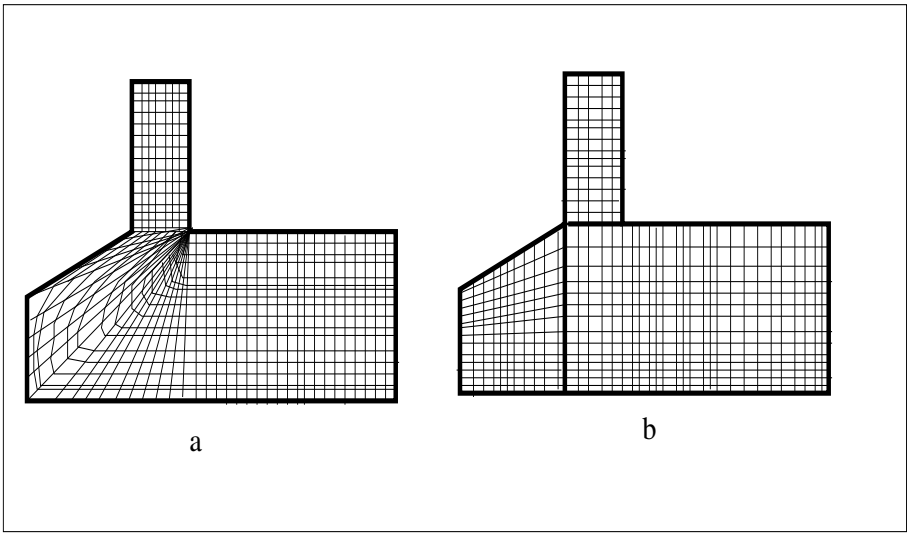
$$R_P^{(n)} = \frac{|\sum_{nb} A_{nb}\phi_{nb} + b_P - A_P\phi_P|}{|\dot{M}\phi_{ref}|}, \tag{8.12}$$

waarbij $\dot{M}$ de totale inlaat massflux is en ϕ_{ref} is een referentie punt in het rekendomein.

8.3 Het genereren van niet-orthogonale grids

Tegenwoordig is het simuleren van stromingsproblemen in een gecompliceerde geometrie eerder regel dan uitzondering. De discretisatie van deze geometriëen in een gestructureerd, niet-orthogonaal grid van hoge kwaliteit was en blijft nog steeds een grote uitdaging binnen CFD. In X-stream wordt het rekendomein in een aantal blokken verdeeld, waarbij elke blok uit een gestructueerd niet-orthogonaal grid bestaat. De voordelen van het genereren van multiblok grids zijn onder andere:

- Vaak is het onmogelijk om het rekendomein met één gestructureerd niet-orthogonaal grid te discretiseren, zie figuur 8.2a. Om deze probleem te ondervangen, wordt het rekendomein in een aantal blokken verdeeld, waarbij elke blok in gestructureerd, niet-orthogonaal grid (block-structured grid) gediscretiseerd is, zie figuur 8.2b. In deze figuur is duidelijk te zien we dat de kwaliteit van het grid veel beter is geworden, de controle volumes zijn bijna orthogonaal, dus geen scheve Cv's meer.

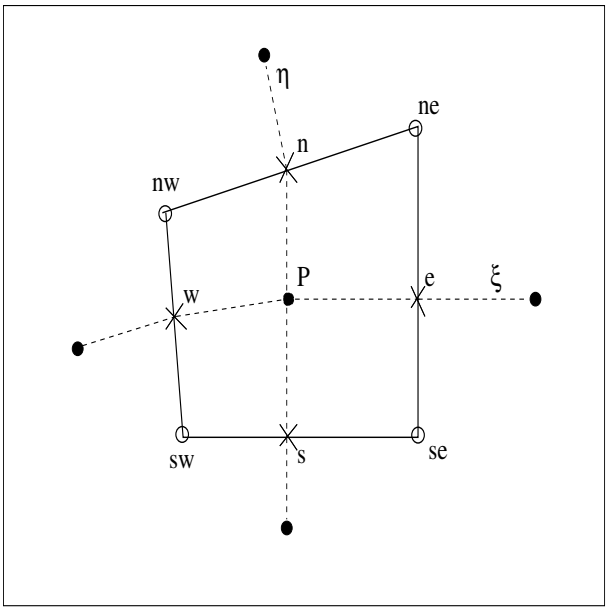


Figuur 8.2: a) Structured grid en b) Blok-structured grid

- Deze blok structuur is de basis voor het paralleliseren. Het verdelen van de rekenwerk over een aantal processoren ,waarbij het rekentijd (wachttijd) minder wordt.
- Lokale grid verfijning, dat is het verfijnen van een blok t.o.v. de andere blokken, zie paragraaf 8.4

Een controle volume is door de coordinaten van de acht hoekpunten (drie-dimensionaal) gespecificeerd. Hier zullen we een 2-dimensionale geometrie bekijken. In figuur

8.3 is de geometrie van een typisch niet-orthogonaal controle volume weergegeven. De vier coordinaten zijn functies van de indices i en j en deze worden in de arrays $x(i,j)$ en $y(i,j)$ opgeslagen.



Figuur 8.3: Een typisch non-orthogonaal cel

De coordinaten van het middelpunt P van een Cv wordt door de coordinaten van de hoekpunten bepaald:

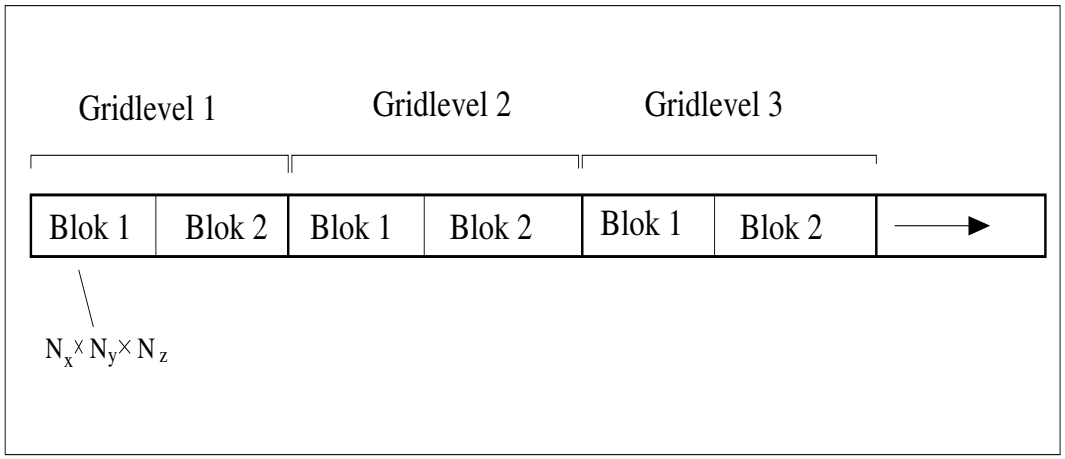
$$\underline{x}_P = \frac{1}{4}(\underline{x}_{ne} + \underline{x}_{nw} + \underline{x}_{sw} + \underline{x}_{se}) = \frac{1}{4}(\underline{x}_e + \underline{x}_w + \underline{x}_s + \underline{x}_n)$$

waarbij $\underline{x}$ de coordinaten (x, y) van de hoekpunten aangeeft.
Het totaal aantal controle volumes wordt door de gebruiker gespecificeerd. De gridpunten in xi-richting en de gridpunten in η-richting worden met ni respectievelijk nj aangegeven. Deze gridpunten worden $ni = nicv + 2$, $nj = njcv + 2$ en het geval van 3D is $nk = nkcv + 2$. Aan de randen worden geen virtuele punten gebruikt. Het verfijnen is gedaan door elke zijvlak van een grove Cv in twee segmenten te verdelen. Dit geeft op elk zijvlak een nieuwe gridpunt. Het verbinden van deze nieuwe gridpunten op de zijvlakken geeft vier nieuwe verfijnde Cv's. Deze procedure wordt herhaald totdat het maximaal gridlevel is bereikt.

Opslaan

De oplossing strategie in X-stream om een goede startoplossing de berekeningen op het fijne grid te beginnen, maakt gebruik van meerdere grids. Het grid wordt dan weer in een aantal blokken verdeeld. Per grid level dienen er veel variabelen opgeslagen te worden. In X-stream worden de variabelen voor alle gridlevels

in een één-dimensionale array opgeslagen. In figuur 8.4 is deze opslaan techniek geschetst.



Figuur 8.4: Opslaan van het data

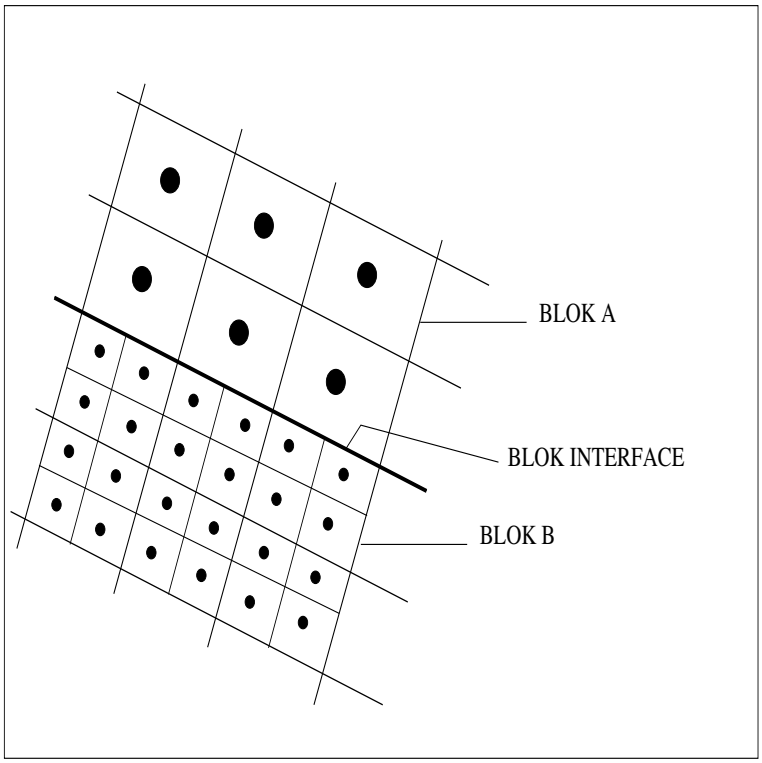
Hiervoor hebben we een index nodig die de locatie van een element voor willekeurige interne controle volume (i, j, k) voor een willekeurige blok (nbl) in een willekeurige grid k in het één-dimensionale array kan identificeren. Deze index is als volgt gedefinieerd:

$$CV(ngr, nbl, i, j, k) = nbl_start(ngr, nbl) + j + (i-1)(N_y+2) + (k-1)(N_y+2)(N_x+2) \tag{8.13}$$

waarbij ngr het grid level is, nbl is het blok nummer en $nbl_start(ngr, nbl)$ is de startindex van het vorige blok in grid ngr.
Elke blok bestaat uit $N_x \times N_y \times N_z$ Cv's, waarbij elke blok en Cv zes vlakken heeft. Elke blok kan twee verschillende randen hebben :
1- External rand: het randsegment valt samen met de randen van het rekendomein
2- Internal rand: De rand segmenten die twee blokken verbinden
De external rand wordt in X-stream met een getal geidentificeerd:
1. Inlet,
2. Outlet,
3. Symmetrie,
4. Wand.
De internal randen worden ook in een een-dimensionale array opgeslagen.

8.4 Local gridverdfijning

Een uitdaging in CFD is het simuleren van stromingen in rekendomeinen met gecompliceerde geometrieen en een hoge gradient in de oplossing. Daarbij moeten deeldomeinen waarbij hoge gradienten in de oplossing voorkomen erg verfijnd worden om een hoge nauwkeurigheid te krijgen. Het verfijnen van het hele rekendomein leidt tot onnodige toename van het computergeheugen en CPU tijd. Om deze problemen te ondervangen wordt slechts het specifieke rekendomein verfijnd. In feite wordt het rekendomein in een aantal blokken verdeeld, waarbij deze blokken elkaar niet overlappen en elke blok is in een eindig aantal controle volumes verdeeld. De gemeenschaplelijk randen van de blokken worden blok-interfaces genoemd. Local blok verfijning wordt gerealiseerd door elk grof controle volume in 8 (3D) nieuwe controle volumes te verdelen. Een voorbeeld van locaal blok verfijning is in figuur 8.5 weergegeven.



Figuur 8.5: Blok interface tussen een verfijnd blok B en niet-verfijnd blok A

8.5 Het bepalen van de snelheid op het zijvlak in Colocated grid

De colocated grid arrangement veroorzaakt problemen (oscillaties in de druk) wanneer CDS gebruikt wordt om de snelheden op het zijvlakken te bepalen. Rie and Chow hebben een methode bedacht om deze oscillaties te voorkomen. Zij hebben de snelheid op het zijvlak gecorrigeerd met een dempingsterm, die uit het verschil van de drukgradient en de geinterpoleerde drukgradient bestaat, zie vergelijking (8.5). Voor de afleiding van deze formule beschouwen we een drie-dimensionaal uniform grid en deze afleiding gaat als volgt:
De gediscretiseerde impulsvergelijking van cellen (i,j,k), (i+1,j,k) en (i+1/2,j,k) zijn

$$A_{i,j,k}U_{i,j,k} = \sum_{(i,j,k)} A_{nb}U_{nb} - (\frac{\partial p}{\partial x})_{(i,j,k)}V_{i,j,k}, \tag{8.14}$$

$$A_{i+1,j,k}U_{i+1,j,k} = \sum_{(i+1,j,k)} A_{nb}U_{nb} - (\frac{\partial p}{\partial x})_{(i+1,j,k)}V_{i+1,j,k}, \tag{8.15}$$

$$A_{i+1/2,j,k}U_{i+1/2,j,k} = \sum_{(i+1/2,j,k)} A_{nb}U_{nb} - (\frac{\partial p}{\partial x})_{(i+1/2,j,k)}V_{i+1/2,j,k}. \tag{8.16}$$

waarbij $V_{i,j,k}$ het volume van een Cv met index (i,j,k). De vergelijking (8.16) bevat de matrixcoefficienten op het zijvlak $A_{i+1/2,j,k}$. Deze termen worden met interpolatie bepaald, zie (8.23). De Taylor reeksen voor de snelheden $U_{i,j,k}$ en $U_{i+1,j,k}$ rondom het gridpunt $(i+1/2,j,k)$ zijn:

$$U_{i+1,j,k} = U_{(i+1/2,j,k)} + \frac{\Delta x}{2}(\frac{\partial U}{\partial x})_{(i+1/2,j,k)} + \frac{\Delta x^2}{8}(\frac{\partial^2 U}{\partial x^2})_{(i+1/2,j,k)} + \frac{\Delta x^3}{48}(\frac{\partial^3 U}{\partial x^3})_{(i+1/2,j,k)} + O(\Delta x^4), \tag{8.17}$$

$$U_{i,j,k} = U_{(i+1/2,j,k)} - \frac{\Delta x}{2}(\frac{\partial U}{\partial x})_{(i+1/2,j,k)} + \frac{\Delta x^2}{8}(\frac{\partial^2 U}{\partial x^2})_{(i+1/2,j,k)} - \frac{\Delta x^3}{48}(\frac{\partial^3 U}{\partial x^3})_{(i+1/2,j,k)} + O(\Delta x^4). \tag{8.18}$$

Na het sommeren van de vergelijkingen (8.17) en (8.18) geldt:

$$U_{i+1/2,j,k} = \frac{1}{2}(U_{(i+1,j,k)} + U_{(i,j,k)}) - \frac{\Delta x^2}{8}(\frac{\partial^2 U}{\partial x^2})_{(i+1/2,j,k)} + O(\Delta x^4) \tag{8.19}$$

De tweede afgeleide in vergelijking (8.19) bepalen we met CDS en dat gaat als volgt:

$$(\frac{\partial^2 U}{\partial x^2})_{(i+1/2,j,k)} = \frac{1}{\Delta x/2}(\frac{\partial U}{\partial x})_{(i+3/4,j,k)} - (\frac{\partial U}{\partial x})_{(i+1/4,j,k)}) = \frac{1}{\Delta x/2}$$
$$(U_{i+1,j,k} - U_{i+1/2,j,k}) - \frac{1}{\Delta x/2}(U_{i+1/2,j,k} - U_{i,j,k}) = \frac{1}{\Delta x^2/4}(U_{i+1,j,k} - 2U_{i+1/2,j,k} + U_{i,j,k}) \tag{8.20}$$

Dit invullen in (8.20) en met verwaarlozing van de $O(\Delta x^4)$ term geeft:

$$U_{i+1/2,j,k} = \frac{1}{2}(U_{(i+1,j,k)} + U_{(i,j,k)}) - \frac{\Delta x^2}{8}(\frac{U_{i+1,j,k} + U_{i,j,k} - 2U_{i+1/2,j,k}}{\Delta x^2/4}) \tag{8.21}$$

Nu kunnen we de vergelijkingen (8.14), (8.15) en (8.16) in de tweede term van de laatste vergelijking invullen dan volgt:

$$U_{i+1/2,j,k} = \frac{1}{2}(U_{(i+1,j,k)} + U_{(i,j,k)}) - \frac{1}{2}(\frac{V_{i+1,j,k}}{A_{i+1,j,k}} + \frac{V_{i,j,k}}{A_{i,j,k}})(\frac{\partial p}{\partial x})_{i+1/2,j,k} + \frac{1}{2}(\frac{V_{i+1,j,k}}{A_{i+1,j,k}}(\frac{\partial p}{\partial x})_{i+1,j,k} + \frac{V_{i,j,k}}{A_{i,j,k}}(\frac{\partial p}{\partial x})_{i,j,k}). \tag{8.22}$$

Voor het afleiden van vergelijking (8.22) hebben we de volgende vergelijkingen gebruikt:

$$\frac{\sum_{(i+1/2,j,k)} A_{nb}U_{nb}}{A_{i+1/2,j,k}} = \frac{1}{2}(\frac{\sum_{(i+1,j,k)} A_{nb}U_{nb}}{A_{i+1,j,k}} + \frac{\sum_{(i,j,k)} A_{nb}U_{nb}}{A_{i,j,k}}) + O(\Delta x^2). \tag{8.23}$$

$$\frac{V_{i+1/2,j,k}}{A_{i+1/2,j,k}} = \frac{1}{2}(\frac{V_{i+1,j,k}}{A_{i+1,j,k}} + \frac{V_{i,j,k}}{A_{i,j,k}}) + O(\Delta x^2). \tag{8.24}$$

De vergelijking (8.22) kunnen we ook als volgt schrijven:

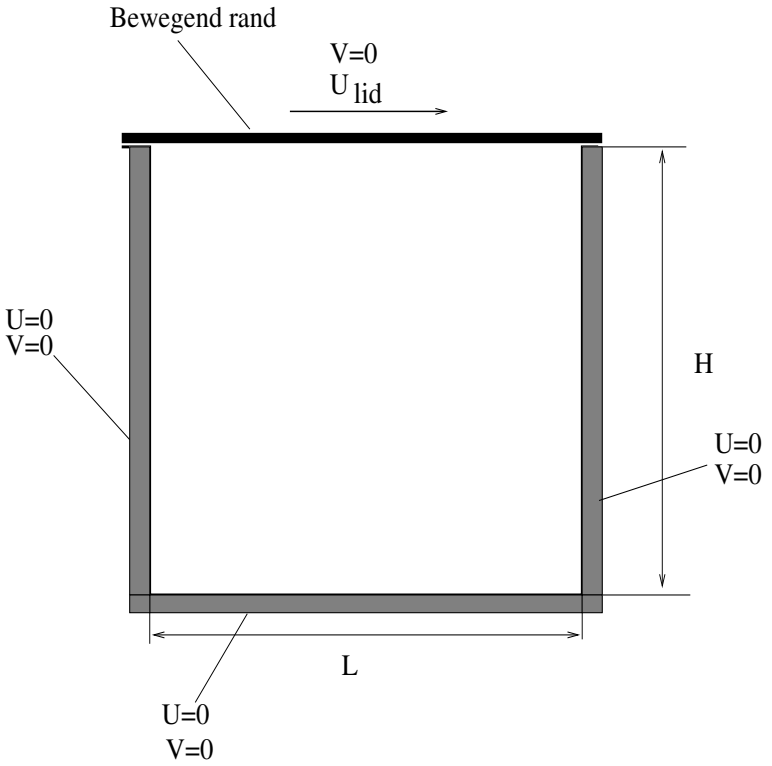
$$U_{i+1/2,j,k} = \overline{U}_{i+1/2,j,k} - (\overline{\frac{V_{i,j,k}}{A_{i,j,k}}})_{i+1/2,j,k}(\frac{\partial p}{\partial x})_{i+1/2,j,k} - (\overline{\frac{V_{i,j,k}}{A_{i,j,k}}\frac{\partial p}{\partial x}})_{i+1/2,j,k} \tag{8.25}$$

Als wij de naamgeving uit hoofstuk 5 aanhouden, dan krijgen we precies de vergelijking (8.5), namelijk

$$U_e = \overline{U_e} - (\overline{\frac{V_P}{A_P}})_e((\frac{\partial p}{\partial x})_e - (\overline{\frac{\partial p}{\partial x}})_e). \tag{8.26}$$

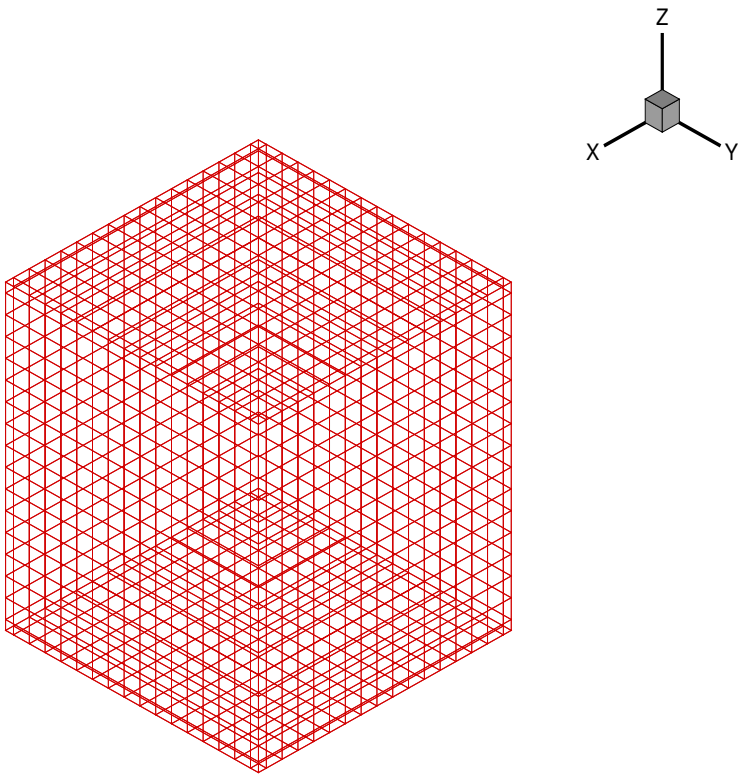
8.6 Testcase

De meest gebruikte testcase om oplosmethoden van incompressibele stromingen te testen is zondermeer de lid driven cavity. De lid driven cavity is een gesloten kubus waarbij de bovenste wand met een constante snelheid U_{lid} beweegt. Deze bewegende wand veroorzaakt via het viskeuse effect aan de fluid-solid interface een stroming in het inwendige. De geometrie van lid driven cavity met de randvoorwaarden is in figuur 8.6 voor 2D geschetst. De andere randen kunnen van het type wand of symmetrie zijn.



