Delft, Maart 2006

Visualiseren van wandelende zandgolven in de Westerschelde

AE2-002 – project

Groep N22A

Steven Bakker	1229265
Diane Cleij	1215434
Mark Groen	1221620
Bart Hummelink	1146726
Jonatan Leloux	1223380
Dante Rogiest	1239635
Roel Vandeberg	1173375
Haci Yildizturan	1205366

TU Delft
Luchtvaart- & Ruimtevaarttechniek

Voorwoord

Dit is het eindverslag van het tweedejaarsproject (AE2-002), van studenten luchtvaart- en ruimtevaarttechniek. Het eindverslag behandelt het in kaart brengen van wandelende zandgolven op de bodem van de Westerschelde.

Deze opdracht heeft als doel het verwerken van verkregen meetresultaten uit de Westerschelde met behulp van een zelfgeschreven computerprogramma in Java. De resultaten geven informatie over de bodem van de Westerschelde, in het bijzonder bepaald over de wandelende zandgolven die hierop aanwezig zijn.

Graag bedanken wij onze opdrachtgevers, R.C. Lindenbergh en M. Snellen, voor hun steun bij dit project. Ook willen wij graag P. Post bedanken voor alle positieve kritiek en input. Verder zijn wij ook nog dank verschuldigd aan onze projectbegeleider T. Epema.

Delft, maart 2006

Steven Bakker
Diane Cleij
Mark Groen
Bart Hummelink
Jonatan Leloux
Dante Rogiest
Roel Vandeberg
Haci Yildizturan

Samenvatting

De Westerschelde is de monding van de rivier de Schelde in de Noordzee in het zuidwesten van Nederland. Het is een belangrijke vaarroute naar de haven van Antwerpen. De Westerschelde staat in open verbinding met de zee, daarom verandert de bodem constant onder invloed van de getijdenwerking. Dit veranderen zou het scheepsverkeer kunnen verstoren doordat de vaargeulen verzanden, daarom is het belangrijk om deze veranderingen te beschrijven.
Het weergeven van veranderingen in diepte wordt gedaan aan de hand van een zogenaamde 'Multi Beam Echo Sounder' (MBES). Hierbij wordt de tijd tussen het uitzenden en ontvangen van een puls gerelateerd aan de diepte van dat punt.

Het doel van dit rapport is het beschrijven van een JAVA programma dat de zandgolven en hun oriëntatie in een gedeelte van de Westerschelde in kaart te brengt en analyseert.
Voor het beschrijven van zandgolven wordt een dataset gebruikt met dieptegegevens en de bijhorende x- en y-coördinaat. Deze dataset is verkregen uit de MBES metingen in de Westerschelde.
Voor het beschrijven van de vorm en grootte van de zandgolven worden hun richtingscoëfficiënten en amplitude berekend met het programma. De asymmetrische vorm van de zandgolven bepaalt de richting van de stroming in dat gebied. De richtingscoëfficiënt wordt verstoord door algemene dieptes als vaargeulen en zandbanken. Het programma zal daarom eerst deze algemene dieptes uit de dataset filteren. Hierna wordt de nieuwe dataset verwerkt om reliëf-, vector- en amplitudekaarten te maken.

Op de verkregen kaarten zijn de stromingsrichting, amplitude en grootte van de zandgolven en de diepte duidelijk te zien. De algemene stromingsrichting is in de richting van het noordwesten. De berekende amplitude hangt af van de diepte waarop de zandgolven liggen ten opzichte van het NAP. In laag gelegen gebieden (tussen de 5 en 10 meter diepte) zijn de zandgolven groter dan in hoger gelegen gebieden (tussen de 0 en 5 meter diepte).
Aan de randen van het beschreven gebied geeft het programma geen correcte data. Dit ligt aan de manier waarop het programma geschreven is. Hier zou in de toekomst beter naar gekeken kunnen worden. Ook het uitfilteren van de algemene diepte uit de originele data kan op een meer nauwkeurige manier worden gedaan, namelijk door gebruik te maken van de Fourier analyse.

Inhoudsopgave

1. Inleiding

De Westerschelde is de monding van de rivier de Schelde in de Noordzee in het zuidwesten van Nederland. Het is een belangrijke vaarroute naar de haven van Antwerpen. De Westerschelde staat in open verbinding met de zee, daarom verandert de bodem constant onder invloed van de getijdenwerking. Dit veranderen zou het scheepsverkeer kunnen verstoren doordat de vaargeulen verzanden, daarom is het belangrijk om deze veranderingen te beschrijven.

Dit rapport heeft tot doel een methode te beschrijven om de zandgolven en hun oriëntatie in de Westerschelde in kaart te brengen. Deze methode zal worden uitgewerkt in een computerprogramma, geschreven in de programmeertaal JAVA. Een manier om de veranderingen van de zeebodem in kaart te brengen, is door te kijken naar de vorm van de zandgolven. Deze wandelende zandgolven worden gevormd door het water, dat het zand opstuwt in de richting van de stroming. De zandgolven zijn verschillend van amplitude en golflengte. Er zal dus ook een manier moeten worden gezocht om deze te onderscheiden van andere oneffenheden op de bodem zoals vaargeulen en zandbanken.

Hoofdstuk 2 beschrijft een deel van de bedding van de Westerschelde en de zandgolven op deze bodem. Hoofdstuk 3 bevat de eisen aan het resultaat van de opdracht. Hoofdstuk 4 beschrijft de manier van onderscheiden van zandgolven op de bodem, nodig om met de data de zandgolven te visualiseren. In hoofdstuk 5 staat een computerprogramma uitgewerkt dat met de data de zandgolven visualiseert. De reliëf- en vectorkaarten gemaakt aan de hand van het computerprogramma worden beschreven in hoofdstuk 6. Ten slotte volgen in hoofdstuk 7 de conclusies en aanbevelingen voor vervolgonderzoek en verbetering van de resultaten.

2. De bedding van de Westerschelde

Het gebied dat in dit verslag onderzocht wordt is een gedeelte van de bodem van de Westerschelde. In dit hoofdstuk wordt de exacte ligging van het gebied bepaald (§2.1) en uitleg gegeven over de zandgolven op de zeebedding (§2.2). In paragraaf 2.3 wordt uitgelegd hoe er dieptemetingen gedaan worden.

2.1 De ligging van het onderzoeksgebied

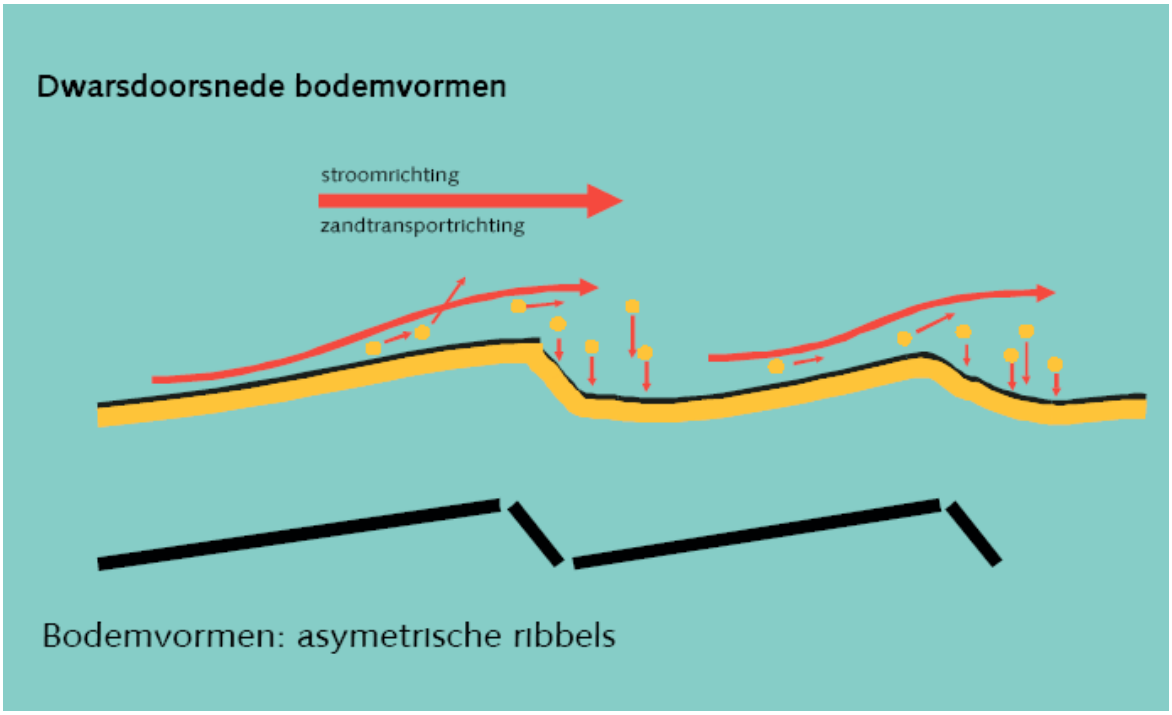
De data die ter beschikking werd gesteld voor dit bodemonderzoek heeft een bereik van 572000-575000 in de x-richting en van 5693000 tot 5697000 in de y-richting. Dit is het zogeheten UTM coördinatensysteem en kan omgerekend worden in de lengte- en breedteligging. Wanneer deze resultaten dan in kaart worden gebracht is te zien dat de data een gebied in de Westerschelde (zie figuur 2.1), ter hoogte van het dorpje Walsoorden, beslaat.