Figuur 8.6: Lid driven cavity

Voor de berekeningen in X-stream hebben we een drie-dimensionaal grid nodig, in figuur 8.7 is dit grid geschetst. Het drie-dimensionale grid is uniform en heeft een afmeting van $1 \times 1 \times 1$. Het grid bevat $16 \times 16 \times 16$ controle volumes.
Randvoorwaarden:
De bovenste wand beweegt met een constante snelheid $V = 1$ en de andere wanden (East, West, North, South en Bottom) bewegen niet, dus $U = V = W = 0$.
Stromingsparameters:
De relaxatiefactoren voor de snelheden (u,v,w) zijn $\alpha_u = \alpha_v = \alpha_w = 0.7$ en de relaxatiefactor voor de druk p is $\alpha_p = 0.4$. Tijdens een SIMPLE iteratie lossen we een aantal gelineariseerd stelsel vergelijkingen op met de SIP solver. Voor deze



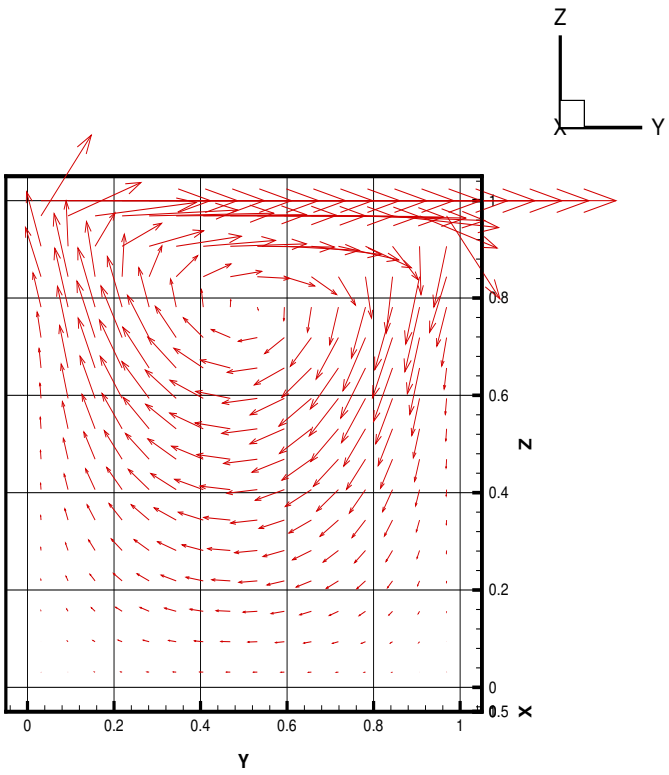
Figuur 8.7: Het grid van het 3D lid driven cavity

solver hebben we een relaxatiefactor $\alpha_{SIP} = 0.92$ gekozen. Voor het discretiseren van de convectieve term zijn drie mogelijke schema's : UDS, CDS en het defferred correctie schema. Wij hebben voor UDS gekozen. We spreken van convergentie van SIMPLE wanneer δ zoals gedefinieerd in paragraaf 8.2 voor alle variabelen (u, v, w en p) onder de waarde van 10^{-4} is gekomen. De dichtheid is $\rho = 1$.
Om met verschillende Reynoldsgetallen berekeningen uit te voeren, hebben we verschillende waarde voor de viscositeit gekozen. De keuzes van de viscositeit ν zijn 0.1, 0.01, 0.05 en 0.001. Het bepalen van de Reynoldsgetallen gaat als volgt:

$$Re_L = \frac{U_{lid} L}{\nu} = \frac{1. * 1.}{0.01} = 100$$

waarbij L de lengte is van de cavity en U_{lid} is de snelheid van de bovenste bewegende wand. Op analoge wijze volgen de andere Reynoldsgetallen $Re = 10$, $Re = 500$ en $Re = 1000$. In figuren 8.8-8.10 is het vectorveld van de stroming in

de cavity bij $Re = 10$, $Re = 100$ en $Re = 1000$ weergegeven. Deze figuren zijn de dwarsdoorsneden van de cavity op $x = 0.5$.

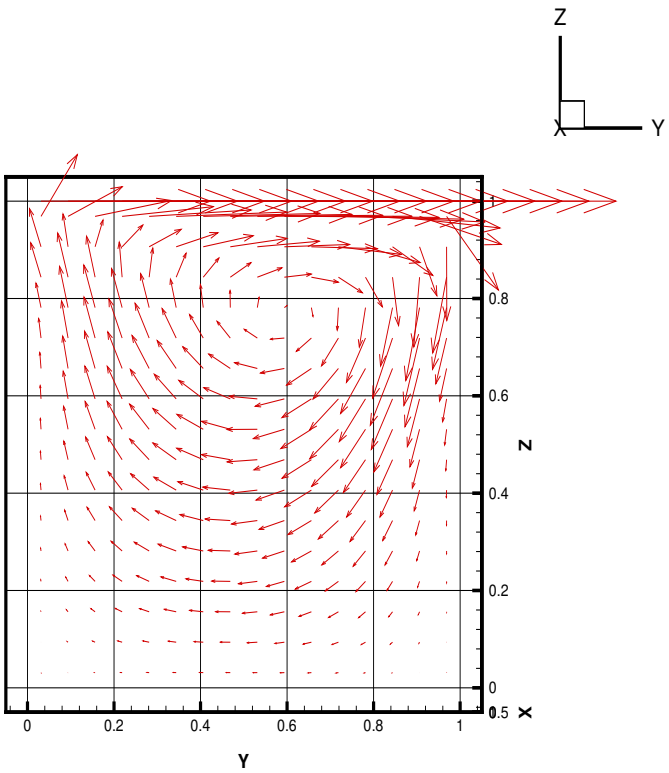


Figuur 8.8: Dwarsdoorsnede van de driven cavity op $x = 0.5$ voor $Re = 10$

We voeren in X-stream berekeningen uit met verschillende startoplossingen voor de variabelen:

- Als wij de berekeningen op een single grid (SIMPLE-SG) uitvoeren, kiezen we startoplossing gelijk aan nul voor de snelheden en de druk,
- Als we de berekeningen op een multi-grid uitvoeren dan wordt de startoplossing voor grid $k + 1$ verkregen door interpolatie (prolongatie) van de geconvergeerde oplossing op grid k (blz. 82). Deze oplossing strategie duiden we aan met (SIMPLE-PG).

In tabellen 8.1-8.4 zijn de SIMPLE iteraties voor de grids $4 \times 4 \times 4$, $8 \times 8 \times 8$ en $16 \times 16 \times 16$ bij verschillende Reynoldsgetallen en verschillende startoplossing (SIMPLE-SG en SIMPLE-PG) weergegeven.

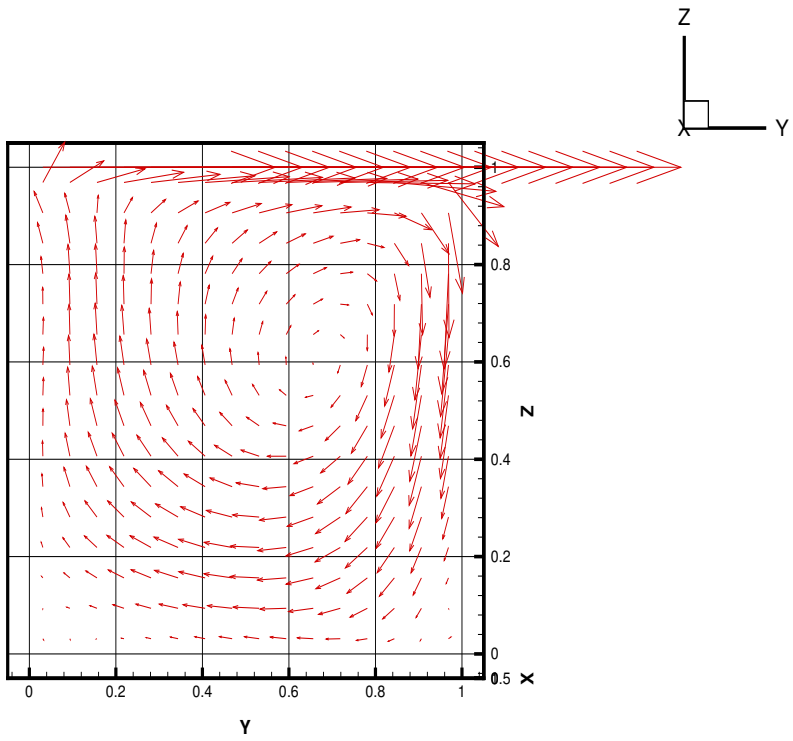


Figuur 8.9: Dwarsdoorsnede van de driven cavity op $x = 0.5$ voor $Re = 100$

$rekenrooster$	$SIMPLE - SG$	$SIMPLE - PG$
$4 \times 4 \times 4$	20	20
$8 \times 8 \times 8$	27	23
$16 \times 16 \times 16$	89	52

Tabel 8.1: SIMPLE iteraties voor $Re = 10$

Uit deze tabellen blijkt dat het beginnen met een goede startvector (SIMPLE-PG) op het fijnste grid voordeliger is dan de berekeningen met nul startoplossing (SIMPLE-SG) op het fijnste grid te beginnen. Zo zien we dat bij $Re = 10$ het aantal SIMPLE iteraties met PG slechts 52 voor grid $16 \times 16 \times 16$, terwijl SIMPLE met SG 89 iteraties nodig heeft. Verder blijkt dat de oplosmethode SIMPLE traag convergeert naarmate het grid verfijnd wordt. Verder hebben we het residu verloop van SIMPLE met beide startoplossingen bekijken. In figuur 8.11 is de



Figuur 8.10: Dwarsdoorsnede van de driven cavity op $x = 0.5$ voor $Re = 1000$

$rekenrooster$	$SIMPLE-SG$	$SIMPLE-PG$
$4 \times 4 \times 4$	21	21
$8 \times 8 \times 8$	24	21
$16 \times 16 \times 16$	64	43

Tabel 8.2: SIMPLE iteraties voor $Re = 100$

norm van het residu van alle variabelen (u, v, w en p) verkregen met SIMPLE-SG weergegeven. Deze resulaten zijn verkregen voor $32 \times 32 \times 32$ controle volumes, convergentie criterium $\delta = 10^{-4}$, $\alpha_u = \alpha_v = \alpha_w = 0.7$ en $\alpha_p = 0.4$. In deze figuur valt ons op dat het residu van de druk nogal oscilleert en dat de druk dan ook de meeste moeite heeft om $\delta = 10^{-4}$ te halen. Het residu van de snelheid u dreigt in het begin te ontploffen, vervolgens met minder oscillerend gedrag dan de andere variabelen convergeert het residu keurig naar 10^{-4}. Verder hebben we het

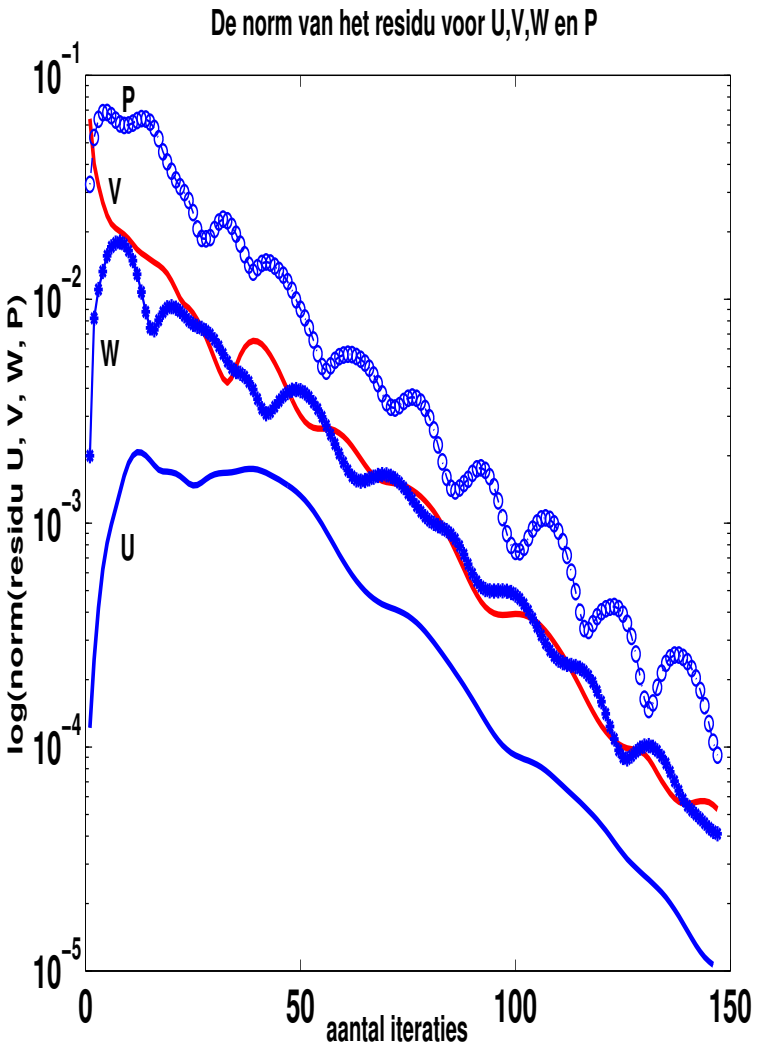
$rekenrooster$	$SIMPLE-SG$	$SIMPLE-PG$
$4 \times 4 \times 4$	19	19
$8 \times 8 \times 8$	25	21
$16 \times 16 \times 16$	81	51

Tabel 8.3: SIMPLE iteraties voor $Re = 500$

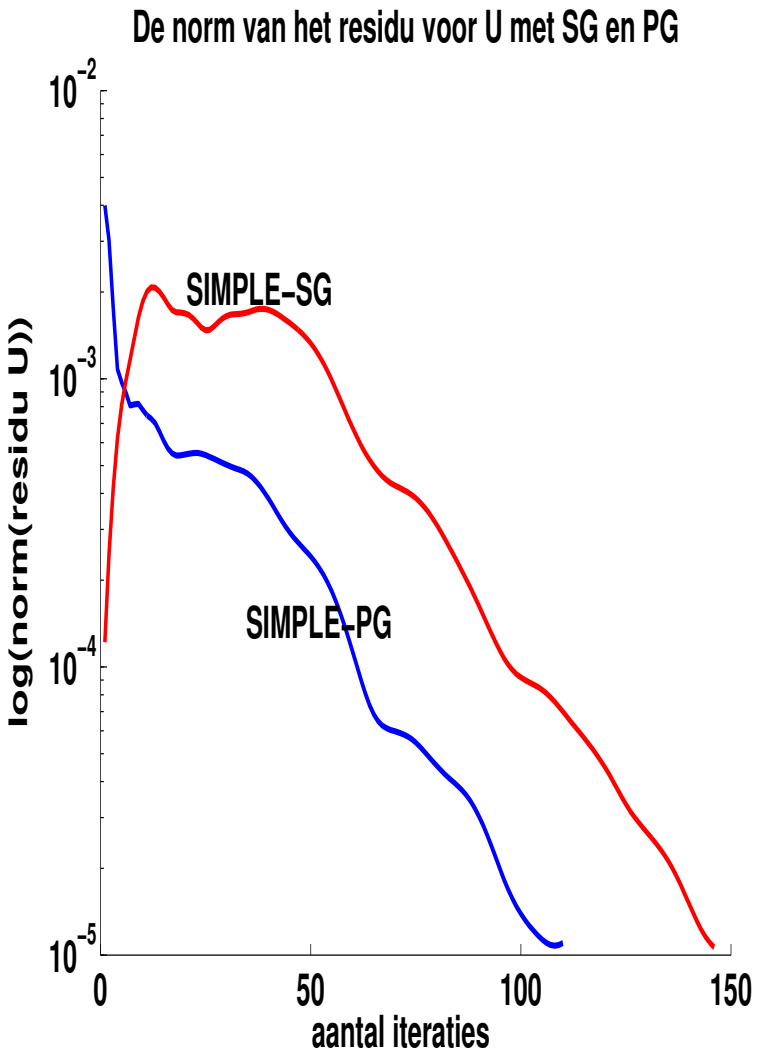
$rekenrooster$	$SIMPLE-SG$	$SIMPLE-PG$
$4 \times 4 \times 4$	24	24
$8 \times 8 \times 8$	31	30
$16 \times 16 \times 16$	52	48

Tabel 8.4: SIMPLE iteraties voor $Re = 1000$

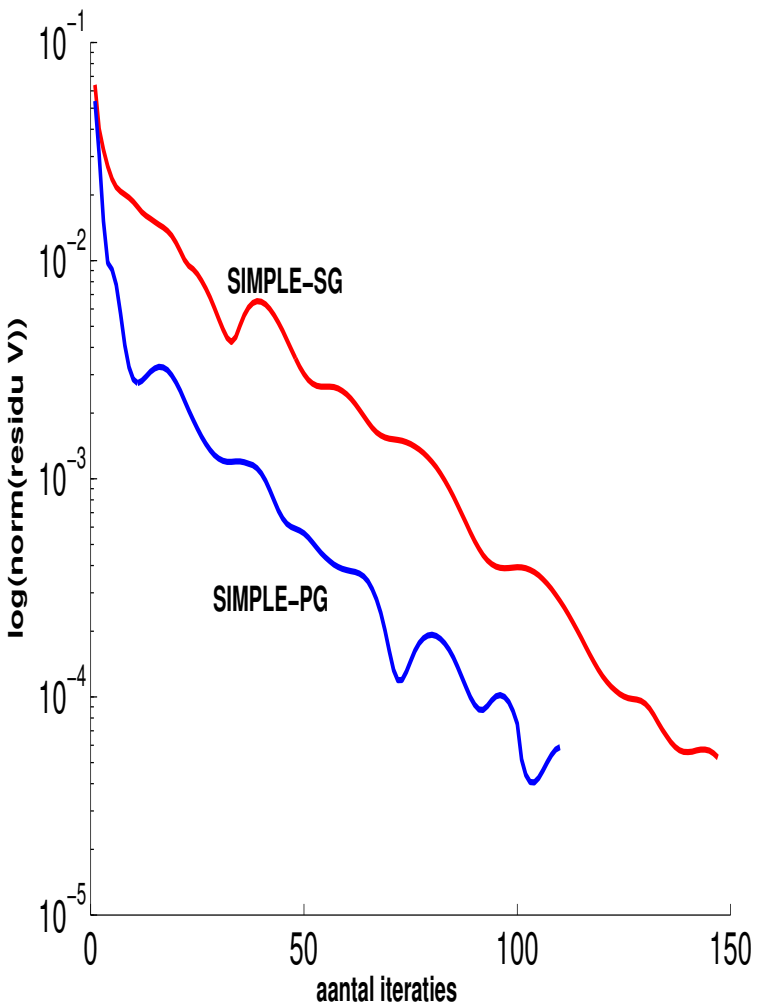
residu van SIMPLE met beide startoplossing PG en SG vergeleken. Dat hebben we voor elke variabele bekeken. In figuren 8.12-8.15 zijn deze resulaten weergegeven. In deze figuren kunnen we constateren dat het residu verloop verkregen met SIMPLE-PG altijd onder het residu van SIMPLE-SG. Het voordeel om met een goede startoplossing te beginnen (PG) boven (SG) wordt met deze resultaten nogmaals bevestigd.



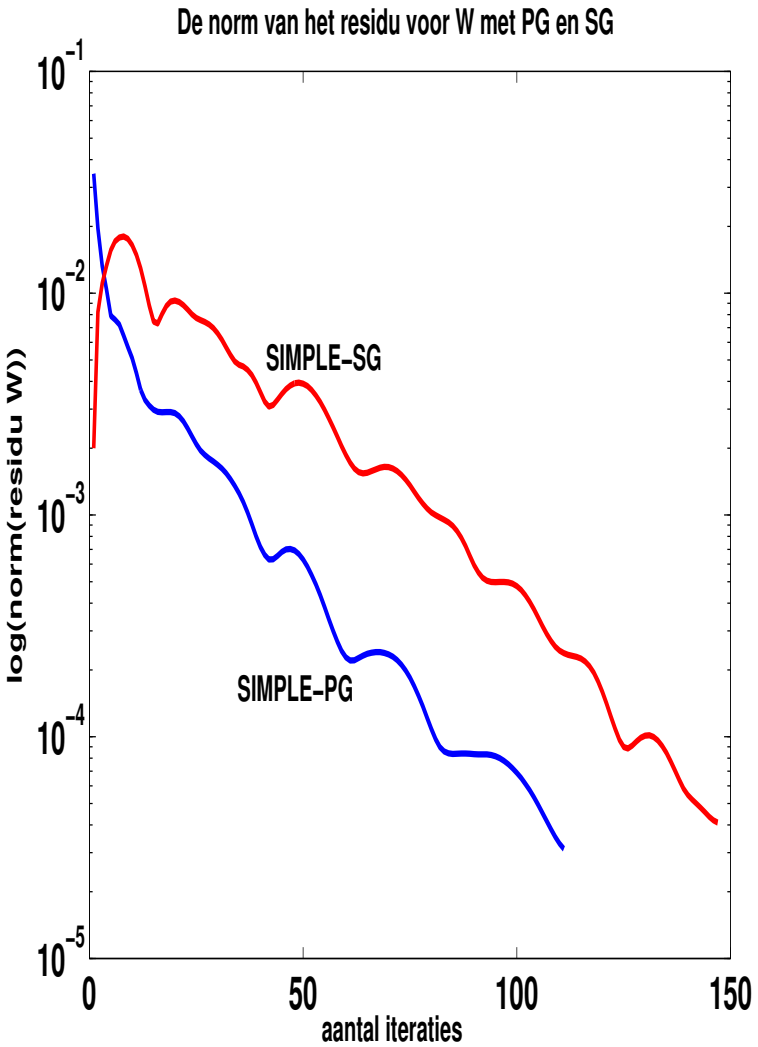
Figuur 8.11: Het convergentie gedrag van de snelheden u, v, w en de druk p voor $Re = 1000$



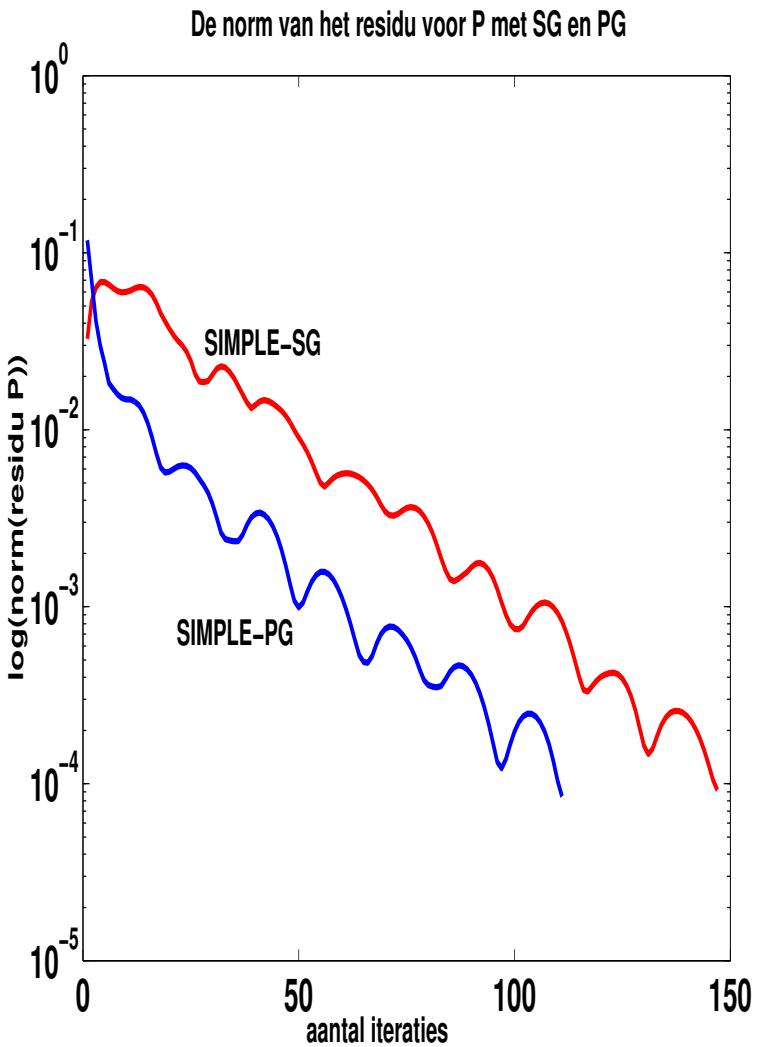
Figuur 8.12: Het convergentie gedrag van de snelheidscomponent u met SIMPLE-SG en SIMPLE-PG voor $Re = 1000$



Figuur 8.13: Het convergentie gedrag van de snelheidscomponent v met SIMPLE-SG en SIMPLE-PG voor $Re = 1000$



Figuur 8.14: Het convergentie gedrag van de snelheidscomponent w met SIMPLE-SG en SIMPLE-PG voor $Re = 1000$



Figuur 8.15: Het convergentie gedrag van de druk p met SIMPLE-SG en SIMPLE-PG voor $Re = 1000$

Hoofdstuk 9

Multigrid methode in X-stream

In hoofdstuk 7 hebben we gezien hoe traag de iteratieve methoden zijn om 2D Laplace vergelijking op te lossen , naarmate het grid verfijnd wordt ofwel naarmate het aantal onbekenden toeneemt. Elke iteratieve methode heeft een andere convergentie gedrag, afhankelijk van het conditie getal van matrix A. Naarmate het aantal onbekenden toeneemt, neemt het conditie getal van A ook toe, waarbij het convergentie gedrag van de solver slechter wordt. Verder blijkt dat het convergentie gedrag van deze solvers in de eerste iteraties erg goed is, maar daarna wordt dit gedrag met de iteratie alleen maar slechter. De oorzaak hiervan is de aanwezigheid van laag frequente fouten in de oplossing. Als het grid verfijnd wordt, domineren deze fouten het convergentie gedrag van de solvers. Het combineren van deze solvers met multigrid is erg efficiënt gebleken om de convergentie van deze solvers te versnellen. Deze trage convergentie gedrag vertoont zich ook in het SIMPLE algoritme in het vorige hoofdstuk om een stromingsprobleem op te lossen. Hierbij hebben we gezien dat SIMPLE traag convergeert op fijne grids. De reden van deze convergentie is de zwakke koppeling tussen de variabelen en de niet-lineariteit van de op te lossen vergelijkingen. Deze zwakke koppeling samen met de noodzaak om met een erg fijn grid te rekenen, voor het verminderen van de discretisatie fout, maken de berekeningen erg duur. Om deze reden zullen we in dit hoofdstuk de oplosmethode SIMPLE met een multigrid variant versnellen. Het versnellen van SIMPLE kunnen we op twee manieren realiseren:

- De inner iteraties van SIMPLE versnellen. Tijdens de SIMPLE iteraties lossen we met SIP solver een stelsel vergelijkingen $A\phi = b$ voor de snelheden en de drukcorrectie. Het blijkt dat SIP solver maximaal 3 inner iteraties nodig heeft om het snelheidsveld te bepalen. Bij het oplossen van de drukcorrectie heeft SIP solver maximaal 10 inner iteraties nodig. Het versnellen van de SIP solver met multigrid heeft dus alleen zin voor de drukcorrectie. De lineaire vergelijking van de drukcorrectie is een Poisson vergelijking en in hoofdstuk 7 hebben we gezien hoe efficient multigrid versnelling voor SIP solver is.

- De SIMPLE iteraties versnellen. Hierbij gebruiken we SIMPLE algoritme

als smoother.

Onze aandacht gaat uit naar de versnelling van de SIMPLE iteraties met een multigrid methode. Voor het implementeren van MGM in X-stream hebben wij de volgende keuzes moeten maken:

- 1. Vanwege de niet-lineariteit van de op te lossen incompressibele vergelijkingen zullen we het Full Approximation Scheme (FAS) algoritme beschouwen. Het voordeel van FAS ten opzichte van het Correction Scheme (CS) is dat de oplossingen en dus niet alleen het residu van het fijne grid geinterpoleerd worden naar het grove grid. Zo worden de niet-lineaire operator en de bronterm continu verbeterd.

- 2. Voor de cycles is de V-cycle gebruikt: We beginnen het stelsel op het fijnste grid op te lossen, vervolgens bezoeken we de onderste grids tot het grofste bereikt is, dan keren we terug totdat weer het fijnste grid is bereikt.

- 3. De Dirichlet randvoorwaarden (Inlet en Wand) worden op elke grid gespecificeerd. De druk op de randen wordt met extrapolatie van de waarden van de eerste twee interne gelegen gridpunten verkregen. De randvoorwaarde van het outlet is een Neuman randvoorwaarde en om deze reden krijgt de waarde van de gridpunt aan de outlet rand elke SIMPLE iteratie de waarde van het eerste interne gelegen gridpunt. Op dit moment hebben we alleen met Dirichlet randvoorwaarden gewerkt.

- 4. De matrices en de rechterleden voor de snelheden en de druk worden voor elk grid G_k bepaald. Dus niet vanuit G_{k+1} geinterpoleerd.

- 5. De convergentie check wordt slechts op het fijnste grid van elke V-cycle gedaan.

Het niet-lineaire Two grid FAS algoritme luidt:

De exacte oplossing voor een variabele (u, v, w em p) op grid G_k, Φ_k, voldoet aan de volgende vergelijking:

$$[\mathcal{A}_k]\{\Phi_k\} = \{\mathcal{S}_k\} \tag{9.1}$$

waarbij $\mathcal{A}_k$ en $\mathcal{S}_k$ zijn de matrixcoefficient, respetievelijk de bronterm. Na N_k^r SIMPLE iteraties op het fijnste rooster G_k (pre-smoothing), wordt een tussenoplossing ϕ_k (u, v, w en p) verkregen. Als wij de tussenoplossing ϕ_k in (9.1) invullen dan krijgen we:

$$[A_k]\{\phi_k\} = \{S_k\} - \{R_k\} \tag{9.2}$$

waarbij R_k is het residu, de matrix A_k en de brontem S_k zijn gebaseerd op de tussenoplossing ϕ_k. Hiermee bedoelen we dat de tussenoplossing ϕ_k ondermeer

gebruikt wordt om de convectieve flux te bepalen. Het verschil van (9.1) en (9.2) geeft :

$$[\mathcal{A}_k]\{\Phi_k\} = \{\mathcal{S}_k\} + [A_k]\{\phi_k\} - \{S_k\} + \{R_k\} \tag{9.3}$$

De vergelijking (9.3) vormt de basis voor het multigrid algoritme.

Het opstellen van de vergelijking op het grove grid G_{k-1}:

De tussenoplossing ϕ_k en het residu R_k worden geinterpoleerd naar grid G_{k-1} en we geven deze aan met $\tilde{\phi}_{k-1}$ en $\tilde{R}_{k-1}$. Op grid G_k worden de matrix A_{k-1} en het rechterlid S_{k-1} verkregen op analoge wijze als op grid G_k. Deze matrix A_{k-1} en het bijbehorende rechterlid S_{k-1} zullen we met een dakje aangeven $(\hat{A}_{k-1}, \hat{S}_{k-1})$. De matrix A_{k-1} en het bijbehorende rechterlid S_{k-1} gebaseerd op de geinterpoleerde tussenoplossing $\tilde{\phi}_k$ zullen we met $\tilde{A}_{k-1}$ en $\tilde{S}_{k-1}$ aangeven. Nu kunnen we de vergelijking opstellen voor grid G_{k-1}, namelijk:

$$\{\hat{A}_{k-1}\}\{\hat{\phi}_{k-1}\} = \{\hat{S}_{k-1}\} + \underline{\{\tilde{A}_{k-1}\}\{\tilde{\phi}_{k-1}\} - \{\tilde{S}_{k-1}\} + \{\tilde{R}_{k-1}\}} \tag{9.4}$$

Tijdens de SIMPLE iteraties op grid G_{k-1} veranderen nu alleen de termen met een dakje. De onderstreepte term in (9.4) verandert niet tijdens de SIMPLE iteraties op grid G_{k-1}. Deze term fungeert als een extra bronterm in vergelijking (9.4). De matrix $\hat{A}_{k-1}$ en de bijbehorende bronterm S_{k-1} worden op grid G_{k-1} geinitieerd door de matrix $\tilde{A}_{k-1}$ en $\tilde{S}_{k-1}$. De geinterpoleerde variabele $\tilde{\phi}_{k-1}$ wordt als volgt bepaald:

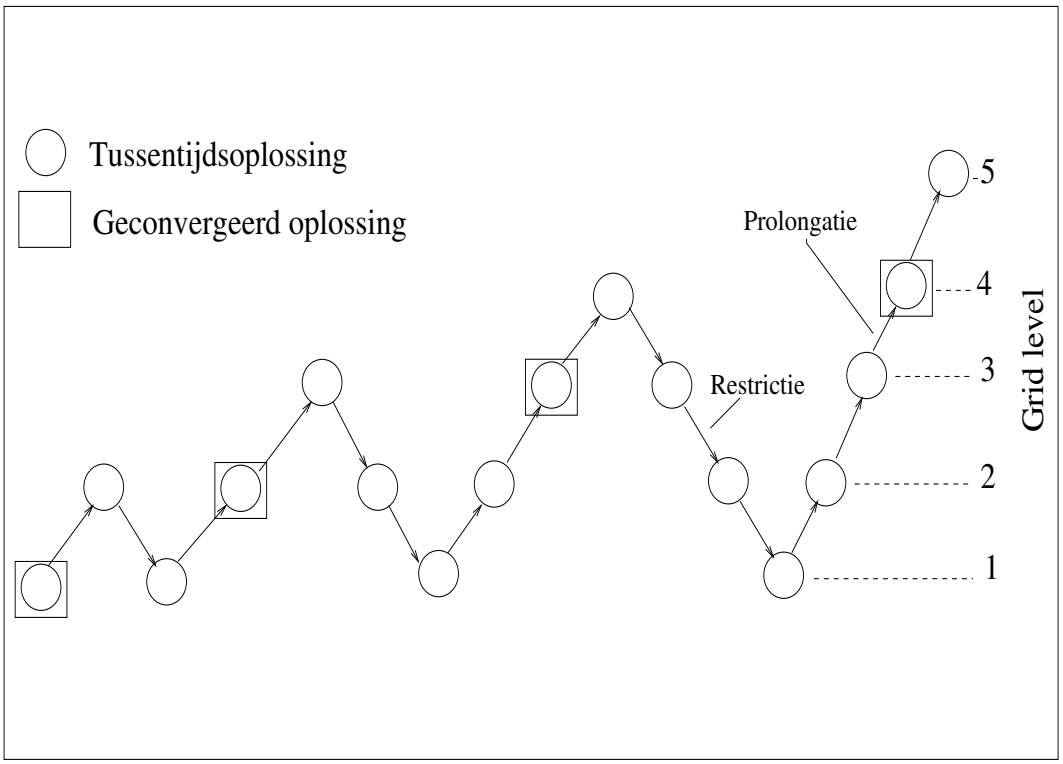
$$\tilde{\phi}_{k-1} = I_k^{k-1}\phi_k \tag{9.5}$$

waarbij I_k^{k-1} een restrictie operator is.
Op grid G_{k-1} doen we een aantal N_{k-1}^r SIMPLE iteraties met de startoplossing $\tilde{\phi}_{k-1}$ ($\tilde{u}$, $\tilde{v}$ en $\tilde{w}$). De druk vergelijking is lineair en om deze reden wordt de druk op grid G_{k-1} met nul gestart. Na deze N_{k-1}^r SIMPLE iteraties krijgen we een nieuwe oplossing $\hat{\phi}_{k-1}$ ($\hat{u}$, $\hat{v}$, $\hat{w}$ en $\hat{p}$). Het verschil $(\hat{\phi}_{k-1} - \tilde{\phi}_{k-1})$ wordt naar grid G_k gextrapoleerd om de eerder verkregen oplossing ϕ_k^{old} op grid G_k te corrigeren, ofwel :

$$\phi_k^{new} = \phi_k^{old} + I_{k-1}^k(\hat{\phi}_{k-1} - \tilde{\phi}_{k-1}) \tag{9.6}$$

waarbij I_{k-1}^k een prolongatie operator is.
Op grid G_k doen we N_f^p SIMPLE iteraties (post-smoothing). De berekeningen op G_k zijn geinitieerd door de ϕ_k^{new}. Hier hebben we slechts één multigrid V-cycle van FAS bescheven. Het multigrid algoritme FAS met V-cycle zullen we met MG-V aanduiden. Het rekenwerk met V-cycle kan gereduceerd worden als wij op het fijn grid met een goede startvector beginnen. In het vorige hoofdstuk hebben we

gezien dat het beginnen van de berekeningen van SIMPLE met goede startoplossing (SIMPLE-PG) voordeliger zijn dan de berekeningen met een nulstartoplossing (SIMPLE-SG). Om deze reden zullen we het bescheven multigrid methode met V-cycle combineren met PG, dus we beginnen de berekeningen op het grofste grid G_1 totdat SIMPLE convergeert, vervolgens wordt de geconvergeerde oplossing van G_1 geextrapoleerd naar grid G_k het multigrid FAS met V-cycles worden gestart, zie figuur 9.1. Dit leidt tot het Full Multigrid algoritme (FMG-FAS). In figuur 9.1 geven we met een vierkant aan dat de oplossing geconvergeerd is na een aantal V-cycles.



Figuur 9.1: Full multigrid methode

Het complete (FMG-FAS) multigrid algoritme kunnen we als volgt samenvatten:
BEGIN MG-PROCEDURE

- IF het huidige grid het grofste grid is dan:
  $\rightsquigarrow$ Los de vergelijkingen voor de snelheden en de druk tot dat volledige convergentie is verkregen.
  $\rightsquigarrow$Extrapoleer de geconvergeerde oplossing naar het volgende grid.

- Else

$\rightsquigarrow$ Pre-smoothing : Doe een aantal SIMPLE iteraties (N_k^r) op het fijn grid G_k en een benaderde oplossing is verkregen.
$\rightsquigarrow$ De oplossing en de residu van het fijne grid G_k worden geinterpoleerd (Restrictie) naar het grove grid G_{k-1}
$\rightsquigarrow$ V-cycle is gestart (N_{cyc}).
$\rightsquigarrow$ Doe een aantal SIMPLE iteraties (N_{k-1}^r) op het grove grid G_{k-1}
$\rightsquigarrow$ De correcties op het grove grid G_{k-1} zijn bepaald en gextrapoleerd (Prolongatie) naar het fijne grid G_k.
$\rightsquigarrow$ Doe een aantal SIMPLE iteraties (N_k^p) op het fijne grid G_k, als het nodig is wordt een nieuwe V-cycle is gestart.
END IF

END MG-PROCEDURE
Het bepalen van $\tilde{R}_{k-1}$

Voor het bepalen van het residu van de snelheden van het fijne grid naar het grove grid, hebben we doordat wij EVM met 'colocated grid arrangement' hebben gewerkt, geen interpolatie methoden gebruikt. Het residu $\tilde{R}_{k-1}$ is de som van de residuen van 8 Cv's die binnen een grof Cv zitten, zie figuur 9.2. We geven de index van een grof Cv aan met (ig, jg, kg) en de index van een Cv op het fijn grid aan met (if, jf, kf).

$$\tilde{R}_{k-1}^{ig,jg,kg} = R_k^{if,jf,kf} + R_k^{if,jf-1,kf} + R_k^{if,jf,kf-1} + R_k^{if,jf-1,kf-1} + \\ + R_k^{if-1,jf,kf} + R_k^{if-1,jf-1,kf} + R_k^{if-1,jf,kf-1} + R_k^{if-1,jf-1,kf-1} \quad (9.7)$$

Het residu $\tilde{R}_{k-1}$ bepalen we voor de snelheden u, v en w.
Het bepalen van de massafluxen op G_{k-1}
De initiaal massafluxen door een zijvlak van een Cv op het grove grid G_{k-1} is de som van de massafluxen door de vier zijvlakken (3D) van een Cv op het fijn grid G_k, zie figuur 9.3. Deze massafluxen worden voor de zijvlakken east, north en top als volgt bepaald.
massaflux door zijvlak east

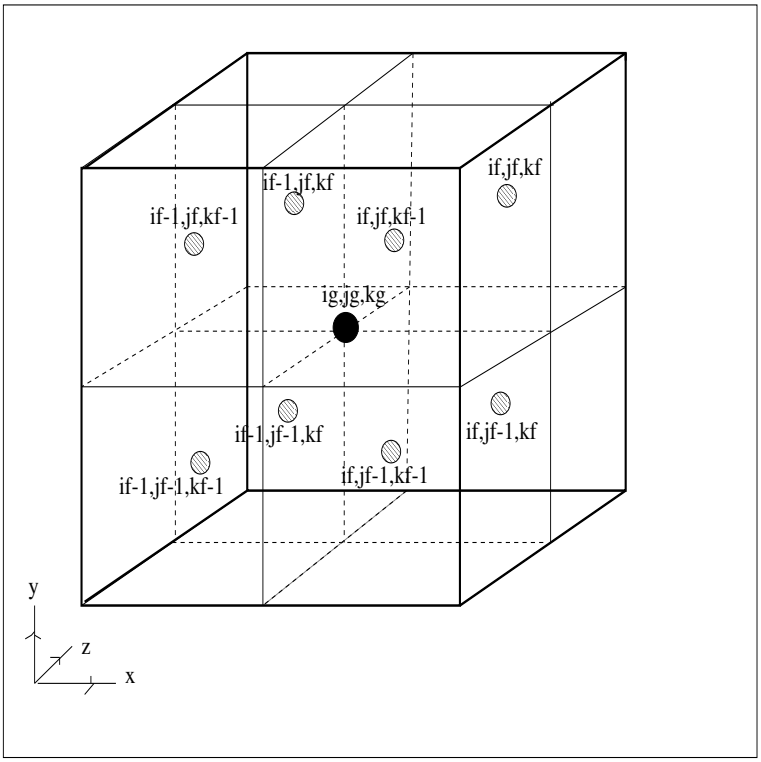
$$\tilde{\dot{m}}_e^{ig,jg,kg} = \dot{m}_e^{if,jf,kf} + \dot{m}_e^{if,jf-1,kf} + \dot{m}_e^{if,jf,kf-1} + \dot{m}_e^{if,jf-1,kf-1} \quad (9.8)$$

massaflux door zijvlak north

$$\tilde{\dot{m}}_n^{ig,jg,kg} = \dot{m}_n^{if,jf,kf} + \dot{m}_n^{if,jf,kf-1} + \dot{m}_n^{if-1,jf,kf} + \dot{m}_n^{if-1,jf,kf-1} \quad (9.9)$$

massaflux door zijvlak top

$$\tilde{\dot{m}}_t^{ig,jg,kg} = \dot{m}_t^{if,jf,kf} + \dot{m}_t^{if,jf-1,kf} + \dot{m}_t^{if-1,jf-1,kf} + \dot{m}_t^{if-1,jf,kf} \quad (9.10)$$