Figuur 2.1: Gedeelte van de Westerschelde dat onderzocht wordt, http://www.earth.google.com

Hierop is duidelijk te zien dat in de metingen ook gebieden voorkomen die boven de zeespiegel liggen (de rechthoek geeft het gebied weer). In het midden zou ook een zandbank uit de data tevoorschijn moeten komen, zoals te zien is op deze kaart.

2.2 Wandelende zandgolven op de bodem

De bodem van de Westerschelde bestaat voornamelijk uit zand. Dit zand is onderhevig aan de getijdenwerking in de Westerschelde. De bodem is dynamisch, wat wil zeggen dat de bodem verandert ten gevolge van de stroming. Zandgolven vormen zich op de bodem en hebben een karakteristieke vorm (zie figuur 2.2), waaruit de stromingsrichting van het water afgeleid kan worden. Als achtergrondinformatie is een publicatie uit het Nederlands tijdschrift voor natuurkunde geraadpleegd.



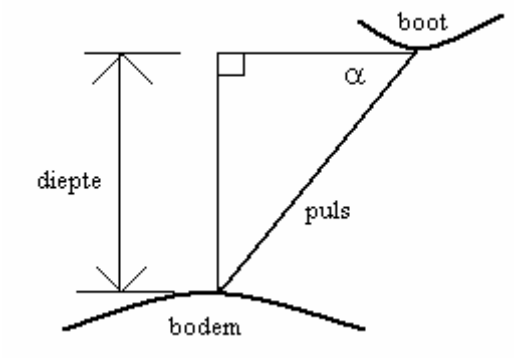
Figuur 2.2: Schematische weergave van de asymmetrische zandgolven en de stromingsrichting, http://www.rikz.nl

De zandgolven zijn asymmetrisch, ze hebben een steile en een minder steile kant. Uit deze eigenschap is de stromingsrichting af te leiden. De stromingsrichting is parallel met de vector die vanaf de minder steile kant naar de top van de zandgolf wijst. De steile kant ontstaat door het wervelen van het water, als het de top van de zandgolf voorbij is.

De richting van de verplaatsing van de wandelende zandgolven is gelijk aan de stromingsrichting. De snelheid waarmee deze zandgolven zich verplaatsen is recht evenredig met de sterkte van de stroming: hoe sterker de stroming hoe sterker de vervorming van de bodem. De verplaatsingsrichting van de zandgolven kan worden gecontroleerd door het verwerken van verschillende datasets. Deze datasets zijn metingen, genomen op een verschillend tijdstip, waaruit de verplaatsing van de zandgolven zou moeten blijken. Er is gekeken naar één dataset, de methode van meerdere datasets kan gebruikt worden ter controle.
De hele bodem is bezaaid met zandgolven, die grotendeels parallel aan elkaar lopen. Uit deze golven kan de amplitude gehaald worden, dat is de helft van het verschil tussen het hoogste en het laagste punt van de golf.

2.3 Dieptemeting met behulp van MBES

Met behulp van 'Echo Sounding' wordt de diepte van het water gemeten. Deze techniek werkt door het uitzenden van akoestische pulsen. De pulsen die uitgezonden zijn vanaf de onderkant van de boot worden door de zeebodem teruggekaatst (zie figuur 2.3). De teruggekaatste pulsen worden ontvangen en verwerkt door hetzelfde systeem. De tijd tussen het uitzenden en het ontvangen van de pulsen wordt gebruikt voor de bepaling van de diepte en de positie van het gemeten punt.



Figuur 2.3: De werking van een 'Echo Sounder'

De snelheid van een akoestische puls in het water is bekend. De afgelegde afstand van een puls kan als volgt worden berekend:

$$s_{pulse} = V_{pulse} \cdot \frac{t}{2} \tag{2.1}$$

Waarbij:

s_{pulse} = de afgelegde afstand van een puls, van de boot tot de zeebodem.
V_{pulse} = de snelheid van een akoestische puls in het water.
t = de tijd tussen het uitzenden en het ontvangen van een puls

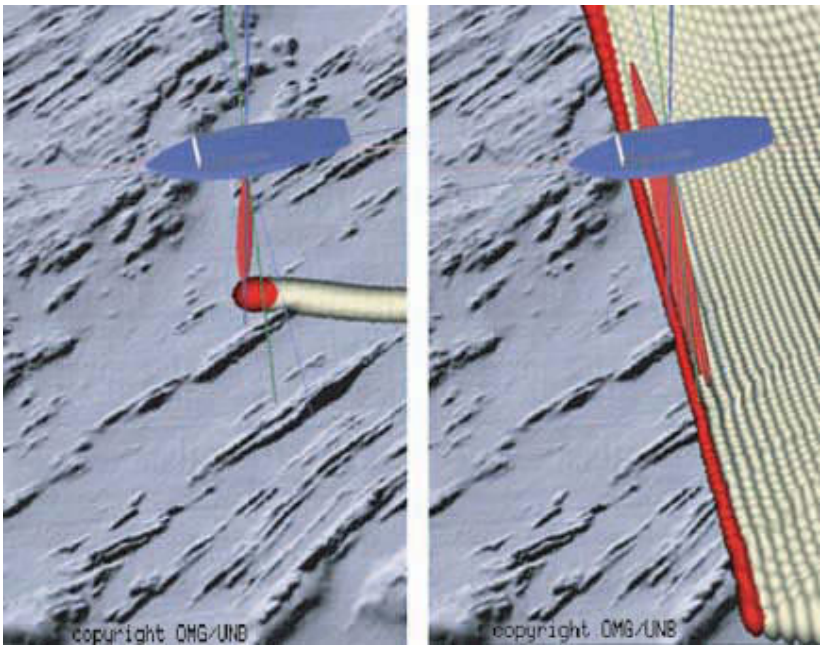
Met goniometrie kan de diepte op het teruggekaatste punt bepaald worden:

$$diepte = s_{pulse} \cdot \sin\alpha \tag{2.2}$$

Waarbij:

α = de bekende hoek tussen de uitgezonden puls en de horizon.

De coördinaten van het gemeten punt worden bepaald ten opzichte van het schip. Ze worden berekend door de afgelegde afstand van een puls in de richting van de horizon op te tellen/af te trekken van de y -coördinaat van de boot. De pulsen worden bij een SBES (Single Beam Echo Sounder) aan een kant van het schip uitgezonden. Met de MBES (Multi Beam Echo Sounder) kunnen de pulsen in de beide richtingen worden uitgezonden. Het voordeel van de MBES is vooral dat hiermee grotere gebieden in één keer kunnen worden onderzocht.



Figuur 2.4: Vergelijking tussen een 'Single' en een 'Multi Beam Echo Sounder', bron: University of Rhode Island [3]

De linkerzijde van figuur 2.4 geeft de SBES weer, terwijl aan de rechterkant een MBES te zien is. De rode cirkel en de rode lijn geven de punten die tegelijk gemeten worden weer. Hierop is duidelijk te zien dat de MBES een groter bereik heeft dan de SBES, omdat de eerste als een soort bezem een groot gebied in één keer scant. Het witte gedeelte in de figuur laat zien welke gebieden al onderzocht zijn (de boot vaart naar links). Een stukje van de verkregen data, gemaakt met een MBES, is te vinden in Appendix A. Appendix A bevat slechts een gedeelte van de dataset, omdat deze te groot is om volledig bij te voegen in dit rapport. Deze data heeft per meetpunt een x-coördinaat en een y-coördinaat en de diepte op dat punt. Uit deze data wordt de uiteindelijke dataset geïnterpoleerd, deze uiteindelijke dataset is de dataset die voor dit rapport is gebruikt.

3 Eisen aan de visualisatie van de zandgolven

In dit hoofdstuk worden de eisen besproken die gelden voor het bepalen van de oriëntatie en de amplitude van de zandgolven. Hier zal de aandacht worden gevestigd op de randvoorwaarden in paragraaf 3.1 en uitgangspunten in paragraaf 3.2.

3.1 Randvoorwaarden

Technisch-constructieve randvoorwaarden (TCR)

TCR1: Het JAVA programma moet de dataset in kunnen lezen. Deze dataset heeft een resolutie van een meter. Dat wil zeggen dat per meter in x- en y-richting een dieptemeting is verkregen. De nauwkeurigheid van de dieptemeting is in de orde van centimeters. Een gedeelte van de data is te zien in Appendix A.