Figuur 9.2: Het bepalen van $\tilde{R}$

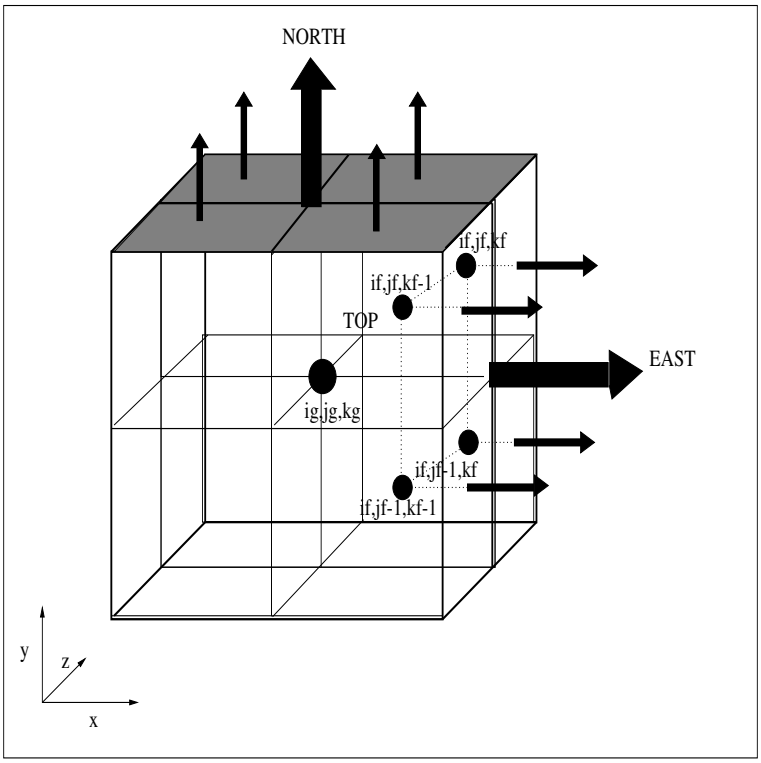
De discretisatie methode EVM met 'colocated grid arrangement' garandeert ons dat wanneer de massafluxen op het fijne grid aan de continuiteitsvergelijking voldoen, dan voldoen de fluxen (9.8-9.10) op de grove grid ook aan de continuiteitsvergelijking. Deze massafluxen worden in de eerste SIMPLE iteratie gebruikt om de matrix coefficient op te bouwen. Tijdens de SIMPLE iteraties op het grove grid G_{k-1} in een V-cycle hoeven we de massafluxen $\tilde{m}_e$,$\tilde{m}_n$ en $\tilde{m}_t$ slechts te corrigeren met de massaflux correcties($\delta\dot{m}_e$, $\delta\dot{m}_n$ en $\delta\dot{m}_t$), die bepaald zijn door de snelheidscorrecties, zie [6], [8], [9], [10] en [13]. Deze snelheden correcties zijn niet de correcties ($u^{'}$, $v^{'}$ en $u^{'}$) in het SIMPLE algoritme (blz. 34), maar deze correctie zijn gedefinieerd door:

$$uc = \hat{u} - \tilde{u}, \tag{9.11}$$

$$vc = \hat{v} - \tilde{v}, \tag{9.12}$$

$$wc = \hat{w} - \tilde{w}. \tag{9.13}$$

Op grove grids van een V-cycle wordt met deze snelheidscorrectie gerekend en dat geldt ook voor vergelijking (8.5), waarbij $\overline{(U)}_e$ vervangen wordt door $\overline{(\hat{u} - \tilde{u})}_e$, zie



Figuur 9.3: massafluxen door de zijvlakken

[6], [8], [9], [10] en [13]. De massaflux correcties op grid G_{k-1} zijn dan:

$$\delta \dot{m}_e = \rho \; uc \; S_e, \tag{9.14}$$

$$\delta \dot{m}_n = \rho \; vc \; S_n, \tag{9.15}$$

$$\delta \dot{m}_t = \rho \; wc \; S_t. \tag{9.16}$$

Op het grid G_{k-1} blijven deze de fluxen bepaalt met $\tilde{u}$, $\tilde{v}$ en $\tilde{w}$ constant.

9.1 Restrictie en prolongatie

Naast de relaxatie van SIMPLE en de multigrid cycle, zijn er nog twee belangrijke componenten binnen multigrid procedure, namelijk de restrictie en de prolongatie operatoren. De restrictie operator I_k^{k-1} wordt gebruikt om de variabelen van het fijne grid naar het grove grid te interpoleren. De prolongatie operator I_{k-1}^k extrapoleert juist de correcties van het grove grid naar het fijne grid.
De restrictie en de prolongatie procedures hangen af van:

- Hoe zijn de variabelen op het rekenrooster gelocaliseerd.

- Hoe worden de grove en de fijne grids gegenereerd. In X-stream worden door halveren van de stapgroote van het grofste grid in alle richtingen fijne grids gegeneerd.

Voor EVM met 'colocated grid arrangement' zijn de variabelen u, v, w en p in het midden van een Cv gelocaliseerd en om deze reden zijn de restrictie en de prolongatie operatoren voor alle variabelen hetzelfde. Dat is een groot voordeel t.o.v. staggered grid arrangement.

9.1.1 Restrictie

In figuur 9.2 is een grove Cv in drie dimensies weergegeven. Na het verfijnen van deze Cv krijgen we acht Cv's. De indices (ig, jg, kg) op het grove grid bepalen we als volgt:

$$ig = if/2 + 1$$
$$jg = jf/2 + 1$$
$$kg = kf/2 + 1$$

Voor gestructuerend grids worden vaak bilineaire (2D) of trilineaire (3D) interpolaties gebruikt om de variabelen of de correcties te interpoleren. Een andere manier om de variabelen of correcties van het fijne grid naar het grove grid te brengen is door gebruik te maken van het gradient in het midden van een Cv:

$$\phi_C = \frac{1}{N_f} \sum_{i=1}^{N_f} [\phi_{F_i} + (grad\phi)_{F_i} \cdot (r_C - r_{F_i})], \tag{9.17}$$

waarbij N_f het aantal fijne rooster Cv's in één grof Cv (acht in $3D$), F_i zijn de middelpunten van 8 Cv's in één grof Cv met C als het middelpunt, r_C en r_{F_i} zijn de afstanden van het oorsprong tot het middelpunt C respectievelijk de middelpunten F_i.

Het bepalen van de gradient

De gradient van ϕ in het midden C van een Cv is met behulp van Gauss theorema bepaald, namelijk

$$\int_V grad\phi dV = \int_S \phi dS \tag{9.18}$$

Uit de vergelijking (9.18) volgt:

$$(grad\phi)_C \approx \frac{1}{\Delta V} \sum_j \phi_j S_j \tag{9.19}$$

waarbij ΔV is het volume van een controle volume en de index j loopt over de zijvlakken e(ast), w(est), n(orth), s(outh), b(ottom) en t(op).
Om het volume te berekenen, nemen we aan dat de controle volume de vorm heeft van een hexaedra, die gedefinieerd is door de coordinaten van de hoekpunten. De hexaedra wordt in zes tedraedra's verdeeld. Het volume van een hexaedra wordt bepaald door de som van volumes van de zes tedraedra's. Bij het interpoleren van de variabelen van het fijn grid naar het grof zijn de waarden van ϕ op het fijn grid bekend, dus $\phi_{if,jf,kf}$, $\phi_{if-1,jf,kf}$, $\phi_{if,jf-1,kf}$, $\phi_{if-1,jf-1,kf}$, $\phi_{if,jf,kf-1}$, $\phi_{if-1,jf,kf-1}$, $\phi_{if,jf-1,kf-1}$ en $\phi_{if-1,jf-1,kf-1}$ en het is de bedoeling om de waarden van ϕ op het grove grid bepalen, dus $\phi_{ig,jg,kg}$, zie figuur 9.2
We schrijven de restrictie operator (9.17) voor de snelheid u voor een Cv uit.

$$FIF1 = \phi^f(3,3,3) + gradfi^f(3,3,3)(centre^g(2,2,2) - centre^f(3,3,3))$$

$$FIF2 = \phi^f(2,3,3) + gradfi^f(2,3,3)(centre^g(2,2,2) - centre^f(2,3,3))$$

$$FIF3 = \phi^f(2,3,2) + gradfi^f(2,3,2)(centre^g(2,2,2) - centre^f(2,3,2))$$

$$FIF4 = \phi^f(3,3,2) + gradfi^f(3,3,2)(centre^g(2,2,2) - centre^f(3,3,2))$$

$$FIF5 = \phi^f(3,2,3) + gradfi^f(3,2,3)(centre^g(2,2,2) - centre^f(3,2,3))$$

$$FIF6 = \phi^f(2,2,3) + gradfi^f(2,2,3)(centre^g(2,2,2) - centre^f(2,2,3))$$

$$FIF7 = \phi^f(3,2,2) + gradfi^f(3,2,2)(centre^g(2,2,2) - centre^f(3,2,2))$$

$$FIF8 = \phi^f(2,2,2) + gradfi^f(2,2,2)(centre^g(2,2,2) - centre^f(2,2,2))$$

$$u^g_{2,2,2} = \frac{1}{8}(FIF1 + FIF2 + FIF3 + FIF4 + FIF5 + FIF6 + FIF7 + FIF8)$$

waarbij $centre^g(.,.,.)$ en $centre^f(.,.,.)$ het middelpunt van het grvoe Cv respectievelijk de middelpunten van het fijne Cv's.
Op analoge wijze kunnen we de snelheden $v^g_{2,2,2}$ en $w^g_{2,2,2}$ op het grove grid bepalen. De variabele ϕ^g voor het hele rekendomein van $NICV \times NJVC \times NKCV$ waarbij $NICV$ $NJCV$ en $NKCV$ het aantal CV's in x-richting, y-richting en z-richting wordt in X-stream met **Subroutine Restrictie(fijngrid, variabele)** bepaalt. De implementatie van deze subroutine in X-stream gaat als volgt:

```
SUBROUTINE Restrictie(fijngrid,variabele)
(1) USE Modules
(2) Declaraties

(3) CALL gradfi(fijngrid,variabele)

(5) DO nbl = 1, nblocks
(6)  figrof =>field_grof(fijngrid-1,nbl)%s(variabele)%fld
(7)  fifijn =>field(fijngrid,nbl)%s(variabele)%fld
```

```
(8)    DO kf = 2, nnkm,2
(9)     DO jf = 2, nnjm,2
(10)     DO if = 2, nnim,2
(11)     ig = if/2+1
(12)     jg = jf/2+1
(13)     kg = kf/2+1
(14)        fistart=0.
(15)      DO k2 = kf, kf+1
(16)       DO j2 = jf, jf+1
(17)        DO i2 = if, if+1
(18)        dist=geom(fijngrid-1,nbl)%centre(ig,jg,kg)- &
                 geom(fijngrid,nbl)%centre(i2,j2,k2)
(19)        fistart = fistart+fifin(i2,j2,k2)+  &
                      (coeff(nbl)%grad(i2,j2,k2,1)*dist)
(20)        END DO
(21)       END DO
(22)      END DO
(23)         ficoa(i,j,k)=(1/8.)*fistart
(24)     END DO
(25)    END DO
(26)   END DO
(27) END DO

END SUBROUTINE Restrictie
```

waarbij :
- nblocks : het aantal blokken,
- fifijn en figrof zijn de variabele op het fijn grid respectievelijk grove grid,
- nkm : het aantal gridpunten in z-richting,
- njm :het aantal gridpunten in y-richting,
- nim :het aantal gridpunten in x-richting,
- centre(ig,jg,kg): het middelpunt van Cv in het grove grid (fijngrid-1) met index (ig, jg, kg),
- centre(i2,j2,k2): het middelpunt van Cv in het fijn grid (fijngrid) met index $(i2, j2, k2)$,
- grad(i2,j2,k2) : het gradient van ϕ in het $centre(i2, j2, k2)$,

Subroutine Restrictie wordt aangeroepen met het gridlevel en de variabele, die geinterpoleerd moet worden. In regel (3) worden de gradienten in het rekendomein 'fijngrid' berekend. In regel (11,12,13) wordt de index (ig, jg, kg) bepaald en vervolgens in regel (15,16,17) worden de indices van de acht Cv's die het grove Cv omsluiten bepaald.
Hier hebben we alleen de waarden van de variabelen op de interne punten bepaald, dus geen randen. Het bepalen van de randen gaat op analoge wijze.

9.1.2 Prolongatie

Een andere operator binnen multigrid procedure is de prolongatie. Prolongatie hebben we nodig om de correctie voor de snelheden δu, δv, δw en de druk δp van het grove grid naar het fijne grid te interpoleren. Hiervoor gebruiken we de volgende operator:

$$\phi'_{F_i} = \phi'_C + (grad\phi')_C \cdot (r_{F_i} - r_C) \tag{9.20}$$

waarbij ϕ' de correctie is. De indices op het fijn grid bepalen we als volgt:

$$if = 2 * ig - 1$$
$$jf = 2 * jg - 1$$
$$kf = 2 * kg - 1$$

De variabele ϕ^f voor het hele rekendomein van $NICV \times NJVC \times NKCV$ waarbij $NICV$ $NJCV$ en $NKCV$ het aantal CV's in x-richting, y-richting en z-richting wordt in X-stream met **Subroutine Prolongatie(fijngrid, variabele)** bepaald. De implementatie van deze subroutine in X-stream gaat als volgt:

```
SUBROUTINE Prolongatie(grofgrid,nphi)

(1) USE Modules
(2) Declaraties
   figrof => field(grofgrid,nbl) %s(nphi) %fld
   ficorrectie => field_cor(grofgrid+1,nbl)%s(nphi) %fld

(4) IF (nphi< nvp) THEN
(5)  DO nbl = 1, nblocks(myproc)
(6)   ficorrectie = ficorrectie- figrof
(8)   field(grofgrid,nbl)%s(nphi)%fld = ficorrectie
(9)  END DO
(10 END IF

(11) CALL  gradfi(grofgrid,nphi)

(12)  DO nbl = 1, nblocks(myproc)
(13)   ficoa => field(ngrcoa,nbl) %s(nphi) %fld
(14)   fifin => field_cor(ngrcoa+1,nbl)%s(nphi) %fld
(15)    DO k = 2, nkm-1
(16)     DO j = 2, njm-1
(17)      DO i = 2, nim-1
(18)      if = 2*(i-1)
(19)      jf = 2*(j-1)
(20)      kf = 2*(k-1)
```

```
(21)        DO k2 = kf, kf + 1
(22)         DO j2 = jf, jf + 1
(23)          DO i2 = if, if + 1
(24)          dist=geom(ngrcoa+1,nbl)%centre(i2,j2,k2)- &
(25)              geom(ngrcoa,nbl) %centre(i,j,k)
(26)          fifin(i2,j2,k2) = ficoa(i,j,k) + &
(27)                      (coeff(nbl)%grad(i,j,k,1)*dist)
(28)          END DO
(29)         END DO
(30)        END DO
(31)       END DO
(32)      END DO
(33)     END DO
(34)   END DO

END SUBROUTINE Prolongatie
```

waarbij :
- nblocks : het aantal blokken,
- fifijn en figrof zijn de variabele op het fijn grid respectievelijk grove grid,
- nkm : het aantal gridpunten in z-richting,
- njm :het aantal gridpunten in y-richting,
- nim :het aantal gridpunten in x-richting,
- centre(ig,jg,kg): het middelpunt van Cv in het grove grid (fijngrid-1) met index (ig, jg, kg),
- centre(i2,j2,k2): het middelpunt van Cv in het fijne grid (fijngrid) met index $(i2, j2, k2)$,
- grad(i2,j2,k2) : het gradient van ϕ in het $centre(i2, j2, k2)$,

9.1.3 Implementatie

Het X-stream pakket moet straks een gebruikerspakket worden, dus de gebruiker moet straks de mogelijkheid krijgen om tussen twee oplosmethoden (Multigrid versnelling en PG) te kunnen kiezen. Met deze doelstelling in het achterhoofd hebben we het inbedden van de implementatie code van MGM in X-stream zodanig dat de code van oplosmethode PG intact is gebleven. Bovendien hebben we getracht om de implementatie code van FMG zo compact mogelijk te houden. Het hoofdprogramma in X-stream met multigrid versnelling ziet er als volgt uit:

```
PROGRAM x_stream
  USE MODULES
  DECLARATIES
  CALL parstart
```

```
(1) multigrid=.false.
(2) CALL init

(3) IF (multigrid) THEN
(4)  ngr_fin=1
(5)  lsr =3
(6)  lsi =5
(7)  cycle(1) = 1
(8)  cycle(2) =4
(9)  cycle(3) = 4
   END IF
!====================================================================
!=================== Multigrid ======================================
!====================================================================
(10) IF (multigrid) THEN
(11)  GRIDLOOP1:  DO
(12)   IF (ngr_fin.GT.1) THEN
(13)    DO nphi = 1, nscalar
(14)     IF (field(1,1)%s(nphi)%lcal) CALL coarse2fine(ngr_fin-
1,nphi)
(15)    END DO

(16)    DO nbl = 1, nblocks(myproc)
(17)     CALL newflx(ngr_fin,nbg,geom(ngr_fin,nbg_local)%vertex, &
               geom(ngr_fin,nbg_local)%centre)
(18)     field(ngr_fin,nbl) %s(nvpp) %fld = field(ngr_fin,nbl) %s(nvp) %fld
      END DO
     END IF

!.. Boudary conditions
(19)   CALL set_bc(ngr_fin) !=>   inlets, walls
(20)   CALL set_scalar_outlet_bc(ngr_fin) !=>   outlets

!     calculate normalisation factors
(21)   CALL calc_norm_factors(ngr_fin)
       ls   = 0
(22)   CALL check_convergence(ngr_fin,status)
       ncyc = 1

(23) CYCLELOOP: DO
(24)   losop= .true.
(25)   voegtoe= .false.
(26)   ls =  1
```

```
!.. Boudary conditions
(27)   CALL set_bc(ngr_fin) !=>   inlets, walls
(28)   CALL set_scalar_outlet_bc(ngr_fin) !=>   outlets

(29) LSG_OUTERLOOP: DO
(30)  IF (field(1,1)%s(nvu)%lcal) CALL calcuvw(ngr_fin)
(31)  IF (field(1,1)%s(nvu)%lcal) CALL outbc(ngr_fin)
(32)  IF (field(1,1)%s(nvp)%lcal) CALL calcp(ngr_fin)
(33)  CALL check_convergence(ngr_fin,status)

(34)  SELECT CASE (status)
(35)    CASE (0)
(36)      ls = ls + 1
(37)    CASE (2)
(38)      EXIT LSG_OUTERLOOP
(39)    END SELECT
   END DO LSG_OUTERLOOP

(40)  IF (ngr_fin.GT.1) THEN
(41)   DO kgrid = ngr_fin, 2, -1
(42)     CALL Restr(kgrid)
     END DO
(43)   DO kgrid = 1, ngr_fin-1
(44)     CALL Inter(kgrid)
     END DO
    END IF

(45)  IF (ncyc < cycle(ngr_fin)) THEN
      ncyc = ncyc + 1
    ELSE
      EXIT CYCLELOOP
    END IF
  END DO CYCLELOOP

(46)  CALL output(ngr_fin)

  PRINT *, 'finished level ', ngr_fin
  IF (ngr_fin<ngrid) THEN
    ngr_fin = ngr_fin + 1
   ELSE
    EXIT GRIDLOOP1
  END IF
(47) END DO GRIDLOOP1
```

```
(48)  ELSE                              ! IF MULTIGRID

!============================================================
!=============== GEEN MULTIGRID ============================
!============================================================
!
!             OPLOSMETHODE SIMPLE met PG
!
!============================================================

(49)        CALL parstop('stop','X-stream',0)

END IF
END PROGRAM x_stream
```

Toelichting

Op regel (1) kunnen we met een logische variabel 'multigrid' kiezen tussen FMG en SIMPLE-PG. Op regel (5) en (6) worden met lsr en lsi het aantal SIMPLE iteraties in een V-cycle per gridlevel gespecificeerd. Op regel (7), (8) en (9) worden het aantal V-cycles per gridlevel gespecificeerd. Op regel (14) wordt in het geval dat een gridlevel groter dan 1 is bereikt en we hebben een geconvergeerd oplossing dan wordt deze oplossing met Subroutine coarse2fine(ngr-fin-1,nphi) geextrapoleerd. Op regel (17) wordt met de geinterpoleerde oplossing de convectieve fluxen berekend. Op regel (30), (31) en (32) worden SIMPLE iteraties uitgevoerd. Na elke iteratie wordt in regel (33) convergetie gecontroleerd. Indien het gridlevel groter is dan 1, dan worden op regels (40-44) de multigrid V-cycling gestart. Op regel (45) wordt nagegaan of het maximaal aantal V-cycles op het gridlevel 'ngr-fin' is bereikt. Op regel (48) krijgen we de andere oplosmethode SIMPLE-PG, dus het huidige oplosmethode in X-stream.

9.1.4 Resultaten

De implementatie van FMG in X-stream zullen we aan de hand het zelfde testcase, 3D driven cavity van het vorige hoodfstuk valideren. Met deze testcase zullen we vier oplosmethoden vergelijken. Het gaat om de volgende:

- Het SIMPLE algoritme met een nul startoplossing (SIMPLE-SG),
- Het SIMPLE algoritme met een goede startoplssing (SIMPLE-PG),
- Het multigrid algoritme FAS met V-cycle (MG-V),
- Het Full multigrid algoritme FMG-FAS (FMG).

In tabel 9.1 hebben we het SIMPLE iteraties van SIMPLE-SG, SIMPLE-PG en FMG om de stroming in 3D cavity bij verschillende grids en voor Re=1000 te vinden. Deze resultaten zijn verkregen door een convergentie criterium $\delta = 10^{-3}$. De iteraties van FMG in tabel 9.1 zijn slechts de iteraties op het fijnste grid.

$rekenrooster$	SG	PG	FMG
$8 \times 8 \times 8$	21	19	14
$16 \times 16 \times 16$	35	30	14
$32 \times 32 \times 32$	98	62	13
$64 \times 64 \times 64$	310	131	13

Tabel 9.1: De iteraties van SG-SIMPLE en FMG voor $Re = 1000$

Uit deze tabel blijkt dat FMG een constant aantal iteraties op het fijnste grid nodig heeft om een oplossing op verschillende grid groottes te vinden. Verder blijkt dat op grids van 8^3 en 16^3 controle volumes het aantal iteraties van de oplosstrategien SIMPLE-SG en SIMPLE-PG niet veel van elkaar verschillen, maar naarmate het grid verfijnd wordt is SIMPLE-PG veel beter dan SIMPLE-SG. Dit verschil is vooral te zien bij het grid van 64^3 controle volumes. Het aantal iteraties van SIMPLE-PG verdubbelt ongeveer wanneer het grid een keer verfijnd wordt.
Verder hebben we de rekentijden van de vier oplosmethoden vergeleken. In tabel 9.2 zijn de rekentijden van SIMPLE-SG en FMG nodig om een oplossing voor verschillende rekenroosters te realiseren weergegeven.

$rekenrooster$	SG	FMG
$8 \times 8 \times 8$	1.24	0.59
$16 \times 16 \times 16$	5.90	4.02
$32 \times 32 \times 32$	117.24	39.63
$64 \times 64 \times 64$	2885.04	359.24

Tabel 9.2: De rekentijden van SG-SIMPLE en FMG voor $Re = 1000$ in seconden

Opvallend is dat de rekentijden van FMG veel kleiner zijn dan SIMPLE-SG. Zo heeft FMG slechts 359 seconden nodig om een oplossing te vinden voor een grid van 64^3 controle volumes, terwijl SIMPLE-SG maar liefst 2885 seconden nodig heeft. Dat is een factor acht. In tabel 9.3 zijn de rekentijden van SIMPLE-PG en FMG weergegeven. Ook hier blijkt FMG veel sneller is dan SIMPLE-PG naarmate het grid verfijnd wordt.
In tabel 9.4 hebben we de twee multigrid methoden met elkaar vergeleken. Uit deze tabel blijkt dat de rekentijden van MG-V en FMG niet veel verschillen voor grids 8^3, 16^3 en 32^3. Het verschil is pas zichtbaar voor grid 64^3.
Tot slot hebben we alle resultaten in tabel 9.5 samengevat. Uit deze tabel kunnen we concluderen dat de oplosmethoden FMG de beste is, beter convergentie gedrag

$rekenrooster$	PG	FMG
$8 \times 8 \times 8$	1.73	0.59
$16 \times 16 \times 16$	7.25	4.02
$32 \times 32 \times 32$	84.72	39.63
$64 \times 64 \times 64$	1300.92	359.24

Tabel 9.3: De rekentijden van PG-SIMPLE en FMG voor $Re = 1000$

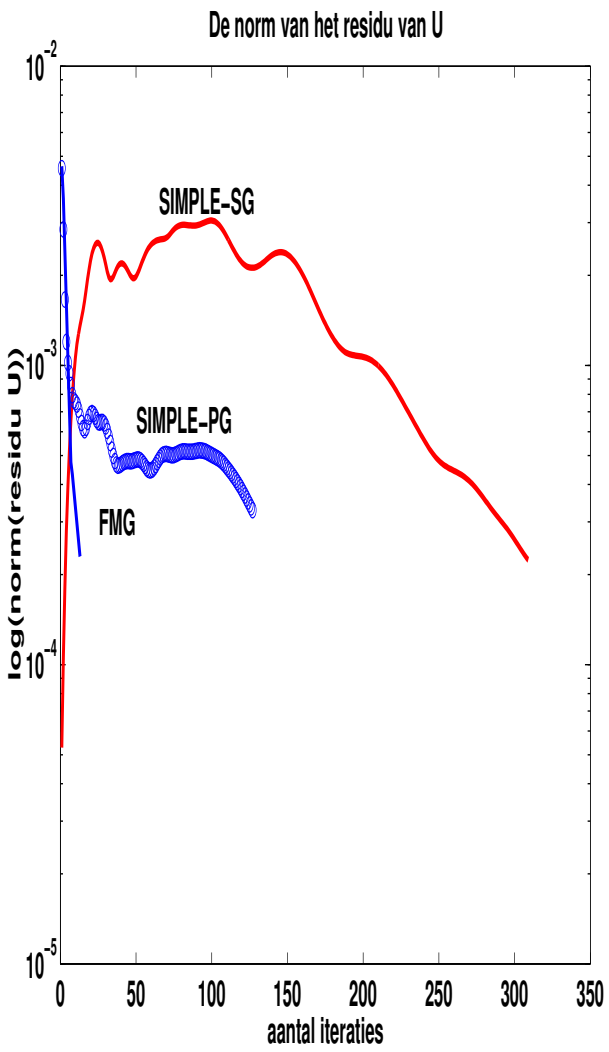
$rekenrooster$	$MG-V$	FMG
$8 \times 8 \times 8$	0.61	0.59
$16 \times 16 \times 16$	4.73	4.02
$32 \times 32 \times 32$	46.65	39.63
$64 \times 64 \times 64$	451.60	359.24

Tabel 9.4: De rekentijden van MG-V en FMG voor $Re = 1000$

dan met FMG kunnen we niet wensen. Deze methode wordt alleen efficient ten opzichte van SIMPLE algoritme naarmate het grid verfijnd wordt.
Verder hebben we het convergentie gedrag van de oplosmethoden SIMPLE-SG, SIMPLE-PG en FMG bekeken. In figuur 9.4 is het convergentie gedrag van de snelheidscomponent u als functie van het aantal SIMPLE iteraties weergegeven.
In figuur 9.5 is het convergentie gedrag van de snelheidscomponent v als functie van het aantal SIMPLE iteraties weergegeven.
In figuur 9.6 is het convergentie gedrag van de snelheidscomponent w als functie van het aantal SIMPLE iteraties weergegeven.
In figuur 9.7 is het convergentie gedrag van de druk p als functie van het aantal SIMPLE iteraties weergegeven.
Deze resulaten zijn verkregen met grid van 64^3 controle volumes, $\delta = 10^{-3}$ en $Re = 1000$. Uit deze figuren is duidelijk te zien dat het convergentie gedrag van FMG als een lineair lijn convergeert naar 10^{-3}.

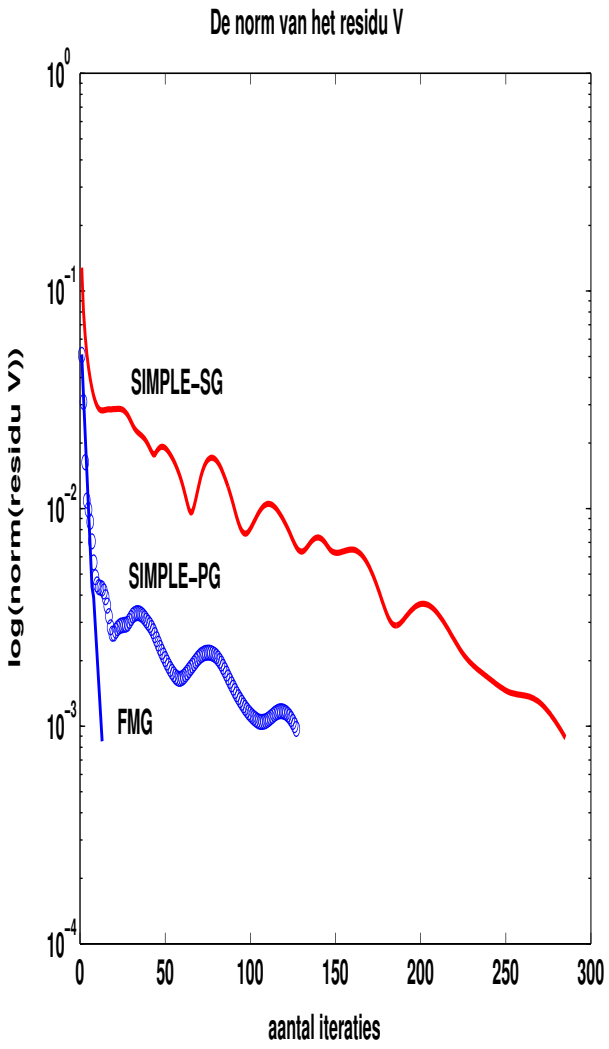
rekenrooster	# SIMPLE			Rekentijd			
	SG	PG	FMG	SG	PG	MG-V	FMG
$8 \times 8 \times 8$	21	19	14	1.24	1.72	0.61	0.59
$16 \times 16 \times 16$	35	30	14	5.90	7.25	4.73	4.02
$32 \times 32 \times 32$	98	62	13	118.0	84.72	46.65	39.63
$64 \times 64 \times 64$	310	131	13	2885.04	1300.92	451.60	359.24

Tabel 9.5: De rekentijd en het aantal iteraties van de vier oplosmethoden voor $Re = 1000$



Figuur 9.4: Het convergentie gedrag van de snelheidscomponent u met SIMPLE-SG, SIMPLE-PG en FMG voor $Re = 1000$

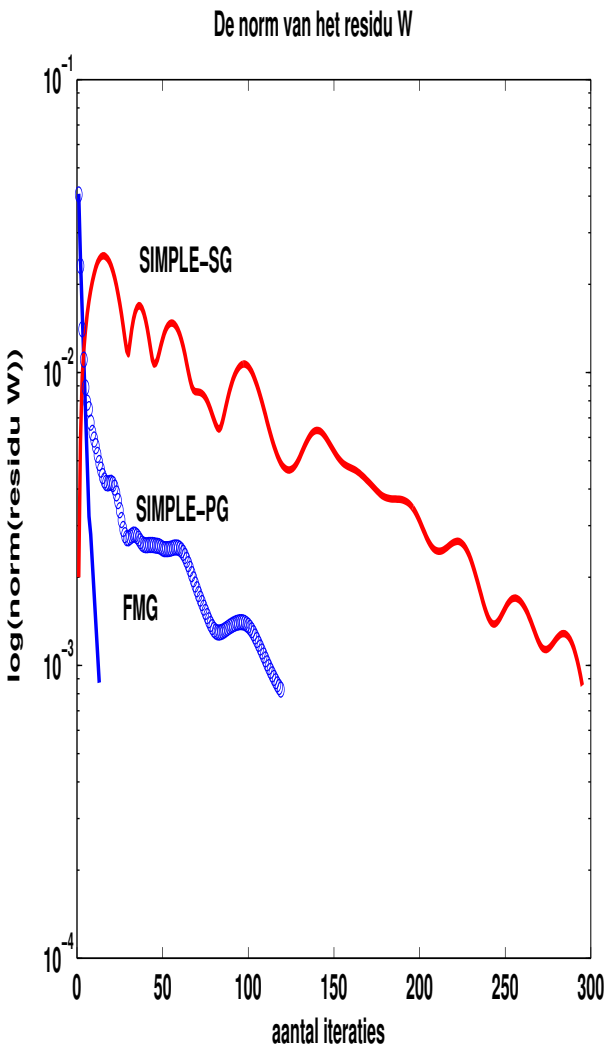
Tot slot hebben we het convergentie criterium $\delta = 10^{-4}$ gekozen. Ook hier hebben we de rekentijden van de oplosmethoden SIMPLE-SG en FMG vergeleken. De resultaten zijn in tabel 9.6 weergegeven. Hieruit blijkt dat het verschil tussen SIMPLE-SG en FMG in rekentijd heel groot is. Voor grid van 64^3 en $Re = 1000$ heeft FMG slechts 570 seconden nodig een oplossing te vinden, terwijl SIMPLE-SG heeft maar liefst 4630 seconden nodig. In figuren 9.8-9.11 is het convergentie gedrag van de snelheden en de druk weergegeven.
Het verkregen resulaten met FMG zijn gevisualiseerd. In figuren zijn de contourplots van de variabelen op dwarsdoorsnede van $x = 0.5$ weergegeven. In figuur



Figuur 9.5: Het convergentie gedrag van de snelheidscomponent v met SIMPLE-SG, SIMPLE-PG en FMG voor $Re = 1000$

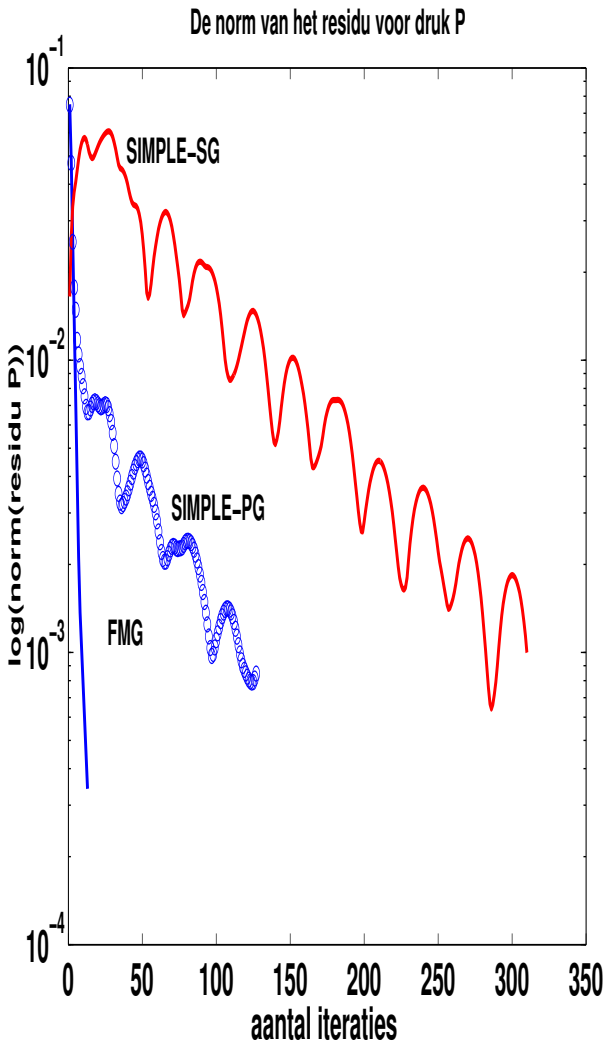
$rekenrooster$	$SIMPLE - SG$	FMG
$8 \times 8 \times 8$	2.05	0.84
$16 \times 16 \times 16$	8.68	5.48
$32 \times 32 \times 32$	176.05	56.00
$64 \times 64 \times 64$	4630.99	570.80

Tabel 9.6: De rekentijd de oplosmethoden SIMPLE-SG en FMG voor $Re = 1000$

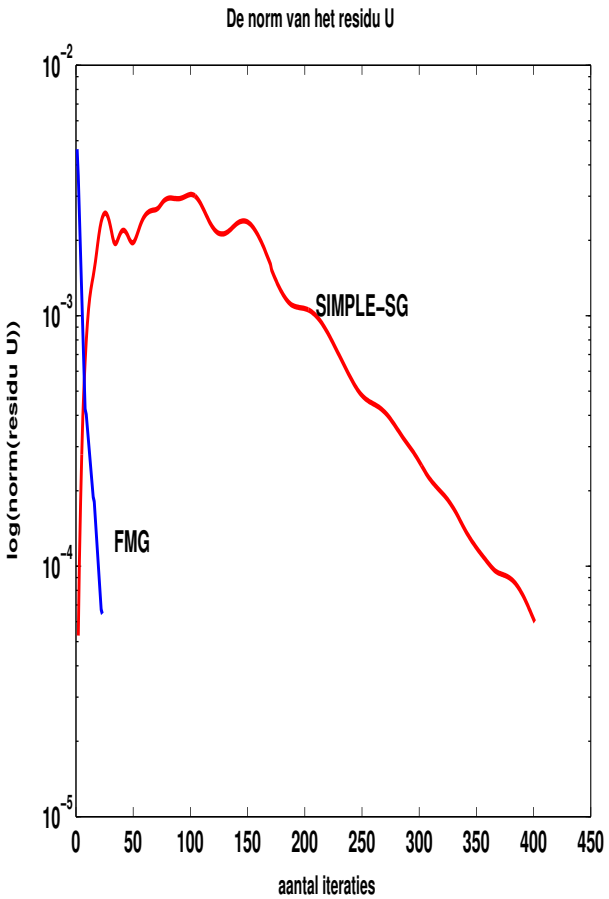


Figuur 9.6: Het convergentie gedrag van de snelheidscomponent w met SIMPLE-SG, SIMPLE-PG en FMG voor $Re = 1000$

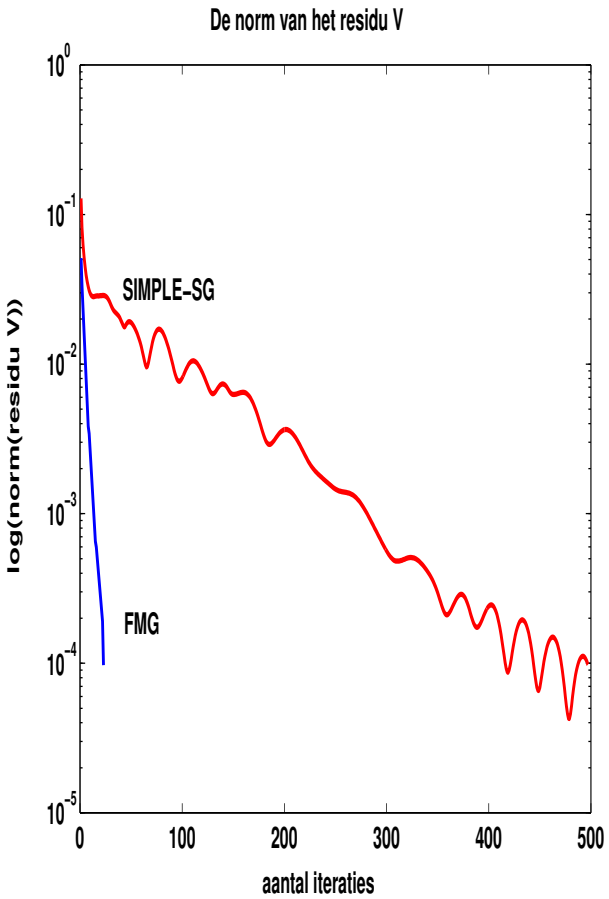
9.12 is de contourplot van de snelheidscomponent v, figuur 9.13 is de contourplot van de snelheidscomponent w, en figuur 9.14 is de contourplot van de druk p. In figuur 9.15 zijn de stroomlijnen op de dwaarsdoorsnede $x = 0.5$ weergegeven.



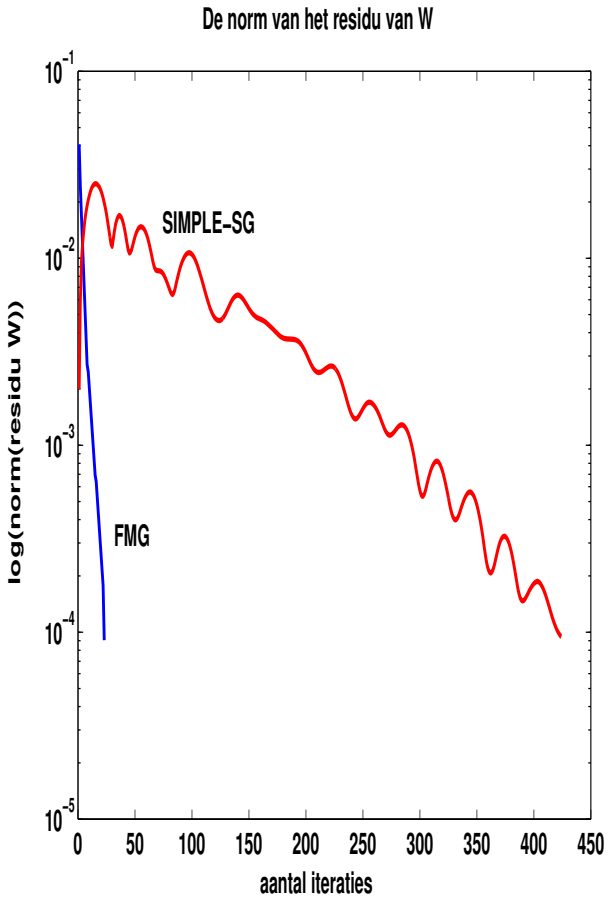
Figuur 9.7: Het convergentie gedrag van de druk p met SIMPLE-SG, SIMPLE-PG en FMG voor $Re = 1000$



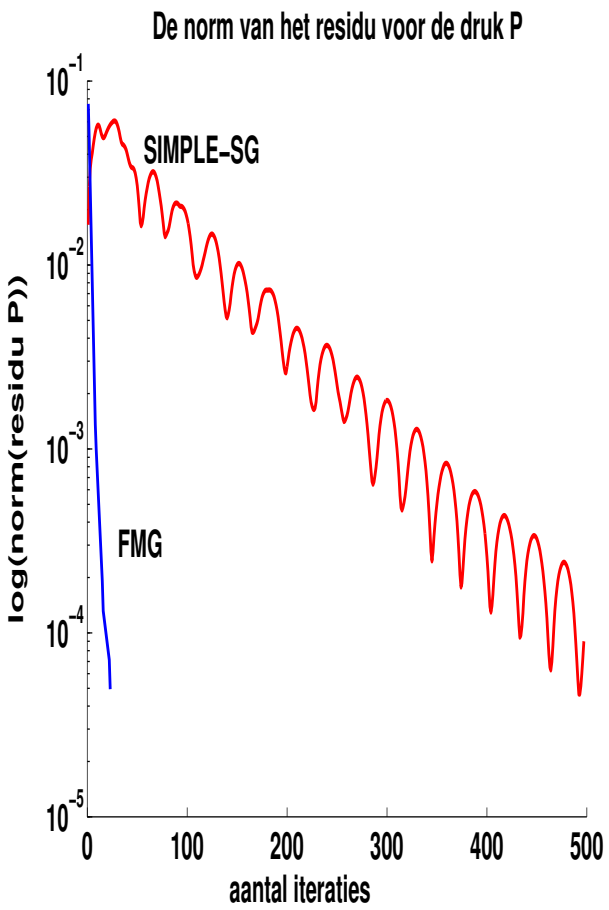
Figuur 9.8: Het convergentie gedrag van de snelheidscomponent u met SIMPLE-SG en FMG voor $Re = 1000$