TCR2: Er moet een JAVA programma geschreven worden zodat de reliëf- en vectorkaarten gemaakt kunnen worden. De kaarten bevatten de volgende gegevens van de zeebodem: de amplitude van de zandgolven, stromingsrichting van het water en de diepte. De kleur van de pijltjes geeft de diepte weer. De lengte van de pijltjes geeft de amplitude weer. De richting van de pijltjes geeft de oriëntatie van een zandgolf weer.

Natuurlijke randvoorwaarden (NR)

NR1: Doordat de meting is gemaakt met een MBES-systeem, kunnen sommige metingen ontbreken. Dat komt doordat er een obstakel aanwezig kan zijn tussen de zeebodem en het schip dat de metingen uitvoert. Als gevolg hiervan zitten er gaten en onnauwkeurigheden in de metingen.

3.2 Uitgangspunten

Technisch-constructieve uitgangspunten (TCU)

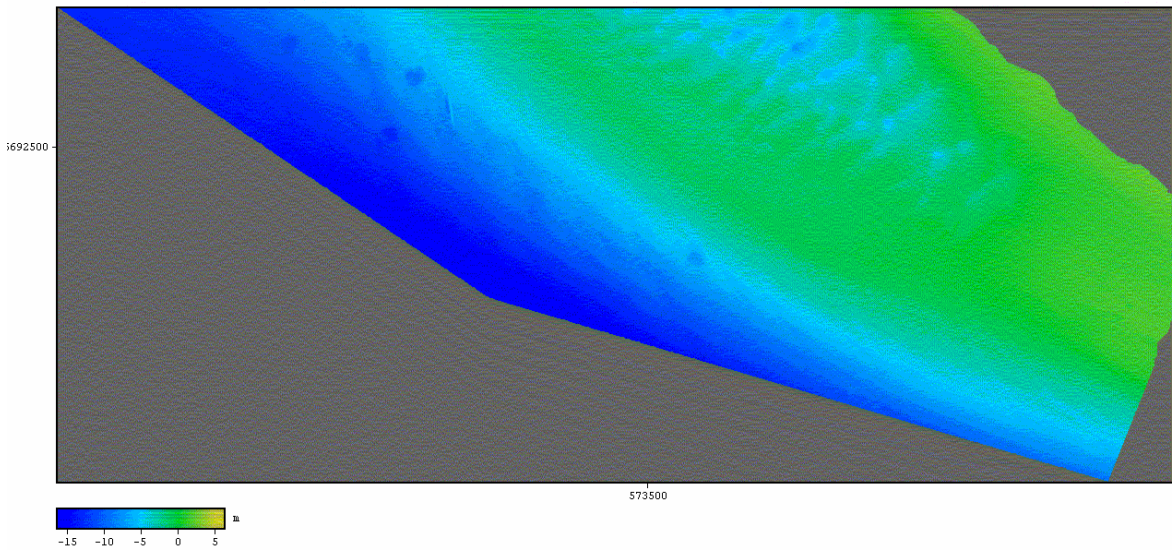
TCU1: Door beperkingen van computers en de JAVA omgeving is de dataset opgedeeld in meerdere delen. Deze delen worden afzonderlijk door het computerprogramma verwerkt, waarna alle deelresultaten tot één geheel worden gemaakt.

4. Methoden voor het visualiseren van de zandgolven

Uit de dataset (voor een gedeelte van deze dataset, zie Appendix A) moet uiteindelijk een vectorkaart met diepteligging volgen. Dit resultaat wordt verkregen door een aantal stappen uit te voeren, die in dit hoofdstuk worden uitgelegd. De algemene werkwijze om een dieptekaart te maken is het onderwerp van paragraaf 4.1, terwijl in paragraaf 4.2 uitgelegd wordt hoe de vectorkaart wordt gemaakt. In paragraaf 4.3 staat een alternatieve methode beschreven aan de hand van een Fourier reeks. Deze methode is niet gebruikt voor het eindresultaat.

4.1 Dieptekaarten

De dataset is opgebouwd uit metingen, elk meetpunt heeft een x- en y-coördinaat en een diepte van dat punt. Hieruit kan een gedetailleerde dieptekaart gemaakt worden. Aan elke diepte wordt een andere kleur toegekend, wat een kleurenkaart geeft. Deze kleurenkaart geeft een beeld van het reliëf op de bodem. De gehele dieptekaart is opgebouwd uit stukken, omdat de dataset in delen ingelezen moet worden. In figuur 4.1 is een gedeelte van de dieptekaart te zien.



Figuur 4.1: Gedeelte van de dieptekaart, uit figuur B-1 van appendix B.

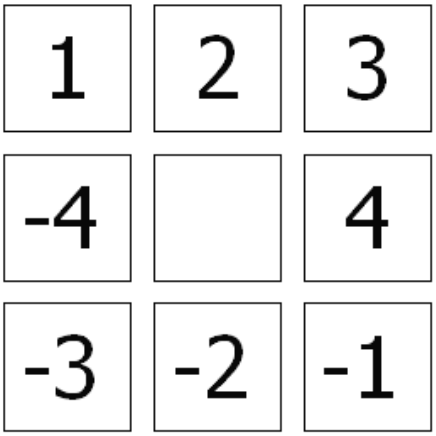
De diepte is te herkennen aan de kleuren die in de legenda zijn weergegeven. Hierop is duidelijk een ondiep stuk te zien (groen) en daarnaast een dieper gedeelte. Het donkerblauwe gedeelte geeft de dieper gelegen gebieden weer. De grijze stukken zijn plaatsen zonder metingen. De diepteverschillen zijn te zien aan het snel overgaan van kleuren. De dieptekaart van het gehele gebied is te zien in Appendix B, figuur B-1.

4.2 Stromingsvectoren op de zeebodem

De vectorkaart moet aan de eisen, beschreven in hoofdstuk 3, voldoen. De kleur van de pijltjes geeft de diepte van de zandgolven weer, hun lengte geeft de amplitude van de zandgolven, en de richting van de pijltjes is de oriëntatie van de zandgolven.

De oriëntatie van de zandgolven

De gebruikte methode om de richting uit de dataset te halen gaat als volgt: Vanuit een punt wordt gekeken naar de acht punten daaromheen, uit die acht punten wordt dan het hoogste punt (t.o.v. het punt van waaruit gekeken wordt) gehaald. Naar dit hogere punt wordt dan vervolgens een pijl getrokken. Het resultaat is een kaartje met pijltjes in acht mogelijke richtingen (zie figuur 4.2).



Figuur 4.2: Mogelijke oriëntatie van de pijlen

Dit proces wordt herhaald voor de gehele dataset, met als resultaat een pijlenkaart. Daarna wordt in een gebied van 45 bij 45 meetpunten (gelijk aan 45 bij 45 meter) de pijltjes gegeneraliseerd. Van de pijltjes in dat gebied wordt dus één lijn gemaakt, aan de hand van het optellen van vectoren in dezelfde richting. De richting waarin het resultaat van deze optelsom het grootst is, is de gegeneraliseerde richting. Omdat de pijltjes ook een tegengestelde richting kunnen hebben wordt enkel de richting weergegeven door een lijn. Het pijltje op de lijn wordt verkregen door te kijken naar de oriëntatie van de zandgolven. Er wordt gezocht naar de gegeneraliseerde lijntjes die op één lijn liggen. Van deze lijnen wordt dan de richtingscoëfficiënt bepaald. Als deze informatie wordt gekoppeld aan de asymmetrie van een zandgolf (zie §2.2) kan geconcludeerd worden dat de richtingscoëfficiënt van een lijn op de steile kant veel groter is dan dat op de minder steile kant. Hieruit volgt dat de pijl in de richting moet staan waarin de richtingscoëfficiënt gemiddeld het kleinst is. De gehele pijlenkaart kan worden gevonden in Appendix B, figuur B-4.

Amplitude van de zandgolven

De amplitude wordt weergegeven door de lengte van de pijlen in de vectorkaart. Aan de hand van de dieptekaart worden vaargeulen en grote zandbanken uit het bodemprofiel gefilterd. Nu blijft er een bodemkaart over waarop enkel de zandgolven te zien zijn. Dit maakt het afleiden van de amplitudes een stuk makkelijker. De richting van de zandgolven is bekend en op deze lijn wordt in de bodemkaart gekeken naar het verschil tussen de hoogste en de laagste punten. Dit verschil is de amplitude van de golf. Kaarten waarin de amplitudes van het gebied worden weergegeven staan in Appendix B.

4.3 Fourier reeks methode

Voor het bepalen van de oriëntatie en de amplitude van de zandgolven is de algemene trend van de zandgolven nodig. De rimpels worden gefilterd zodat er een functie bepaald wordt van de algemene trend. Dit kan bereikt worden door gebruik te maken van een Fourier reeks. De benodigde vergelijkingen zijn de volgende:

$$f(x) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty} a_n \cos(nx) + \sum_{n=1}^{\infty} b_n \sin(nx) \tag{4.1}$$

$$a_n = \frac{1}{L}\int_0^{2L} f(x') \cos\left(\frac{n\pi x'}{L}\right) dx' \tag{4.2}$$

$$b_n = \frac{1}{L}\int_0^{2L} f(x') \sin\left(\frac{n\pi x'}{L}\right) dx' \tag{4.3}$$

Waarbij:

f(x) = de gefilterde trendlijn
f(x') = de functie van de datapunten
n = 0, 1, 2, 3 tot oneindig
L = de totale lengte van de x-as

De functie *f(x')* is niet bekend, in plaats van *f(x')* kan de dieptewaarde voor elk punt worden ingevuld. Om de coëfficiënten a_n en b_n te berekenen worden de uitkomsten voor elk punt langs de x-as bij elkaar opgeteld. Hoe groter *n* wordt hoe meer de functie *f(x)* de functie *f(x')* benaderd. Om de algemene trendlijn te bepalen moet n niet te groot genomen worden. De grafiek van de functie *f(x)* kan geplot worden voor elke *n*. Zo kan worden bepaald welke *f(x)* het beste past bij een algemene trend.

Het geschreven JAVA programma leest de dataset in. Vervolgens bepaalt het programma de totale lengte van de x-as. Voor elk punt wordt de coëfficiënten a_n en b_n berekend. Vervolgens wordt *f(x)* bepaald voor elk punt. Het geschreven JAVA programma verwerkt de ingelezen dataset niet goed. Daarom is het gewenste resultaat niet bereikt. De Fourier reeks zou een goede alternatieve methode kunnen zijn, om de algemene trend van de zandgolven te bepalen, mits deze werkt.

5. Visualiseren van de zandgolven met behulp van Java

Het verwerken van de dataset en het uiteindelijke visualiseren van de zandgolven gebeurt met de computerprogrammeertaal JAVA. In dit hoofdstuk wordt uitgelegd hoe het ontwikkelde programma werkt. In paragraaf 5.1 is de structuur van het programma te zien. Verdere uitleg over de werking en details van het programma en de onderliggende methodes volgt in paragraaf 5.2.

5.1 Stroomdiagram van het hoofdprogramma

Het programma structuur diagram in Appendix C is zo opgebouwd dat het een helder beeld geeft van de verschillende methodes die worden gebruikt. De methodes zijn ingedeeld in vier groepen. Deze vier groepen zijn het inleesgedeelte, het filtergedeelte, het verwerkingsgedeelte en het outputgedeelte. De werking van iedere methode op zich wordt beschreven in paragraaf 5.2.

In het inleesgedeelte van het programma wordt de aangeleverde dataset verwerkt in een ‘array’. Deze ‘array’ is gevuld met objecten die worden aangemaakt in de methode ‘MyPoint.’ Aan één van de objecten wordt een dieptewaarde toegekend.

Het filtergedeelte van het programma verwijdert zoveel mogelijk het algemene diepteprofiel van het gebied. Dit levert een beter resultaat op bij de volgende verwerkingsstappen. De gegevens die uit de filter komen kunnen op drie manieren worden verwerkt. Ze kunnen worden weggeschreven naar een dataset, die vervolgens weer opnieuw kan worden ingevoerd en gefilterd. Wanneer de data genoeg gefilterd is kan deze worden doorgestuurd naar de verwerking. Daarnaast kunnen de gegevens worden ‘geplot’ door het ‘output’-gedeelte.

In het verwerkingsgedeelte worden de richtingen van de zandgolven bepaald. Deze richtingen worden hierna gegeneraliseerd voor gebieden van 45 bij 45 punten. Hier wordt ook de juiste richting toegekend aan de pijlen. Bovendien wordt hier de gemiddelde amplitude in het gebied berekend. Deze amplitude per gebied kan ook apart worden geplot door het outputgedeelte.

In het outputgedeelte kunnen de gegevens door middel van een ‘puntplotter’ of een ‘pijltjesplotter’ worden weergegeven in een ‘drawingwindow’. Deze ‘plotters’ halen de waardes voor hun kleuren uit verschillende methoden. De methoden ‘Kleur’, ‘KleurComp’ en ‘KleurAmp’ geven respectievelijk de kleuren voor de diepte, de gecompenseerde diepte en de amplitude.

5.2 Visualisatie van de Westerschelde bedding

Een methode is een deel van een computerprogramma, dat een voorgeschreven functie uitvoert. Hier worden de verschillende methodes van het programma uitgelegd. Een 'array' is een term in JAVA voor een database met daarin alle objecten die aan een punt zijn toegevoegd. Door middel van een 'array' kunnen eigenschappen worden weggehaald of toegevoegd.

5.2.1 Inlezen van de data in het programma

In dit gedeelte van het programma wordt de data eerst in kleinere datasets opgesplitst. Dit omdat de gehele data zo'n 10 miljoen punten bevat en dus niet in één keer kan worden ingevoerd. Vervolgens worden de datasets gelezen om de grootte van de aan te maken array vast te stellen. Hierna wordt deze array gevuld met de dieptewaardes uit de data. Aan elk punt van de array worden verschillende objecten toegevoegd, die worden aangemaakt in de methode 'MyPoint'.

'Samplecreator'

Omdat de hoeveelheid data erg groot is, is het handig om tussendoor testjes uit te kunnen voeren op kleinere gebieden om te kijken of het programma werkt. Daarom is er de methode 'samplecreator' geschreven. Deze methode krijgt de array binnen die gebaseerd is op het bestand dat ingelezen is en geeft vervolgens een kleinere array terug. Deze kleinere array beslaat een gebied uit het bestand dat zelf gekozen kan worden door X_{min}, X_{max}, Y_{min} en Y_{max} op te geven.
De 'samplecreator' heeft ook nog een tweede functie. Omdat de x-as maximaal 3300 pixels groot is, en de 'drawingwindow' alleen binnen de afmetingen van het bureaublad kan plotten, kan er niet in een keer een plot gemaakt worden in de gehele x-richting. Deze zal dus opgedeeld moeten worden in drie delen en in drie delen geplot moeten worden. Dit wordt ook gedaan door de 'samplecreator'.

'InputReaderLength'

In deze methode wordt een array aangemaakt. Er worden verschillende arrays met data gemaakt, omdat de gehele data te groot is om in één keer te verwerken. Voor het maken van een array is een dataset nodig waarbij elke rij of kolom dezelfde lengte heeft. Omdat dat bij de verschillende datasets niet het geval is zullen de maximale en minimale x- en y-waarden worden gevonden. Hier zal de grootte van de array op worden gebaseerd. Deze methode zal uiteindelijk de x- en y-lengte van de array kunnen geven. Ook kan deze methode worden aangeroepen om de minimale x- en y-coördinaten uit de originele dataset te halen.

'InputReader'

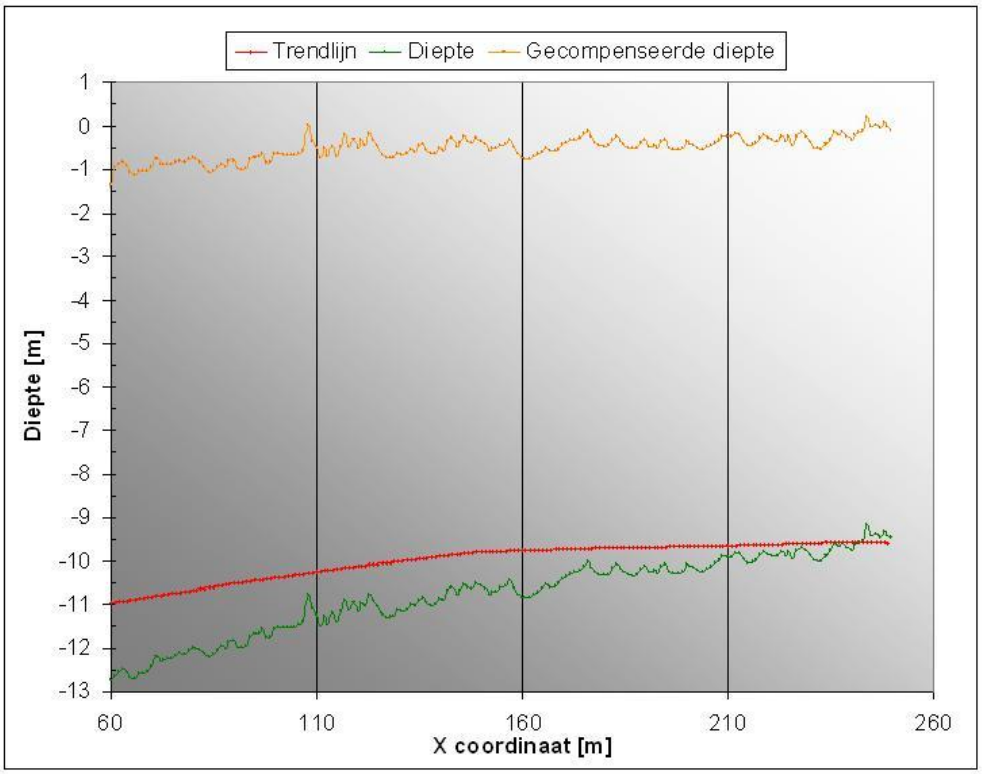
In deze methode wordt de data in de array gezet, die gebaseerd is op x- en y-lengte uit 'InputReaderLength.' Eerst wordt de array gevuld met een waarde van –1000. Dit wordt gedaan omdat er niet voor elk punt in de array een waarde zal zijn uit de dataset. Bij verdere berekeningen aan de dataset kunnen deze punten, door hun specifieke waarde, er eenvoudig worden uitgefilterd. In de dataset staan de x- en y-coördinaten en de diepte op dat punt. Deze worden uit de data gehaald en opgeslagen onder respectievelijk 'Xcoord, Ycoord, Depth.' De x- en y-coördinaten komen niet overeen met de plaatsnummers in de array. Om de data te laten beginnen bij punt (0,0) in de array worden de minimale x- en y-coördinaten van de originele coördinaten afgehaald. Op de punten in de array waar de dataset een waarde heeft zal de waarde van -1000 worden gewijzigd in de diepte uit de dataset. Uiteindelijk geeft deze methode een array met een deel van de dataset terug.