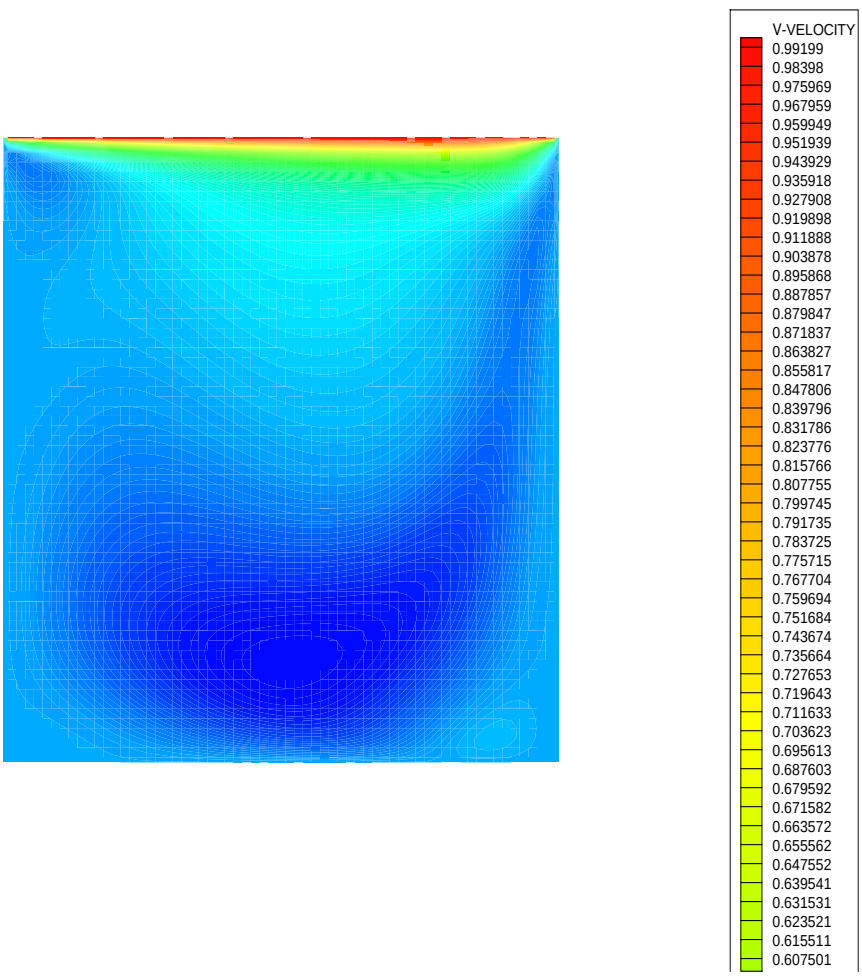
Figuur 9.9: Het convergentie gedrag van de snelheidscomponent v met SIMPLE-SG en FMG voor $Re = 1000$



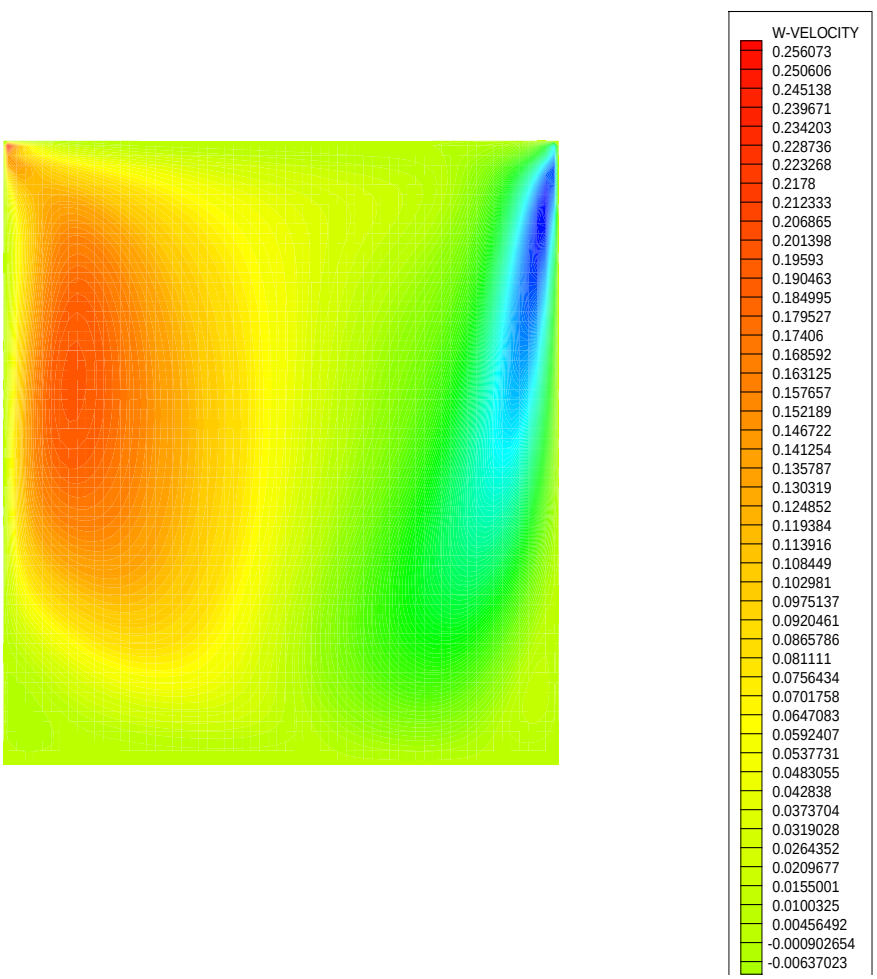
Figuur 9.10: Het convergentie gedrag van de snelheidscomponent w met SIMPLE-SG en FMG voor $Re = 1000$



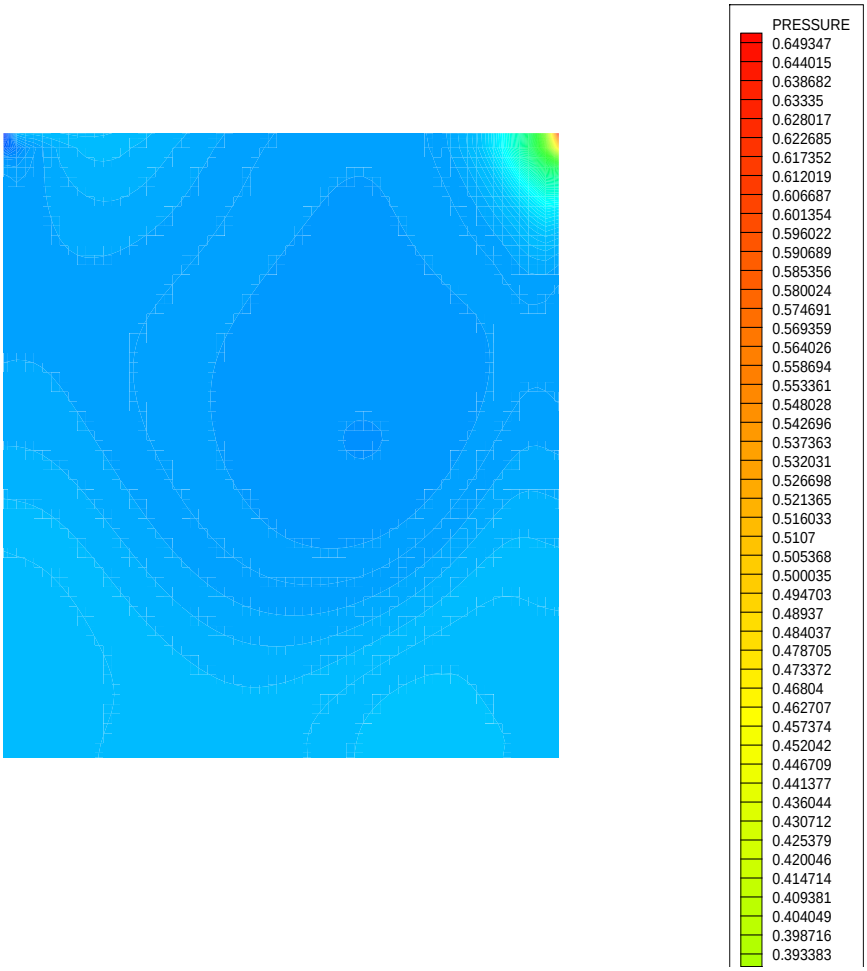
Figuur 9.11: Het convergentie gedrag van de druk p met SIMPLE-SG en FMG voor $Re = 1000$



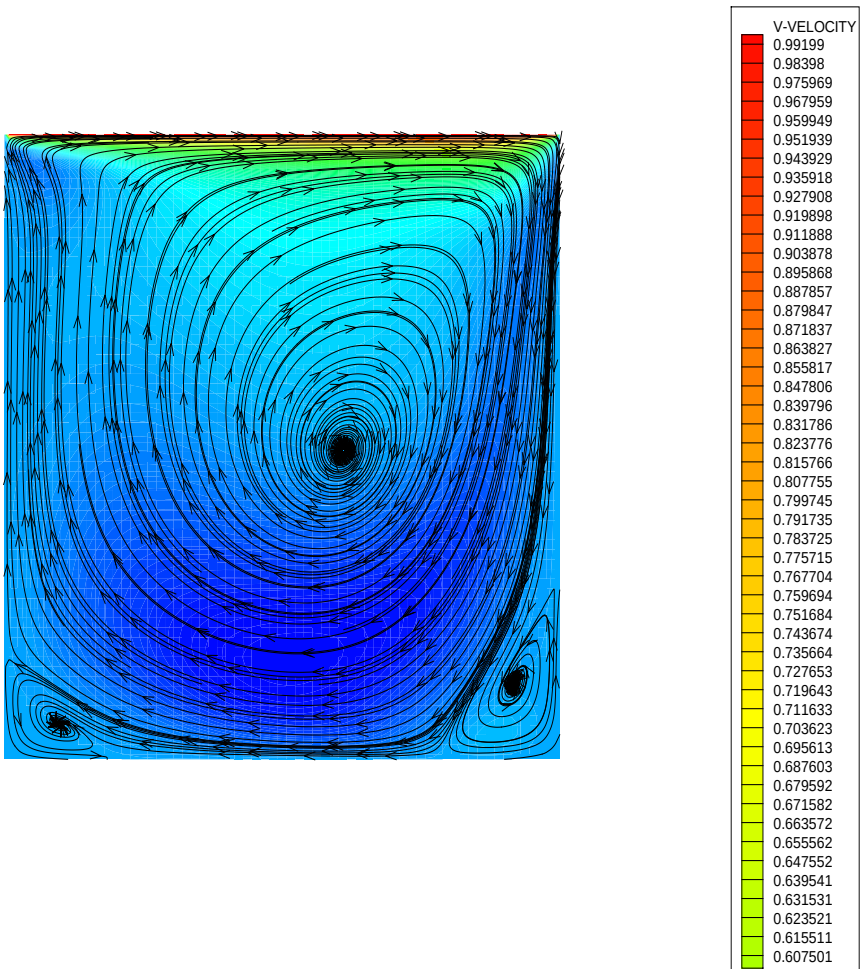
Figuur 9.12: De contour plot van de snelheidscomponent v op $x = 0.5$ en voor $Re = 1000$



Figuur 9.13: De contour plot van de snelheidscomponent w op $x = 0.5$ en voor $Re = 1000$



Figuur 9.14: De contour plot van de druk p op $x = 0.5$ en voor $Re = 1000$



Figuur 9.15: De stroomlijnen op dwarsdoorsnede $x = 0.5$ voor $Re = 1000$

In figuur 9.15 wordt de grote wervel soms primary vortex en de kleine wervels worden secondery vortices genoemd. Het bestaan en het aantal secondary vortices hangt af van het Reynolds getal.

Hoofdstuk 10

Conclusies

In dit verslag zijn de volgende onderwerpen onderzocht:

- In X-stream wordt voor het discretisatie van de convectieve term een nieuwe schema, het zogenaamd deferred correction schema, gebruikt. De voordelen en de nadelen van dit schema zijn onbekend. Om deze reden is dit schema aan de hand van een één-dimensionale convectie-diffusie vergelijking met drie schema's vergeleken. Deze schema's zijn CDS, UDS en Hybride. Het is gebleken dat voor grove grids deferred correctie veel beter is dan de andere schema's. CDS en deferred correctie zijn even nauwkeurig (tweede orde), met een belangrijk verschil dat deferred correctie bij het kiezen van een optimale waarde van β geen numerieke oscillaties genereert. Voor fijne grids verdient CDS de voorkeur. Het grote nadeel van deferred correctie is het kiezen van een optimale waarde β.

- Voor het oplossen van het gelineariseerde stelsel vergelijkingen wordt in X-stream de SIP solver gebruikt. In deze iteratieve methode wordt gebruik gemaakt van een relaxatie parameter. De SIP solver is aan de hand van een twee-dimensionale Laplace vergelijking getest. Het is gebeleken dat de relaxatie parameter α een grote invloed heeft op het aantal iteraties. Een goede keuze van α reduceert het aantal iteraties aanzienlijk. De SIP solver doet het veel beter dan de iteratieve methode gebaseerd op de Incompleet Cholesky decompositie.
  Het is gebleken dat de SIP solver veel beter is dan Gauss-Seidel. De SIP solver heeft veel minder iteraties nodig dan Gauss-Seidel en nog belangrijker heeft altijd minder rekentijd nodig om dezelfde problemen op te lossen. Beide solvers zijn met een multigrid methode versneld. De efficientie van multigrid is enorm, vooral op erg verfijnde grid. Het is gebleken dat het aantal iteraties met multigrid onafhankelijk is van het aantal controle volumes en dat maakt deze methode zondermeer een ideale oplosmethode.
  Het is gebleken dat de SIP solver een betere smoother is voor multigrid dan Gauss-Seidel, maar omdat een iteratie met GS goedkoper is dan de SIP sol-

ver merken we aan de hand van de rekentijd geen grote verschil tussen beide solvers in combinatie met multigrid methode.

- In X-stream hebben we twee oplosstrategiëen aan de hand van een drie-dimensionale lid-driven cavity getest. De berekeningen met SIMPLE zijn op het fijnste grid met een nuloplossing gestart (SIMPLE-SG) en met een goede startoplossing (SIMPLE-PG). Het is gebleken dat SIMPLE-PG het aantal SIMPLE iteraties ten opzichte van SIMPLE-SG aanzienlijk reduceert. Het verschil in SIMPLE iteraties blijft groeien naarmate het grid verfijnd wordt.

- De SIMPLE algoritme is in X-stream met een multigrid methode (FMG-FAS) versneld. Hierbij fungeert het SIMPLE algoritme als smoother. De implementatie code van FMG-FAS in X-stream is aan de hand 3D lid driven cavity getest. De resultaten van FMG-FAS zijn met SIMPLE-SG, SIMPLE-PG en MG-V vergeleken. Het is gebleken dat FMG-FAS de rekentijd t.o.v. SIMPLE-SG en SIMPLE-PG aanzienlijk reduceert. FMG-FAS heeft steeds een constant aantal SIMPLE iteraties voor alle grids nodig. Het verschil in rekentijd tussen FMG-FAS, SIMPLE-SG en SIMPLE-PG wordt dan ook steeds groter naarmate het grid verfijnd wordt. Multigrid is zondermeer een ideaal oplosmethode voor de incompressibele vergelijkingen.
  De berekeningen zijn verkregen met $\delta = 10^{-4}$. Om vooralsnog onopgehelderde redenen zijn we er niet in geslaagd om resulaten te krijgen met convergentie criterium die lager is dan 10^{-4}.

Hoofdstuk 11

Appendix A

We beschouwen de volgende één-dimensionale convectie-diffusie vergelijking:

$$\frac{d}{dx}(\rho u\phi) = \frac{d}{dx}(\Gamma\frac{d\phi}{dx}), \tag{11.1}$$

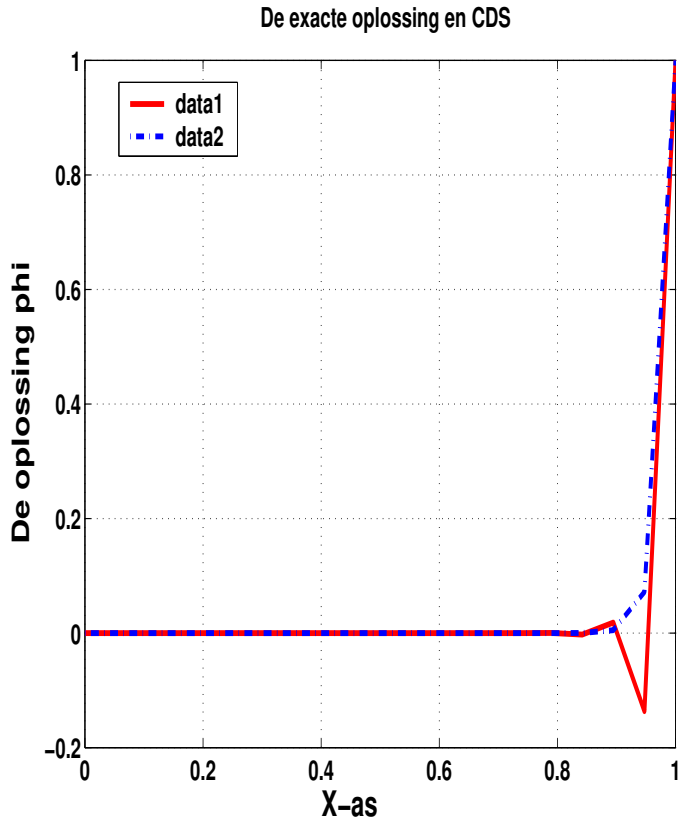
met de volgende Dirichlet randvoorwaarden: $\phi = \phi_0$ op $x = 0$, $\phi = \phi_L$ op $x = L$. Omdat de exacte oplossing van (11.1) eenvoudig te vinden is, wordt dit voorbeeld vaak gebruikt om de discretisatie schema's te testen. Eenvoudig is na te gaan dat de exacte oplossing voor (11.1) te schrijven is als:

$$\phi(x) = \phi_0 + \frac{e^{x\frac{Pe}{L}} - 1}{e^{Pe} - 1}(\phi_L - \phi_0), \tag{11.2}$$

met $Pe = \frac{\rho uL}{\Gamma}$.
De diffusie term in vergelijking (11.1) is met CDS benadert. Voor de convectieve term zijn vier schema's gebruikt, namelijk: CDS, UDS, Hybrid en deferred correctie. Het verkregen stelsel $A\phi = b$ is met een iteratieve methode (Gauss-Seidel) opgelost.
Het is bekend dat de benadering van de convectie term met CDS numerieke oscillaties genereert als Pe groter is dan 2. Om deze reden wordt vaak UDS gebruikt, maar UDS is minder nauwkeurig. Een andere mogelijkheid is het Hybride schema: UDS als Pe groter is dan 2 en CDS als Pe kleiner dan 2.
De benadering van de convectie term met deze schema's en de deferred correctie schema zijn met de exacte oplossing van (11.1) vergeleken. De resultaten worden gegeven voor $N = 20$, $L = 1$, $\phi_0 = 0$, $\phi_L = 1$, $\rho = 1.0$, $\Gamma = 0.02$, $u = 1.0$ en $\delta = 10^{-5}$. Hier is N het aantal roosterpunten.

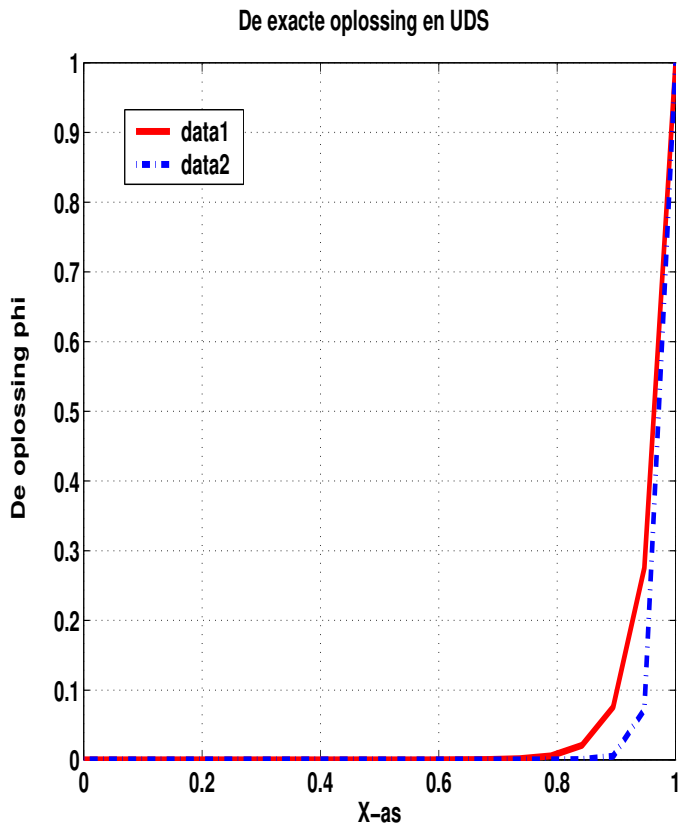
In figuur 10.1 zijn de resultaten van de exacte oplossing en CDS weergegeven.
In figuur 10.2 zijn de resultaten van de exacte oplossing en UDS weergegeven.
In figuur 10.3 zijn de resultaten van de exacte oplossing en deferred correctie voor $\beta = 0.75$ weergegeven.

In figuur 10.4 zijn de resultaten van deferred correctie voor $\beta = 0.75$ en CDS weergegeven.
In figuur 10.5 zijn de resultaten van deferred correctie voor $\beta = 0.75$, UDS en de exacte oplossing weergegeven.