'MyPoint'

In deze methode worden zogenaamde 'setters' en 'getters' aangemaakt voor de volgende objecten: 'Depth', 'Vector', 'AverageVector', 'Dir', 'AverageDepth' en 'CompDepth'. Deze objecten kunnen worden aangeroepen en worden veranderd vanuit de andere methodes. In het object 'Depth' wordt de diepte opgeslagen of aangeroepen. In het object 'Vector' wordt een lengte en een richting aan de vector gegeven. In het object 'AverageVector' worden de x- en y-coördinaten en de hoek aan de gemiddelde vector toegewezen. Ook wordt meegegeven of er op het punt in de 'array' een pijltje moet worden weergegeven. In het object 'Dir' wordt gekeken of een pijltje een richting kleiner dan 0 heeft. Als dit het geval is, wordt het pijltje in tegenovergestelde richting geplaatst. In de objecten met 'AverageDepth' en 'CompDepth' worden variabelen met eenzelfde naam opgeslagen of teruggegeven.

5.2.2 Filteren van de data

In dit gedeelte van het programma wordt eerst, per gebied van 30 bij 30 punten, een gemiddelde diepte berekend. Deze gemiddelde diepte wordt gebruikt om de trendlijn van het totale gebied te berekenen. Vervolgens wordt deze trendlijn van de originele data afgehaald. De nieuwe data heet de gecompenseerde diepte, zie figuur 5.1.



Figuur 5.1: Verschillen in trendlijn, diepte en gecompenseerde diepte

'Gemdepth'

In deze methode wordt de gemiddelde diepte berekend van een bepaald vlak. Er wordt gekeken naar oppervlakten van 30 bij 30 meter. In de array staan de gemeten dieptes per meter opgeslagen onder 'Depth'. Met deze dieptes wordt de gemiddelde diepte bepaald van het vlak. Deze worden hierna opgeslagen in de array onder 'AvDepth'. Er zijn echter punten waar geen metingen van zijn gemaakt. Deze punten hebben een waarde –1000 meegekregen in de array. Die waardes worden niet opgenomen in de berekening van de gemiddelde diepte. Om deze methode te testen wordt ze toegepast op een testgebied. Het resultaat hiervan is geplot, zie figuur 5.2. Hierop is duidelijk te zien dat de methode aan vlakken van 30 bij 30 meter dezelfde waardes toekent.

Figuur 5.2: Voorbeeld plot van gemiddelde diepte

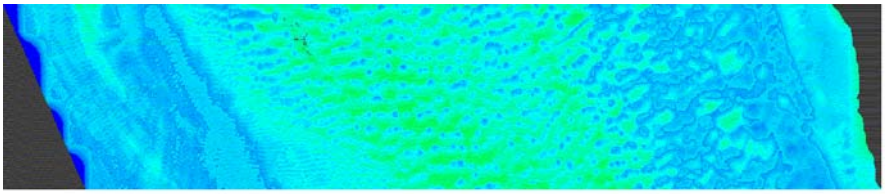
'Trendline'

In deze methode wordt een algemene dieptekaart gemaakt waar de zandgolven uit worden gefilterd. Hiervoor worden de waardes die onder 'AvDepth' zijn opgeslagen gebruikt. Eerst worden de waardes onder 'AvDepth' van twee naast elkaar liggende vlakken met elkaar vergeleken. Het verschil van de twee waarden wordt gedeeld door het aantal tussen gelegen punten. Deze waarde wordt dan vermenigvuldigd met de plaats van het punt en dit wordt opgeteld bij de gemiddelde diepte van het punt. Dit wordt herhaald voor alle vlakken zodat de overloop van elk vlak van links naar rechts vloeiend is. Hierna wordt dit alles herhaald maar dan worden er kolommen van 30 onder elkaar liggende punten vergeleken. De uiteindelijke dieptes worden weer opgeslagen onder 'AvDepth'. Van een gedeelte van de data is een plot gemaakt, te zien in figuur 5.3. Er is duidelijk een algemene diepte te zien, onafhankelijk van de zandgolven.

Figuur 5.3: Voorbeeld plot van de trend, algemene diepte

'CompDepth'

In deze methode wordt een dieptekaart gemaakt waarbij alleen de zandgolven zichtbaar zijn in plaats van de vaargeulen en andere algemene diepteverschillen.
Hiervoor worden de waardes onder 'AvDepth' van de waardes onder 'depth' afgehaald. Hierna blijven enkel de dieptes die de zandgolven definiëren over. Ook van deze diepte is een plot gemaakt, een voorbeeld is figuur 5.4. De gehele gecompenseerde dieptekaart is weergegeven in appendix B. De kleurverandering per diepte is nu minder omdat de diepteverschillen een stuk kleiner zijn. Met de nieuwe waarden voor de diepte zijn de zandgolven beter te beschrijven.



Figuur 5.4: Voorbeeld gecompenseerde diepteplot

5.2.3 Verwerking van de data

In dit gedeelte van het programma worden de gemiddelde richting en amplitude per gebied berekend. Met het gedeelte WriteToFile wordt de berekende data weggeschreven naar een nieuw bestand.

'Vector'

In deze methode wordt gekeken naar de helling in 8 verschillende richtingen rondom een bepaald punt. Aan elk punt wordt een richting en een lengte meegegeven. De richting wordt bepaald door de diepte van de omliggende punten. De richting die wordt meegegeven correspondeert met het hoogste omliggende punt. De lengte wordt bepaald door het diepteverschil tussen het hoogste omliggende punt en het punt zelf.

'Generalise'

In deze methode worden de vectoren per punt samengevoegd tot een gemiddelde vector per vlak. De grootte van dit vlak kan worden veranderd door de variabele resolutie aan te passen. Eerst worden per absolute richting de lengtes van de vectoren opgeteld, zodat er per richting een totale lengte ontstaat. Dit wordt gedaan voor alle vectoren binnen een vlak. Hierna worden de x- en y-waarden van de vector berekend per richting. Vervolgens wordt er een correctiefactor aangebracht voor de lengte van de vector die in het 'drawingwindow' wordt getekend. Hierna worden, aan bepaalde punten in een array, de lengte en de x- en y-waarden van de vector toegewezen. Dan wordt er via de methode 'RiCo' gekeken of de richting negatief of positief is, als deze negatief is zal er met 'MyPoint' via het object 'SetDir' bij de alfa 90 graden worden opgeteld. Aan het einde worden alle lengtes weer op 0 gezet en kan de algemene vector van het volgende vlak worden berekend. Welk waarde gekozen moet worden kan worden ingesteld. Dit gebeurt door de parameter 'choice' een waarde van 0 tot 2 mee te geven. 0 is voor de diepte, 1 is voor de gemiddelde diepte en 2 is voor de gefilterde diepte.

'Selectrichting'

Deze methode krijgt de richting, in de vorm van een getal tussen de –4 en 4 (zie figuur 4.2), binnen. Per richting wordt dan de hoek, die deze richting maakt met de lijn tussen het punt en richting 4, teruggegeven.

'Amplitude'

In deze methode wordt van een klein gebied uitgerekend wat de gemiddelde amplitude van de zandgolven is. De methode wordt aangeroepen met een x-coördinaat, een y-coördinaat, een straal en één van de richtingen zoals gedefinieerd in figuur 4.2. Dit laatste is mogelijk omdat wanneer deze methode wordt aangeroepen, de stromingsrichting al bekend is. In deze stromingsrichting wordt telkens over een lengte van 30 meter gekeken naar het hoogste en laagste punt. De maximale golflengte van een zandgolf is namelijk ongeveer 30 meter. Zo wordt over het hele gebied gekeken met intervallen van 10 meter. Uit alle diepteverschillen over dit gebied wordt een gemiddelde amplitude berekend.

'Richtingscoëfficiënt'

In deze methode wordt van een klein gebied uitgerekend welke richting de stroming op staat. In de methode 'Generalise' is voor dit gebied een algemene richting vastgesteld voor de stroming (loodrecht op de zandgolven). Deze richtingen zijn de volgende vier algemene richtingen uit de acht richtingen in figuur 4.2: 1 of –1, 2 of –2, 3 of –3 en 4 of –4. Deze methode geeft bijvoorbeeld als resultaat de richting 1 wanneer de algemene richting 1 of –1 bekend is. Deze methode maakt gebruik van het feit dat de zandgolven niet symmetrisch zijn zoals ook figuur 2.2 laat zien. Binnen deze methode wordt langs de algemene richting gekeken naar het gemiddelde van de positieve en de negatieve richtingscoëfficiënten. Wanneer de positieve richtingscoëfficiënten absoluut gezien kleiner zijn dan de negatieve staat de stroming in de richting waarlangs gekeken wordt. Zo niet dan staat de stroming 180 graden de andere kant op.

'WriteToFile'

De 'WriteToFile'-methode schrijft punten weg naar een '.pts'-bestand. Dit bestand heeft dezelfde opbouw als de ruwe data zodat via dezelfde methode, 'InputReader', ingelezen kan worden.
De file bestaat uit een drie kolommen met getallen. De eerste kolom is de x-waarde, i in dit geval, de tweede kolom is de y-waarde (j) en de laatste kolom bevat een parameter die bij het punt hoort. Dit zou een diepte kunnen zijn maar ook een waarde voor de amplitude van de golven of een waarde van de gefilterde diepte. Alleen de laatste optie is gebruikt.

5.2.4 Output van het programma

In dit gedeelte van het programma kunnen zowel de gekregen als berekende data worden geplot in een zogenaamd 'drawingwindow'. De richtingen worden geplot als pijltjes met een kleur die de diepte aangeeft. De overige data wordt weergegeven met een kleur per punt of gebied.

'Draw Vector'

De methode 'Draw Vector' maakt het mogelijk om in het 'drawingwindow' pijltjes te tekenen. Deze pijltjes geven de richting van de zandgolven aan.

'Colorplot'

De methode 'colorplot' maakt een 'drawingwindow' aan, gebaseerd op de lengte van de array die hij binnen krijgt. Dan tekent hij per positie in de array een pixel in een bepaalde kleur (afhankelijk van de diepte) en op een bepaalde positie (afhankelijk van de arraypositie). Wat dan volgt is een dieptekaart.

'Kleur', 'Kleurcomp' en 'kleuramp'

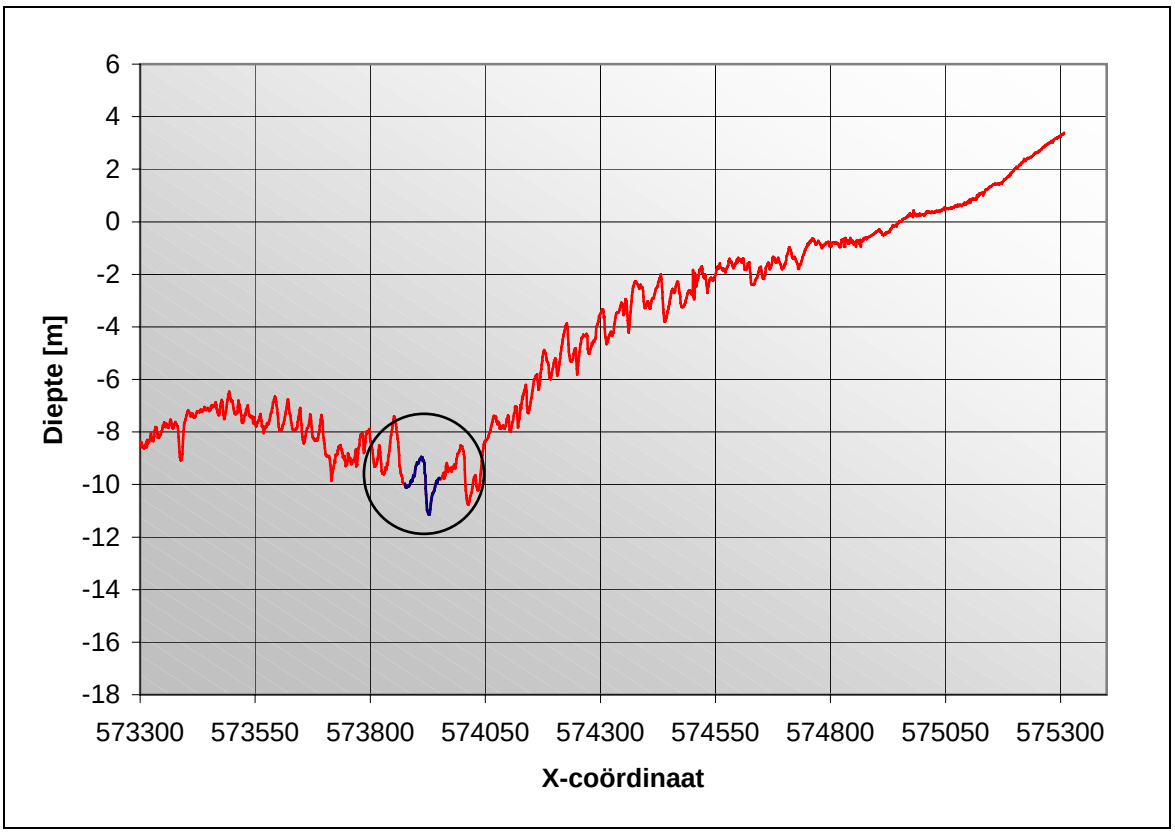
Deze methoden leveren de juiste kleur aan 'Colorplot' horende bij een bepaalde diepte. De kleuren die de methode 'Kleur' teruggeeft worden gebruikt bij het plotten van de werkelijke diepte en de kleuren die de methode 'Kleurcomp' teruggeeft worden gebruikt bij het plotten van de gecompenseerde diepte (waar de trend van afgetrokken is). De methode 'kleuramp' geeft de kleuren terug voor het plotten van de amplitude.

6. Resultaten

Dit hoofdstuk bevat de resultaten van het computerprogramma. Ten eerste wordt gekeken naar de output van het programma: de verdeling van de zandgolven op de bodem van de Westerschelde. Hierna volgt het computerprogramma zelf: hoe nauwkeurig zijn de resultaten en wat zijn hun beperkingen.

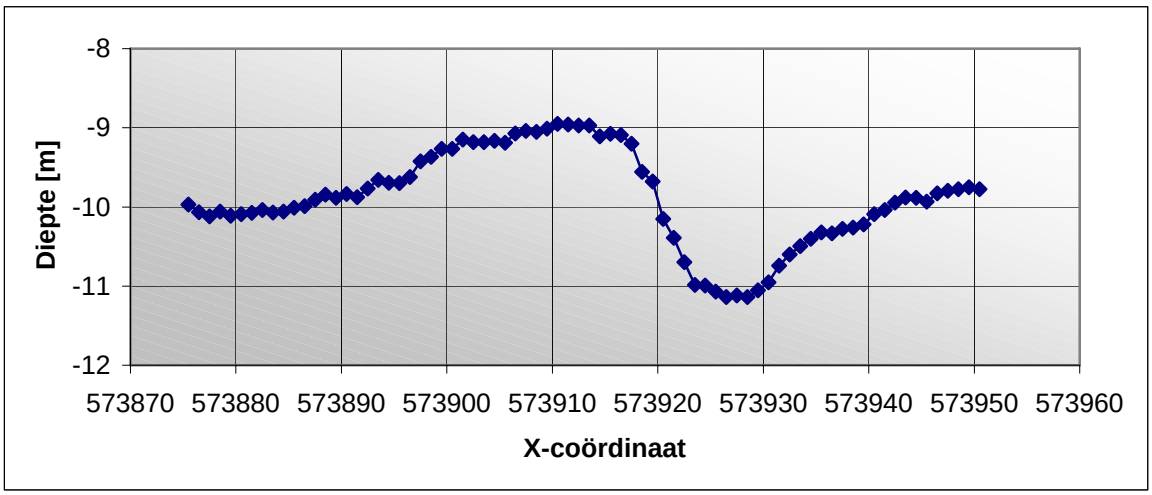
6.1 De verdeling en structuur van de zandgolven

Om een idee te krijgen over hoe de bodem er uitziet is halverwege het gebied een dwarsdoorsnede gemaakt van de bodem (figuur 6.1). In deze doorsnede is duidelijk de algemene trend te zien van de vaargeul oplopend naar de oever. Verder zijn zandgolven te zien die duidelijk groter worden op grotere dieptes.