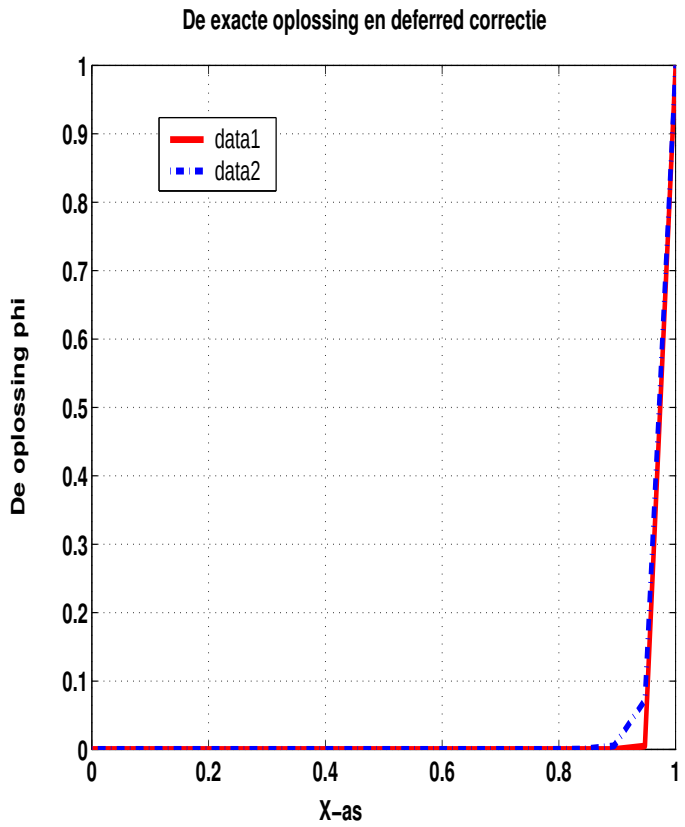
Figuur 11.1: CDS schema en de exacte oplossing voor N=20

We zien duidelijk dat bij de benadering met CDS numerieke oscillaties gegeneerd worden, maar de meeste resultaten liggen dicht bij de exacte oplossing. De grafiek van CDS is met rode kleur weergegeven.



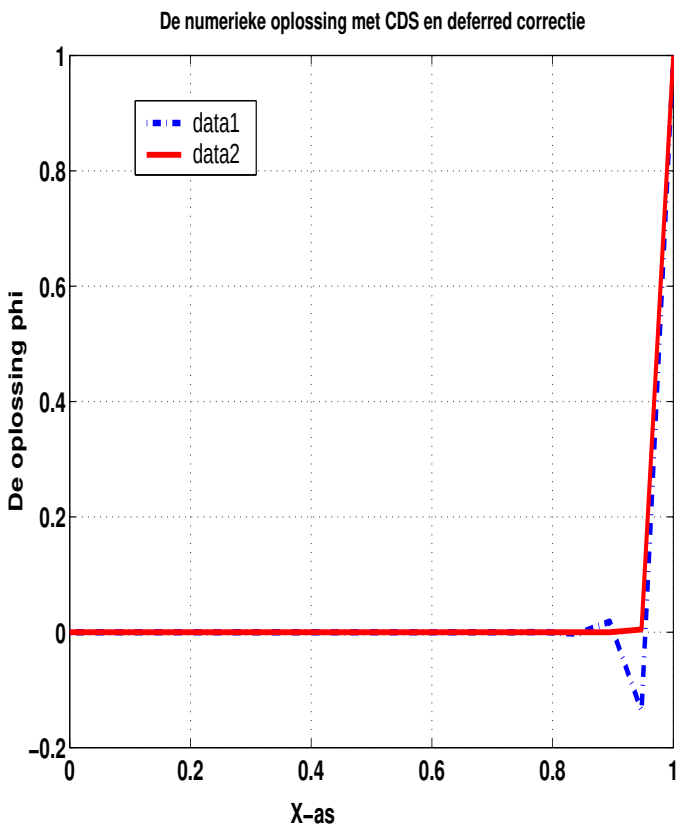
Figuur 11.2: UDS schema en de exacte oplossing voor N=20

We zien dat bij de benadering met UDS geen numerieke oscillaties gegeneerd worden, maar de resultaten vergeleken met de exacte oplossing zijn minder nauwkeurig. De grafiek van UDS is met rode kleur weergegeven.



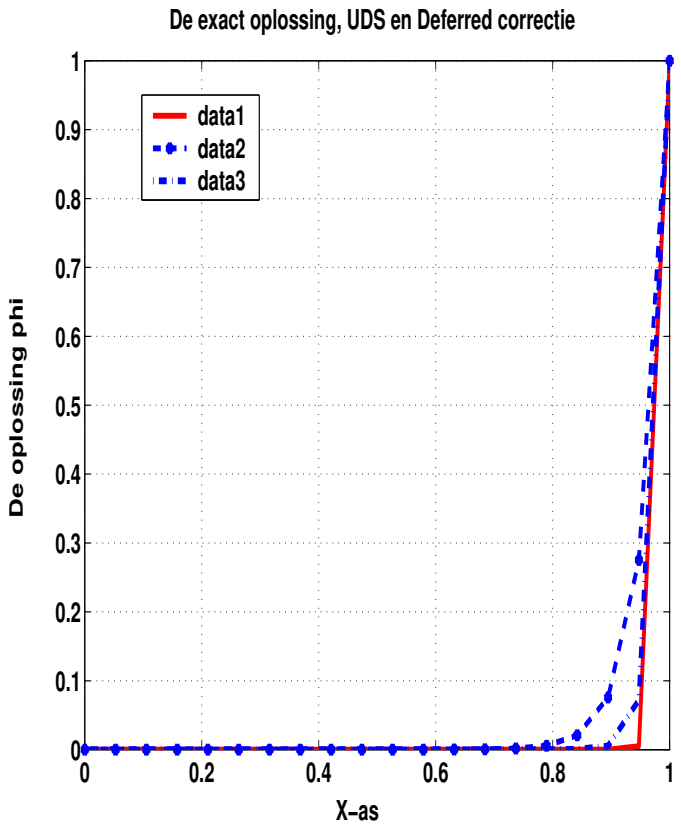
Figuur 11.3: Deferred correctie schema en de exacte oplossing voor N=20

We zien dat bij de benadering met deferred correctie voor $\beta = 0.75$ geen numerieke oscillaties gegeneerd worden en de resultaten zijn bijna identiek aan de exacte oplossing. De grafiek van de exacte oplossing is met blauw kleur weergegeven.



Figuur 11.4: Deferred correctie en CDS schema's voor N=20

We zien duidelijk dat de deferred correctie en CDS schema's bijna dezelfde resultaten geven, behalve dat CDS numerieke oscillaties genereert. De grafiek van CDS is met blauw kleur weergegeven.



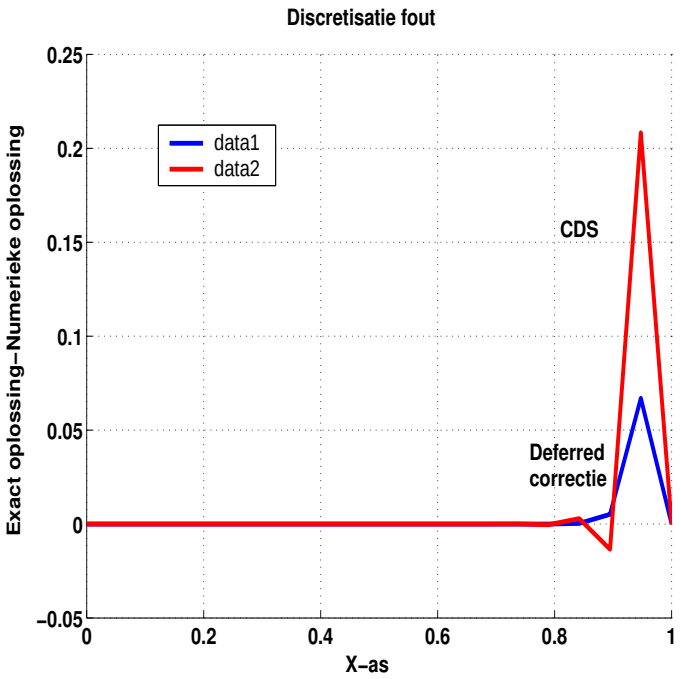
Figuur 11.5: Deferred correctie en UDS schema's met de exacte oplossing voor N=20

Merk op dat de resultaten van deferred correctie ten opzichte van de exacte oplossing veel beter zijn dan die van het UDS schema. In figuur 10.5 is de onderste grafiek van deferred correctie, de grafiek in het midden is van de exacte oplossing en de bovenste grafiek is van UDS.

Verder hebben we voor verschillende schema's het aantal Gauss-Seidel iteraties bekijken. In tabel 10.1 zijn deze iteraties voor grids met gridpunten $N = 10$, $N = 20$, $N = 30$ en $N = 40$ weergegeven. De resultaten met deferred correctie zijn met optimale β verkregen. Uit deze tabel blijkt dat voor grove grids met $N = 10$ en $N = 20$ het deferred correctie schema het minimaal aantal iteraties nodig heeft, terwijl CDS voor $N = 10$ tot divergentie leidt. Voor fijn grids met $N = 30$ en $N = 40$ geeft CDS beter resultaten.

rekenrooster	*UDS*	CDS	*Deferred correctie*
$N = 10$	9	Divergentie	5
$N = 20$	18	16	8
$N = 30$	31	6	18
$N = 40$	47	15	35

Tot slot hebben we in figuur 10.6 het discretisatie fout van CDS (rode kleur) en deferred correctie voor $N = 20$ bekijken. In deze figuur kunnen we zien dat de fout van CDS in de beurt van $x = 1$ veel groter is dan die van deferred correctie.



Figuur 11.6: Discretisatie fout van CDS en Deferred correctie voor N=20

Conclusie

Aan de hand van deze test kunnen we concluderen dat deferred correctie schema nauwkeurig is dan UDS en even goed als CDS met een belangrijke verschil dat deferred correctie schema geen numerieke oscillaties genereert. Het blijkt dat voor kleine N dan $N = 20$ het parameter β kleiner gekozen moet worden dan $\beta = 0.75$. De vraag die nog beantwoord moet worden, is hoe kunnen we deze β kiezen om oscillaties in de oplossing te voorkomen.
Voor deze test zijn de resultaten met het hybride schema identiek aan de resultaten verkregen met UDS.

Hoofdstuk 12

Appendix B

Implementatie

In dit hoofdstuk zullen we de implementatiecode beschrijven. In deze programma is het mogelijk om zowel GS als MGM te gebruiken. De belangrijke opreraties van MGM (smoothing, prolongatie en restrictie) zijn in verschillende subroutines verwerkt zodat deze code makkelijk aan te passen is. Hieronder zullen we elke subroutine van commentaar voorzien. Om de code makkelijk te begrijpen, definieren we alle begrippen, die in dit programma zijn gebruikt.

```
1. CVX, CVY : het aantal controle volumes
      in x-richting en y-richting,
2. AN, AS, AW, AE, AP en b_p : de matrix coefficienten,
3. MAX_GRID : het fijnste grid,
4. MAX_CYCLE : het maximaal aantal multigrid cycles,
5. DX, DY : de stapgrootte in x en y-richting,
6. L : de index van de een-dimensionale array,
7. CORR : de correctie,
8. MAX_RES, RES_MIN : maximale en minimale residu,
9. MGM : een index om een keuze te maken tussen GS en mul-
tigrid,
10. NULDE_RES : het residu na de eerste iteratie, deze is
nodig voor het normaliseren.

!_____________________________________________________
  MODULE TYPE_MODULE
!_____________________________________________________
!            Multigrid module
! In deze module zijn alle globale variabelen gedefinieerd.
!_____________________________________________________
!_____________________________________________________
```

```
IMPLICIT NONE
REAL  :: MAX_RES, RESIDU, MIN_RES, NULDE_RES
REAL(kind=kind(0.0d0)), ALLOCATABLE:: AE(:),AW(:),AN(:),AS(:), &
    AP(:),b_p(:),PHI(:),X(:),Y(:),XCE(:),YCE(:),EXACT_OPL(:)
INTEGER, ALLOCATABLE :: INDGRX(:),INDGRY(:),INDGRXY(:)
INTEGER, ALLOCATABLE :: NITER(:),INDI(:),INDJ(:)
REAL  ::  XMAX, YMAX
INTEGER ::  MAX_GRID, MAX_CYCLE
INTEGER  :: MGM, CVX, CVY, ITER
CHARACTER(LEN=20):: FILE_DATA, FILE_UIT

!_____________________________________________________________
  END MODULE TYPE_MODULE
!_____________________________________________________________
```

Een belangrijke nieuwe onderdeel binnen het programmertaal FORTRAN 90 ten opzichte van FORTRAN 77 is het gebruik van MODULES. Deze modules zijn handig om alle globale variabelen, dus variabelen die nodig zijn in het hele programma. Door 'USE MODULE' in een subroutine op te nemen, worden deze variabelen ook in deze subroutine bekend.

```
!_____________________________________________________________
!_____________ Het hoofdprogramma ____________________________
!_____________________________________________________________

PROGRAM TWD_LAPLACE
 USE TYPE_MODULE
 IMPLICIT NONE

INTEGER ::   VCYCLE, NGR, I, J, INPF

INTERFACE
 SUBROUTINE GAUSS(NGR,VCYCLE)
  USE TYPE_MODULE
  INTEGER,INTENT(IN):: NGR, VCYCLE
 END SUBROUTINE GAUSS

 SUBROUTINE RESTRICTIE(NGR)
  USE TYPE_MODULE
  INTEGER,INTENT(IN):: NGR
 END SUBROUTINE RESTRICTIE

 SUBROUTINE PROLONGATIE(NGR)
  USE TYPE_MODULE
```

```
  INTEGER,INTENT(IN):: NGR
 END SUBROUTINE PROLONGATIE

 SUBROUTINE GRID(NGR)
  USE TYPE_MODULE
  INTEGER,INTENT(IN):: NGR
 END SUBROUTINE GRID

 SUBROUTINE INIT(NGR)
  USE TYPE_MODULE
  INTEGER,INTENT(IN):: NGR
 END SUBROUTINE INIT

 SUBROUTINE CALCOEF(NGR)
  USE TYPE_MODULE
  INTEGER,INTENT(IN):: NGR
 END SUBROUTINE CALCOEF

END INTERFACE

!_____________________________________________________________
!    Initialisatie, inputfile, geheugen reserveren.
!_____________________________________________________________

 ITER=0
 CALL INPUT_DATA
 CALL ALLOCATE

 AW = 0.
 AE = 0.
 AN = 0.
 AS = 0.
 AP = 0.
 b_p  = 0.
 PHI = 0.

!_____________________________________________________________
!           Multigrid    ===> MGM = 2
!           Gauss-Seidel ===> MGM = 1
!           MAX_GRID        ===> fijnste rekenrooster
!_____________________________________________________________

 IF (MGM==2) THEN
  DO NGR=1, MAX_GRID
```

```
   CALL GRID(NGR)
   CALL INIT(MAX_GRID)
   CALL CALCOEF(NGR)
  END DO
 ELSE
  CALL GRID(MAX_GRID)
  CALL CALCOEF(MAX_GRID)
  CALL INIT(MAX_GRID)
 END IF

  CALL GAUSS(MAX_GRID,1)
!_____________________________________________________________
! MAX-CYCLE ===> maximaal aantal cycles
! Hier wordt V-cycles van multigrid methode uitgevoerd
!_____________________________________________________________

 IF (MGM==2) THEN
  DO  VCYCLE= 1, MAX_CYCLE
   DO NGR=MAX_GRID, 2, -1
    CALL RESTRICTIE(NGR)
    CALL GAUSS(NGR-1,VCYCLE)
   END DO

   DO NGR =1, MAX_GRID -1
    CALL PROLONGATIE(NGR)
    CALL GAUSS(NGR+1,VCYCLE)
   END DO
  END DO
 END IF

END PROGRAM TWD_LAPLACE

!_____________________________________________________________
!______ Einde hoofdprogramma _________________________________
!_____________________________________________________________
```

In het hoofdprogramma maken we onderscheid tussen de multigrid solver en Gauss-Seidel. In het geval van GS (MGM=1) bepalen we de coefficienten op het fijnste grid $(MAXGRID)$ en in het geval multigrid (MGM=2) starten we de multigrid cycles (VCYCLE). Voor elk cycle wordt subroutine RESTRICTIE van het fijnste grid tot het grofste aangeroepen en van het grofste grid tot het fijnste grid wordt prolongatie aangeroepen.

```
!_____________________________________________________________
```

```
!__________ Subroutines _____________________________________
!_____________________________________________________________

!=============================================================
 SUBROUTINE GRID(NGR)
!=============================================================
!  In deze subroutine wordt het rekenrooster NGR gemaakt.
!_____________________________________________________________

USE TYPE_MODULE
 INTEGER, INTENT(IN)::   NGR

!...Local variabel
 REAL ::  DX, DY
 INTEGER::   I, J

 DX=XMAX/REAL(INDGRX(NGR)-2)
 X(INDI(NGR)+1)=0.
 DO I=2+INDI(NGR),INDI(NGR)+INDGRX(NGR)-1
  X(I)=X(I-1)+DX
 END DO
 X(INDI(NGR)+INDGRX(NGR))=X(INDGRX(NGR)+INDI(NGR)-1)

 DY=YMAX/REAL(INDGRY(NGR)-2)
 Y(1+INDJ(NGR))=0.
 DO J=2+INDJ(NGR),INDGRY(NGR)+INDJ(NGR)-1
  Y(J)=Y(J-1)+DY
 END DO
 Y(INDGRY(NGR)+INDJ(NGR))=Y(INDGRY(NGR)+INDJ(NGR)-1)

 XCE(1+INDI(NGR))=X(1+INDI(NGR))
 XCE(INDGRX(NGR)+INDI(NGR))=X(INDGRX(NGR)+INDI(NGR)-1)
 DO I=2+INDI(NGR),INDGRX(NGR)+INDI(NGR)-1
  XCE(I)=0.5*(X(I)+X(I-1))
 END DO

 YCE(1+INDJ(NGR))=Y(1+INDJ(NGR))
 YCE(INDGRY(NGR)+INDJ(NGR))=Y(INDGRY(NGR)+INDJ(NGR)-1)
 DO J=2+INDJ(NGR),INDGRY(NGR)+INDJ(NGR)-1
  YCE(J)=0.5*(Y(J)+Y(J-1))
 END DO

!=============================================
 END SUBROUTINE GRID
```

```
!================================================
```

In subroutine GRID worden de coordinaten van de gridpunten van het rekenrooster bepaald, vervolgens worden de coordinaten van de controle volumes bepaald.

```
!================================================
 SUBROUTINE CHECKCONV(ITERATIE,STATUS)
!================================================
! In deze subroutine wordt de convergentie gecontroleerd
!_______________________________________________________
USE TYPE_MODULE
INTEGER, INTENT(IN) :: ITERATIE
INTEGER, INTENT(OUT) :: STATUS

!...Local variabel
 REAL ::  SOM_RES, RESI
 INTEGER :: I, J, L

STATUS=0
SOM_RES=0.
RESI=0.

DO I=2,INDGRX(MAX_GRID)-1
 DO J=2,INDGRY(MAX_GRID)-1
  L=(I-1)*INDGRY(MAX_GRID)+J+ INDGRXY(MAX_GRID)
  RESI=b_p(L)-AP(L)*PHI(L)-AE(L)*PHI(L+INDGRY(MAX_GRID))-
&
  AW(L)*PHI(L-INDGRY(MAX_GRID))-AS(L)*PHI(L-1)-AN(L)*PHI(L+1)
   SOM_RES=SOM_RES+ABS(RESI)
  END DO
 END DO

 IF(ITER.EQ.0) NULDE_RES=SOM_RES
 ITER =ITER +1
 RESIDU=SOM_RES/NULDE_RES

IF (RESIDU.GE.MAX_RES) THEN
   STATUS=1
 ELSE IF (RESIDU.GE.MIN_RES.AND. &
     & (ITERATIE.EQ.NITER(MAX_GRID)))  THEN
   STATUS=2
 ELSE  IF(RESIDU.LE.MIN_RES) THEN
   STATUS=3
END IF
```

```
!=============================================
 END SUBROUTINE CHECKCONV
!=============================================
```

In subroutine CHECKCONV wordt de convergentie gecontroleerd. Deze subroutine roepen we voor zowel GS als MGM alleen op het fijnste grid $(MAX-GRID)$ aan. Hiervoor gebruiken STATUS :
- STATUS = 1 : Divergentie, het residu is groter dan een hele grote getal $(MAXRES)$,
- STATUS = 2 : Het maximale aantal iteraties $(NITER(MAXGRID)$ is bereikt en we hebben nog geen convergetie bereikt,
- STATUS = 3 : Convergetie, het residu is kleiner dan een toelaatbare kleine getal $(MINRES)$.

```
!=============================================
 SUBROUTINE GAUSS(NGR,VCYCLE)
!=============================================
! In deze subroutine wordt er een aantal iteraties
! uitgevoerd en slechts op het fijnste grid wordt
! er convergentie check gedaan.
!_____________________________________________________
USE TYPE_MODULE
IMPLICIT NONE
INTEGER, INTENT(IN) :: NGR,VCYCLE

INTERFACE

 SUBROUTINE CHECKCONV(ITERATIE,STATUS)
  INTEGER,INTENT(IN):: ITERATIE
  INTEGER,INTENT(OUT):: STATUS
 END SUBROUTINE CHECKCONV

END INTERFACE

!...Local variabel
 INTEGER:: ITERATIE,I,L,J,STATUS,INPF,LI
 REAL :: RES

 RES=0.
 DO ITERATIE=1, NITER(NGR)
  DO I=2,INDGRX(NGR)-1
   LI=(I-1)*INDGRY(NGR)
    DO J=2, INDGRY(NGR)-1
    L= LI+INDGRXY(NGR)+J
```

```
    RES=b_p(L)-AP(L)*PHI(L)-AE(L)*PHI(L+INDGRY(NGR))-    &
  & AW(L)*PHI(L-INDGRY(NGR))-AS(L)*PHI(L-1)-AN(L)*PHI(L+1)
    PHI(L)=PHI(L)+RES/AP(L)
   END DO
  END DO

 IF(NGR.EQ.MAX_GRID)  THEN
  CALL CHECKCONV(ITERATIE,STATUS)
  SELECT CASE (STATUS)
   CASE(1)
     WRITE(*,*) 'DIVERGENTIE'
     STOP
   CASE(2)
    IF(VCYCLE==MAX_CYCLE.OR.MGM==1) THEN
     WRITE(*,*) 'CONVERGENTIE IS NIET VOLDAAN'
     WRITE(*,*) ' geef de naam van uitputfile'
     READ (*,*) FILE_UIT
     OPEN(UNIT=2, FILE=FILE_UIT)
     DO I =2, INDGRX(MAX_GRID)-1
      DO J=2, INDGRY(MAX_GRID)-1
      INPF= (i-1)*INDGRY(MAX_GRID)+j+INDGRXY(MAX_GRID)
      WRITE(2,*) ITERATIE, PHI(INPF),EXACT_OPL(INPF), INPF, VCYCLE
      END DO
     END DO
      CLOSE(2)
    END IF
   CASE(3)
     WRITE(*,*) 'CONVERGENTIE IS VOLDAAN'
     WRITE(*,*) ' geef de naam van uitputfile'
     READ (*,*) FILE_UIT
     OPEN(UNIT=2, FILE=FILE_UIT)
     WRITE(2,*) STATUS,'STATUS'
     DO I =2, INDGRX(MAX_GRID)-1
      DO J=2, INDGRY(MAX_GRID)-1
      INPF= (I-1)*INDGRY(MAX_GRID)+J+INDGRXY(MAX_GRID)
      WRITE(2,*) ITERATIE, PHI(INPF),EXACT_OPL(INPF),INPF, VCYCLE
      END DO
     END DO
      CLOSE(2)
      STOP
  END SELECT
 END IF
END DO
```

```
!=========================================
 END SUBROUTINE GAUSS
!=========================================
```