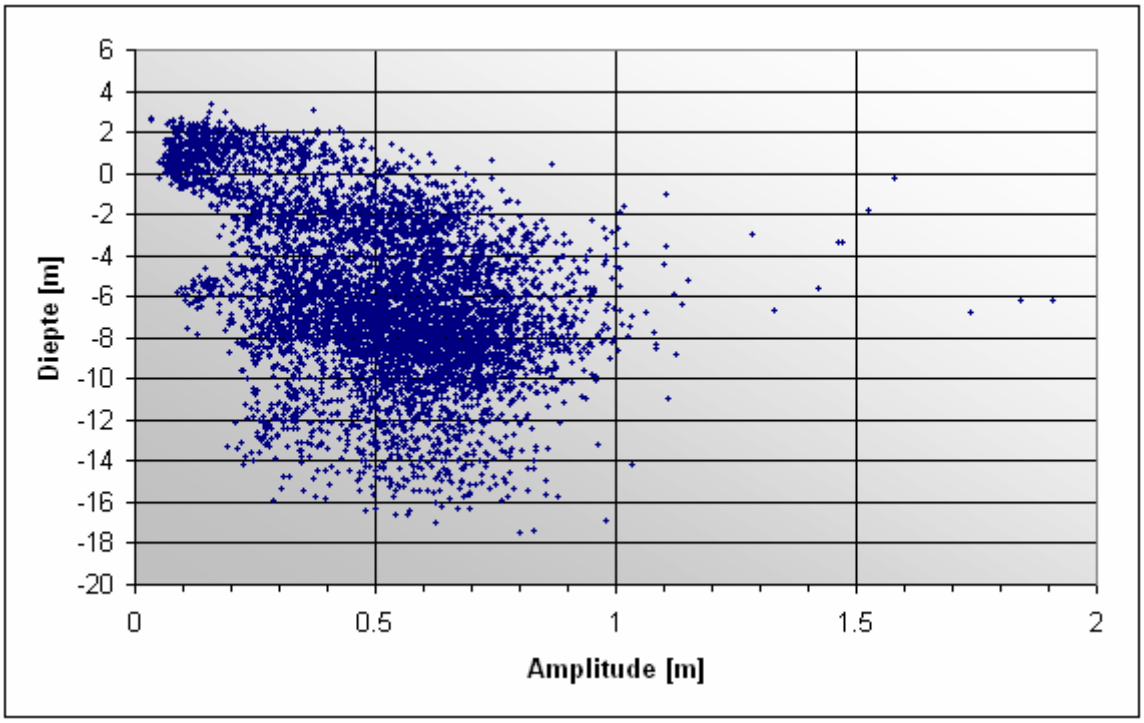


Figuur 6.1: Doorsnede van de bodem halverwege het gebied

In figuur 6.2 is ingezoomd op één zandgolf aangegeven in figuur 6.1. Deze kan worden vergeleken met de theoretische zandgolf in figuur 2.2. Voor de zandgolf in figuur 6.2 geldt dat hij asymmetrisch is. Dit komt overeen met de theorie en zou betekenen dat de stroming voor dit geval van links naar rechts zou staan. Verder is in figuur 6.2 te zien dat de golflengte van de zandgolf ongeveer 50 meter is en de amplitude ongeveer één meter.



Figuur 6.2: Eén zandgolf uit de doorsnede van figuur 6.1



Figuur 6.3: De amplitude van de zandgolven uitgezet tegen hun diepte

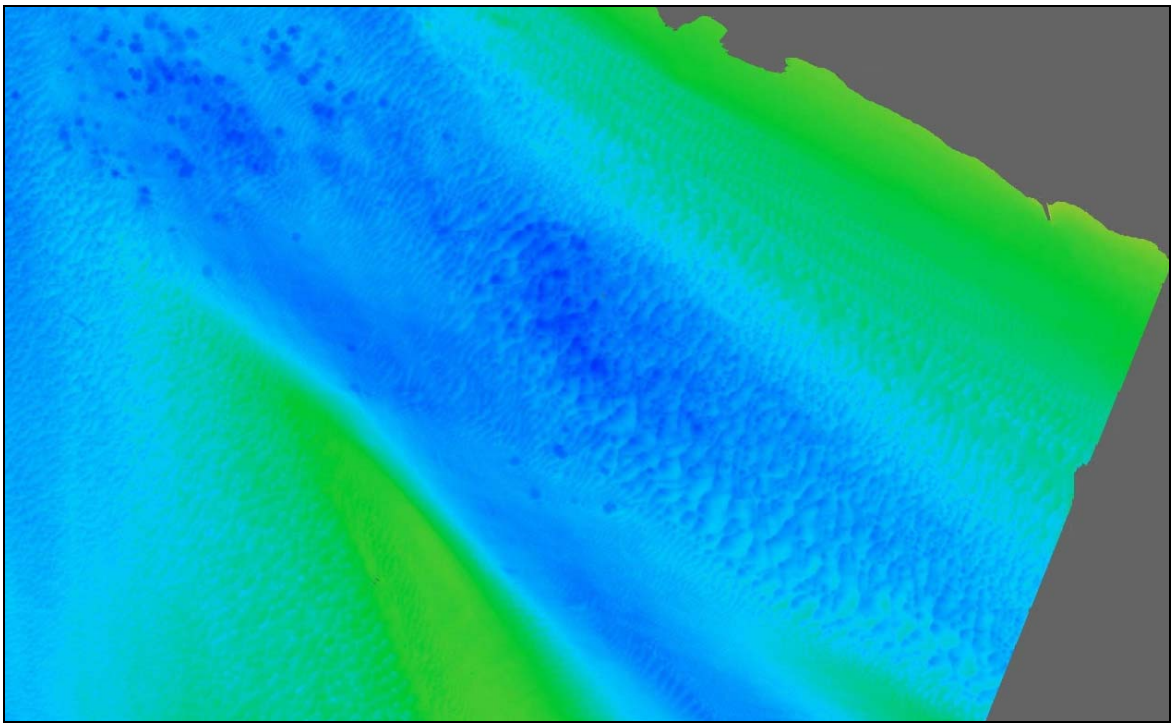
Figuur 6.3 toont het verband aan tussen de diepte waarop de zandgolven liggen en hun amplitude. Hieruit blijkt dat zandgolven die op een kleine diepte liggen ook een kleinere amplitude hebben.

6.2 De stromingen in de Westerschelde

In deze paragraaf wordt gekeken naar een illustratief deel in het noordoosten van het onderzoeksgebied. Hier zijn namelijk verschillende aspecten zichtbaar. Schuin door het midden van dit gedeelte loopt de vaargeul met rechtsboven de oever en linksonder de zandbank. Dit zelfde deel wordt op een aantal verschillende manieren bekeken. Als eerste wordt gekeken naar een diepteplot van dit gebied. Vervolgens naar een gecompenseerde diepteplot, een amplitudeplot en een vectorplot. De volledige plots zijn te zien in Appendix B.

Diepteplot

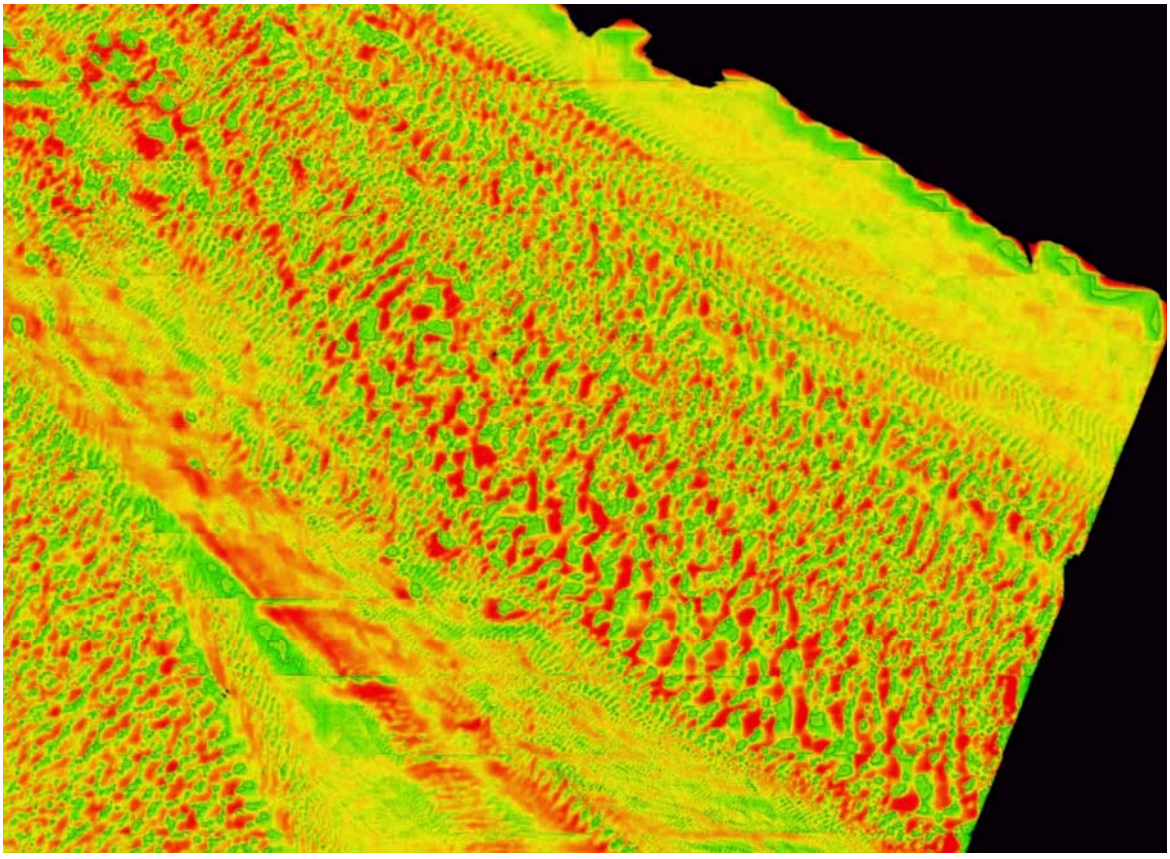
In de diepteplot (figuur 6.4) hebben de diepere delen een donkerblauwe kleur en de ondiepe delen een meer groene kleur. Hierdoor is duidelijk de vaargeul te zien, de oever en de zandbank. Ook zijn de zandgolven te zien, deze lijken groter te zijn in het midden van de vaargeul en minder groot aan de kant.