In subroutine GAUSS wordt multigrid smoothig gedaan. Na elke iteratie (sweep) wordt met de nieuwe oplossing het residu bepaald, waarbij de convergentie controle met subroutine CHECKCONV alleen op het fijnste grid wordt gedaan die afhankelijk is van het STATUS.

```
!=========================================
 SUBROUTINE INIT(NGR)
!=========================================
! In deze subroutine wordt de exacte oplossing bepaald en
! en de randvoorwaarden voor de numerieke oplossing.
!_________________________________________________
USE TYPE_MODULE
INTEGER,INTENT(IN):: NGR

!...Local variabel
 INTEGER:: I,J,L,LI

 DO I=1,INDGRX(NGR)
  DO J=1,INDGRY(NGR)
   LI=(I-1)*INDGRY(NGR)+INDGRXY(NGR)
   L=LI+J
   EXACT_OPL(L)=SINH(3.14*XCE(I+INDI(NGR)))*SIN(3.14*YCE(J+INDJ(NGR)))
   PHI(L)=0.
  END DO
 END DO

 DO J=1,INDGRY(NGR)
  PHI((INDGRX(NGR)-1)*INDGRY(NGR)+J+INDGRXY(NGR))= &
  &  SINH(3.14)*SIN(3.14*YCE(INDGRY(NGR)+INDJ(NGR))
 END DO

!=========================================
 END SUBROUTINE INIT
!=========================================
```

In subroutine INIT wordt de exacte oplossing van de Laplace vergelijking bepaald en worden de Dirichlet randvoorwaarden verwerkt.

```
!=========================================
 SUBROUTINE CALCOEF(NGR)
```

```
!=============================================
! In deze subroutine wordt de matrix A en het rechterlid
! voor het rekenrooster NGR bepaald.
!_____________________________________________
USE TYPE_MODULE
INTEGER,INTENT(IN):: NGR

!...Local variabel
INTEGER:: I, J, L

DO I=2,INDGRX(NGR)-1
 DO J=2,INDGRY(NGR)-1
  L=(I-1)*INDGRY(NGR)+INDGRXY(NGR)+J
 AE(L)=-(Y(J+INDJ(NGR))-Y(J+INDJ(NGR)-1))/(XCE(I+1+INDI(NGR))&
      & -XCE(I+INDI(NGR)))
 AN(L)=-(X(I+INDI(NGR))-X(I+INDI(NGR)-1))/(YCE(J+INDJ(NGR)+1)&
      & -YCE(J+INDJ(NGR)))
 AW(L)=-(Y(J+INDJ(NGR))-Y(J+INDJ(NGR)-1))/(XCE(I+INDI(NGR)) &
      & -XCE(I+INDI(NGR)-1))
 AS(L)=-(X(I+INDI(NGR))-X(I+INDI(NGR)-1))/(YCE(J+INDJ(NGR))&
      & -YCE(J+INDJ(NGR)-1))
 AP(L)=-(AE(L)+AW(L)+AN(L)+AS(L))
 b_p(L)=0.
 END DO
END DO

!=============================================
 END SUBROUTINE CALCOEF
!=============================================
```

In subroutine CALCOEF worden matrix coefficienten AW, AE, AN, AS, AP en b_p bepaald. Deze coefficienten zijn afhankelijk van het grid (NGR). Tijdens een multigrid cycle hebben we elke grid deze coefficienten nodig.

```
!=============================================
 SUBROUTINE INPUT_DATA
!=============================================
USE TYPE_MODULE

!... Local variabel
 CHARACTER(LEN=20):: FORM
 INTEGER :: I, NGRM

  WRITE(*,*) ' geef de naam van inputfile'
```

```
  READ (*,*) FILE_DATA
  OPEN (UNIT=4,FILE=FILE_DATA)
  READ (4,'(I1)') MGM
  print*, MGM
  READ  (4,*) XMAX, YMAX
  print*, XMAX, YMAX
  READ  (4,*) CVX, CVY
  print*, CVX, CVY
  READ  (4,*) MAX_RES, MIN_RES
  print*, MAX_RES, MIN_RES

  READ  (4,*) MAX_GRID
  ALLOCATE(NITER(MAX_GRID))
  READ  (4,*)   NITER(MAX_GRID)
  print*, MAX_GRID, NITER(MAX_GRID)

   IF(MGM==2) THEN
     NGRM=MAX_GRID-1
     READ (4, *) MAX_CYCLE
     WRITE (*, *) MAX_CYCLE
     FORM='( I2)'
     WRITE(FORM(2:2),'(I1)') NGRM
     READ (4, FORM) (NITER(I),I=1,MAX_GRID-1)
     WRITE (*, FORM) (NITER(I),I=1,MAX_GRID-1)
   END IF
   CLOSE(4)

!=======================================
 END SUBROUTINE INPUT_DATA
!=======================================
```

In subroutine $INPUTDATA$ wordt het lezen van de gegevens uit een inputfile geregeld. Deze inpufile bevat bevat de vogende gegevens:

- aantal cellen (Cv's) in x-richting, respectievelijk y-richting,
- maximaal aantal gridniveau's
- aantal relaxatieslagen van Gauss per gridniveau
- convergentie criterium δ
- maximaal aantal multigrid V-cycles
- maximaal aantal iteraties voor GS
- de keuze voor de solver (GS of MGM)

```
!=========================================
 SUBROUTINE RESTRICTIE(NGR)
!=========================================
USE TYPE_MODULE

!...Local variabel
 INTEGER ,INTENT(IN):: NGR

INTERFACE
 SUBROUTINE RESUDFC(NGR)
  INTEGER,INTENT(IN):: NGR
  END SUBROUTINE RESUDFC
END INTERFACE

 CALL RESUDFC(NGR)

!=========================================
END SUBROUTINE RESTRICTIE
!=========================================
```

In subroutine RESTRICTIE wordt een onderdeel van een multigrid V-cycle gerealiseerd. Binnen deze subroutine wordt subroutine RESUDFC aangeroepen om het residu van het fijne grid naar het grove grid te brengen.

```
! =========================================
 SUBROUTINE VINTGF(NGR)
!=========================================
USE TYPE_MODULE
INTEGER, INTENT(IN):: NGR

!...Local variabel
 INTEGER :: IG,JG,IF,JF,INPG,INPF
 REAL  :: CORR(INDGRXY(NGR+1)+1:INDGRXY(NGR+1)+ &
& INDGRX(NGR+1)*INDGRY(NGR+1))
 REAL :: t,u,v, w

! GEBIED 1

 DO IG=2,INDGRX(NGR)-2
  IF=2*IG-1
  DO JG=2,INDGRY(NGR)-2
   JF=2*JG-1
   INPG=(IG-1)*INDGRY(NGR)+JG+INDGRXY(NGR)
   INPF=(IF-1)*INDGRY(NGR+1)+JF+INDGRXY(NGR+1)
```

```
   t=0.75*PHI(INPG)+0.25*PHI(INPG+1)
   u=0.25*PHI(INPG)+0.75*PHI(INPG+1)
   v=0.75*PHI(INPG+INDGRY(NGR))+0.25*PHI(INPG+INDGRY(NGR)+1)
   w=0.25*PHI(INPG+INDGRY(NGR))+0.75*PHI(INPG+INDGRY(NGR)+1)
   CORR(INPF)=0.75*t+0.25*v
   CORR(INPF+1)=0.75*u+0.25*w
   CORR(INPF+INDGRY(NGR+1))  =0.25*t+0.75*v
   CORR(INPF+INDGRY(NGR+1)+1)=0.25*u+0.75*w
  END DO
 END DO

! GEBIED 2

 DO IG=2,INDGRX(NGR)-2
  INPG=(IG-1)*INDGRY(NGR)+INDGRXY(NGR)+2
  IF=2*IG-1
  INPF=(IF-1)*INDGRY(NGR+1)+INDGRXY(NGR+1)+2
  CORR(INPF)     =0.375*PHI(INPG)+0.125*PHI(INPG+INDGRY(NGR))
  CORR(INPF+INDGRY(NGR+1))=0.375*PHI(INPG+INDGRY(NGR))+0.125*PHI(INPG)
  INPG=(IG-1)*INDGRY(NGR)+INDGRXY(NGR)+INDGRY(NGR)-1
  INPF=(IF-1)*INDGRY(NGR+1)+INDGRXY(NGR+1)+INDGRY(NGR+1)-
1
  CORR(INPF)  =0.375*PHI(INPG)+0.125*PHI(INPG+INDGRY(NGR))
  CORR(INPF+INDGRY(NGR+1))=0.375*PHI(INPG+INDGRY(NGR))+0.125*PHI(INPG)
 END DO

 DO JG=2,INDGRY(NGR)-2
  INPG=INDGRY(NGR)+INDGRXY(NGR)+JG
  JF=2*JG-1
  INPF=INDGRY(NGR+1)+INDGRXY(NGR+1)+JF
  CORR(INPF)   =0.375*PHI(INPG)+0.125*PHI(INPG+1)
  CORR(INPF+1)=0.375*PHI(INPG+1)+0.125*PHI(INPG)
  INPG=(INDGRX(NGR)-2)*INDGRY(NGR)+INDGRXY(NGR)+JG
  INPF=(INDGRX(NGR+1)-2)*INDGRY(NGR+1)+INDGRXY(NGR+1)+JF
  CORR(INPF)   =0.375*PHI(INPG)+0.125*PHI(INPG+1)
  CORR(INPF+1)=0.375*PHI(INPG+1)+0.125*PHI(INPG)
 END DO

! GEBIED 3

 INPF=INDGRXY(NGR+1)+INDGRY(NGR+1)
 INPG=INDGRXY(NGR)+INDGRY(NGR)
 CORR(INPF+2)     =0.25*PHI(INPG+2)
 CORR(INPF+INDGRY(NGR+1)-1)=0.25*PHI(INPG+INDGRY(NGR)-1)
```

```
 INPF=INDGRXY(NGR+1)+(INDGRX(NGR+1)-2)*INDGRY(NGR+1)
 INPG=INDGRXY(NGR)   +(INDGRX(NGR)-2)*INDGRY(NGR)
 CORR(INPF+2)     =0.25*PHI(INPG+2)
 CORR(INPF+INDGRY(NGR+1)-1)=0.25*PHI(INPG+INDGRY(NGR)-1)

!_____________________________________________
!------CORRIGEER DE VARIABELE PHI-------------
!_____________________________________________

 DO IF =2, INDGRX(NGR+1)-1
  DO JF =2, INDGRY(NGR+1)-1
   INPF=(IF-1)*INDGRY(NGR+1)+JF+INDGRXY(NGR+1)
   PHI(INPF)=PHI(INPF)+CORR(INPF)
   END DO
 END DO

!=============================================
 END SUBROUTINE VINTGF
!=============================================
```

In subroutine VINTGF wordt de correctie berekend op grid (NGR) geinterpoleerd met bilineaire interpolatie naar grid (NGR+1) bepaald. Deze subroutine wordt in subroutine PROLONGATIE aangeroepen.

```
!=============================================
 SUBROUTINE PROLONGATIE(NGR)
!=============================================
USE TYPE_MODULE
INTEGER ,INTENT(IN):: NGR

INTERFACE
 SUBROUTINE VINTGF(NGR)
  USE TYPE_MODULE
  INTEGER,INTENT(IN):: NGR
 END SUBROUTINE VINTGF
END INTERFACE

 CALL VINTGF(NGR)

!=============================================
 END SUBROUTINE PROLONGATIE
!=============================================
```

In subroutine PROLONGATIE wordt een onderdeel van multigrid cycle gerealiseerd.

```
!================================================
 SUBROUTINE RESUDFC(NGR)
!================================================
USE TYPE_MODULE
INTEGER,INTENT(IN):: NGR

!...Local variabel
 REAL :: RESU(INDGRXY(NGR)+1:INDGRXY(NGR)+INDGRX(NGR)*INDGRY(NGR))
 INTEGER :: IG,JG,IF,JF,INPG,INPF

 DO IF=2,INDGRX(NGR) -1
  DO JF=2,INDGRY(NGR)-1
   INPF=(IF-1)*INDGRY(NGR)+JF+INDGRXY(NGR)
   RESU(INPF)=-AE(INPF)*PHI(INPF+INDGRY(NGR))-     &
  &            AW(INPF)*PHI(INPF-INDGRY(NGR))-    &
  &            AS(INPF)*PHI(INPF-1)-AN(INPF)*PHI(INPF+1)   +    &
  &            b_p(INPF)-AP(INPF)*PHI(INPF)
  END DO
 END DO

 DO IG=2,INDGRX(NGR-1) -1
  IF=2*IG-2
   DO JG=2,INDGRY(NGR-1) -1
    JF=2*JG-2
    INPG=(IG-1)*INDGRY(NGR-1)+JG+INDGRXY(NGR-1)
    INPF=(IF-1)*INDGRY(NGR)+JF+INDGRXY(NGR)
    b_p(INPG)=RESU(INPF)+RESU(INPF+1)+RESU(INPF+INDGRY(NGR))+    &
   &          RESU(INPF+INDGRY(NGR)+1)
    PHI(INPG)=0.
  END DO
 END DO

!================================================
 END SUBROUTINE RESUDFC
!================================================
```

In subroutine RESUDFC wordt het residu op grid (NGR) berekend, die vervolgens gebruikt wordt om het residu op het grove grid (NGR-1) te berekenen. Deze subroutine wordt in subroutine RESTRICTIE aangeroepen.

```
!================================================
 SUBROUTINE ALLOCATE
!================================================
USE TYPE_MODULE
```

```
ALLOCATE(AW(INDGRXY(MAX_GRID)+INDGRX(MAX_GRID)*INDGRY(MAX_GRID)))
ALLOCATE(AE(INDGRXY(MAX_GRID)+INDGRX(MAX_GRID)*INDGRY(MAX_GRID)))
ALLOCATE(AN(INDGRXY(MAX_GRID)+INDGRX(MAX_GRID)*INDGRY(MAX_GRID)))
ALLOCATE(AS(INDGRXY(MAX_GRID)+INDGRX(MAX_GRID)*INDGRY(MAX_GRID)))
ALLOCATE(AP(INDGRXY(MAX_GRID)+INDGRX(MAX_GRID)*INDGRY(MAX_GRID)))
ALLOCATE(b_p(INDGRXY(MAX_GRID)+INDGRX(MAX_GRID)*INDGRY(MAX_GRID)))
ALLOCATE(PHI(INDGRXY(MAX_GRID)+INDGRX(MAX_GRID)*INDGRY(MAX_GRID)))
ALLOCATE(EXACT_OPL(INDGRXY(MAX_GRID)+INDGRX(MAX_GRID)*INDGRY(MAX_GRID)))
ALLOCATE(X(INDI(MAX_GRID)+INDGRX(MAX_GRID)))
ALLOCATE(XCE(INDI(MAX_GRID)+INDGRX(MAX_GRID)))
ALLOCATE(Y(INDJ(MAX_GRID)+INDGRY(MAX_GRID)))
ALLOCATE(YCE(INDJ(MAX_GRID)+INDGRY(MAX_GRID)))

!================================================
 END SUBROUTINE ALLOCATE
!================================================
!_______________________________________________
!_________ Einde subroutines ___________________
!_______________________________________________
```

In subroutine ALLOCATE wordt het geheugen voor alle één-dimensionale arrays gereserveerd. Dat is nodig om het verspillen van het computergeheugen te voorkomen.

Hoofdstuk 13

Appendix C

In deze paragraaf geven we een voorbeeld van een inputfile van X-stream. Voor de toelichting van de gebruikte parameters in deze inputfile, zie blz. 4 en 5.

```
!-------------------------------------------------------
---------------
! X-stream KEYWORD INPUT FILE
!-------------------------------------------------------
---------------
! This file contains values for all available keywords
! A default file can be regenerated by running FAN3D with the following
! input file:
! PRINTDEFAULTS <filename for default keywords and values>
! The information is printed as follows:
! KEYWORD     value
! For characters the maximum value is 155 characters
! All text after an exclamation mark (!) is ignored
! No distinction is made between uppercase and lowercase keywords.
! Format is completely free; trailing or prepending spaces are ignored.
!-------------------------------------------------------
---------------
MESSAGELEVEL 1
TITLE  cube flow
!-----------------------------------!
! LSG: Number of outer iterations per grid level
!    1   2   3    4
!-----------------------------------!
LSG  200 200 2000 20000
FILOUT  cube.out
FILGRD  cube.grd
FILRES  cube
LREAD   false
```

```
LWRITE  false
LTEST   false
LONGEO  false
KREST   1
NBLMON  1
IMON    3
JMON    3
KMON    3
NBLPR   1
IPR     2
JPR     2
KPR     2
LTIME   false
LTWRIT  false
ITSTEP  1
NOTT    1
DT      1.
GAMT    1.
SORMAX  1.0E-4
SMALL   1.E-15
GREAT   1.E+31
DENSIT  1.0
VISCOS  .1
PRANL   100.0
ALFA    .92
ASC_PRINT_IND  1
DEFAULT_NSW   1
DEFAULT_URF   0.6
DEFAULT_SOR   1.e-7
DEFAULT_GDS   0.0
UIN     0.
VIN     0.
WIN     0.
TIN     0.
TEIN   0.01
EDIN   0.01
!-----------------------------------------------------
----------------------
! 1 = NSW  (Number of iterations)
! 2 = URF  (Underrelaxation factor)
! 3 = SOR  (convergence criterion for inner iterations)
! 4 = GDS  (Blending factor)
! 5 = printv (print variable in ascii form) (default = 0)
!     0  = never
```

```
!     1  = variable on finest grid to output file
!     2  = variable on finest grid to TECPLOT file
!     3  = variable on finest grid to AVS file
! 6 = Record to read from (default = 0)
! 7 = Record to write to  (default = 0)
!
! variable                      1      2     3     4  5  6  7
!-------------------------------------------------------
----------------------
U-VELOCITY                     4    0.7   .    0.0   2   1   1
V-VELOCITY                     4    0.7   .    0.0   2   1   1
W-VELOCITY                     4    0.7   .    0.0   2   1   1
PRESSURE                      10    0.3   .     .    2   1   1
TURBULENT_KINETIC_ENERGY       0    0.6   .     .    .   .   .
TURBULENT_DISSIPATION          0    0.6   .     .    .   .   .
TEMPERATURE                    0    0.6   .     .    .   .   .
VISCOSITY                      1    1.0   .              1   1
EDDY_VISCOSITY                 0    0.9   .              1   1
DENSITY                        1     .    .              1   1
PRESSURE-CORRECTION            1    0.1   2              1   1
FLUX_IN_X-DIRECTION            1     .    .              1   1
FLUX_IN_Y-DIRECTION            1     .    .              1   1
FLUX_IN_Z-DIRECTION            1     .    .              1   1
```

Bibliografie

[1] T. Peeters en P. Boerstoel, *X-stream, Programmer's Guide, TNO-TPD Internal Report, Delft, 1999*

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

[3] L. Post, *Modelling of flow and Combustion in a Glass Melting Furnace, Thesis, TU Delft, 1988*

[4] J.H. Ferziger and M. Peric, *Computational Methods for Fluid Dynamics, Second Edition, Springer, Berlin, 1999*

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

[6] J.H. Ferziger, *Numerical Methods for engineering application, Second Edition, John Wiley and Sons, Danvers (USA), 1998*

[7] T.M.R. Ellis, I.R. Philips, and T.M. Lekey, *Fortran 90 programming, Second Edition, Strathclyde, John Wiley and Sons, 1994*

[8] K. M. Smith, W. K. Cope, and S. P. Vanka, *A Multigrid Procedure for three-Dimensional Flows on Non-Orthogonal Colocated Grids, Intl. Journal for Num. Methods in Fluids, Vol. 17, pp. 887-904, 1993*

[9] M. Hortmann, M. Peric, and G. Scheuerer, *Finite Volume Multigrid Prediction of Laminair Natural Convection: Benchmark Solutions, Intl. Journal for Num. Methods in Fluids, Vol. 11, pp. 189-207, 1990*

[10] C. Becker, J. H. Ferziger, M. Peric, and G. Scheuerer, *Finite Volume Multigrid Solutions of Two-Dimensional Incompressibele Navier-Stokes Equations, Notes on Numerical Fluid Mechanics, Vol. 23, pp. 37-47, 1988*

[11] I. Demirdzic, Z. Lilek, and M. Peric, *Fluid Flow and Heat Transfer Test Problem for Non-Orthogonal Grids: Convection: Benchmark Solutions, Intl. Journal for Num. Methods in Fluids, Vol. 15, pp. 329-354, 1992*

[12] P. K. Khosla and S. G. Rubin, *A Diagonally Dominant Second Order Accurate Implicit Scheme, Comput. Fluids, Vol. 2, pp. 207-209, 1974*

[13] Z. Lilek, S. Muzaferija, and M. Peric, *Efficiency and Accuracy Aspects of A Full-Multigrid SIMPLE Algorithm for Three-Dimensional Flows, Numerical Heat Transfer, Part B, Vol. 31, pp. 23-42, 1997*

[14] P. Wesseling, *An Introduction to Multigrid Methods, John Wiley and Sons, Chichester, 1992, www.mgnet.org/mgnet-books-wesseling.html*