Figuur 6.4: Gedeelte van de dieptekaart uit figuur B-1 van appendix B

Gecompenseerde diepteplot

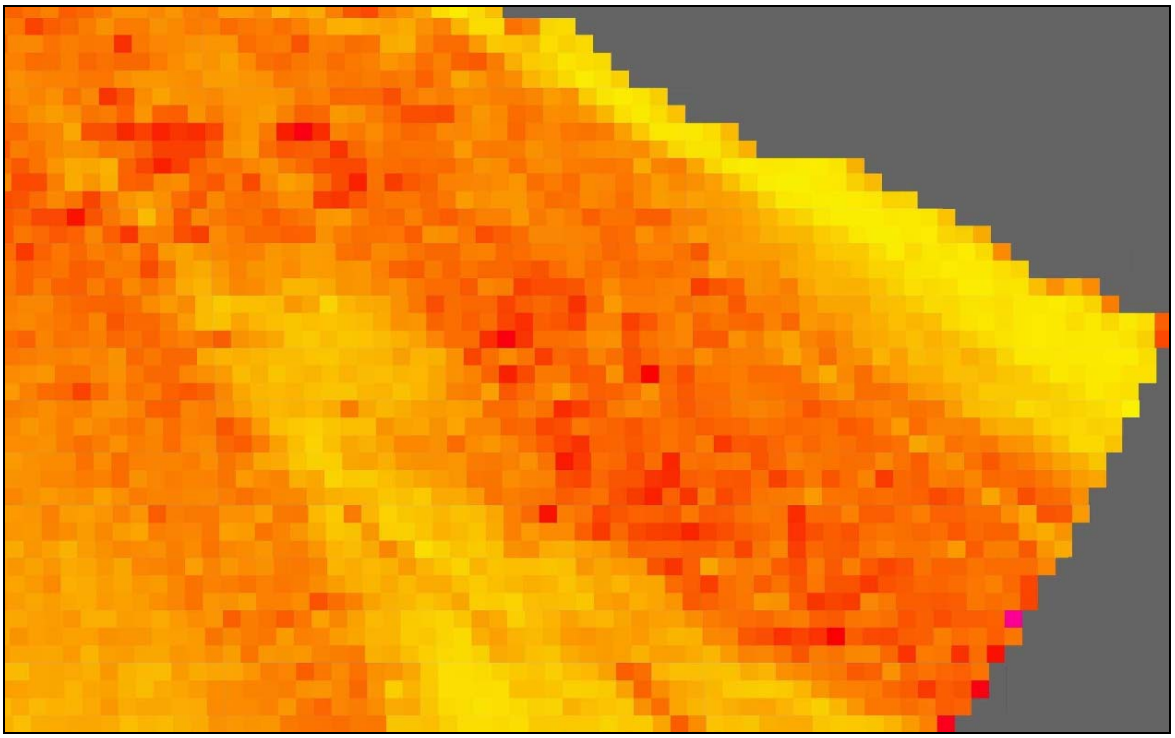
In de diepteplot in figuur 6.4 is duidelijk het diepteverschil te zien tussen de oever en de vaargeul. Dit is echter niet hetgeen waar het onderzoek om gaat. De informatie over de zandgolven is opgeslagen in kleinere diepteverschillen. Om dit duidelijker te maken is in de gecompenseerde diepteplot de algemene trend verwijderd. De gecompenseerde diepteplot is geïnverteerd, omdat de verschillen met de kleuren groen en rood duidelijker zichtbaar zijn, dan met de kleuren groen en blauw. In figuur 6.5 is een deel te zien, van figuur B-3 uit Appendix B, de kaart met de geïnverteerde gecompenseerde diepte. Wanneer de berekeningen voor de amplitude en de stromingsrichting op de gecompenseerde diepten wordt uitgevoerd zijn de uitkomsten hiervoor ook veel betrouwbaarder.



Figuur 6.5: Gedeelte van de geïnverteerde gecompenseerde dieptekaart uit figuur B-3 van Appendix B

Amplitudeplot

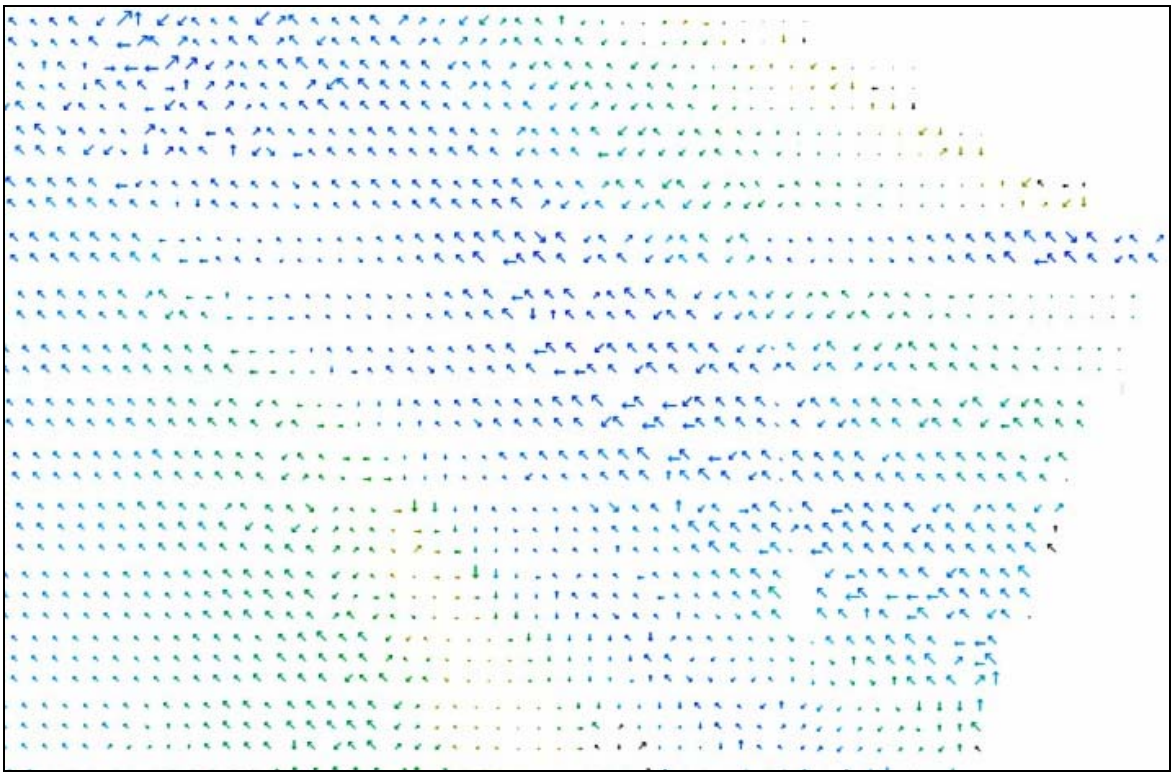
De amplitudeplot in figuur 6.6 laat zien dat de zandgolven in het midden van de vaargeul een grotere amplitude hebben en de zandgolven aan de kant een kleinere amplitude. Bij het berekenen van de amplitude blijkt dat deze lopen van 1,5 meter in de vaargeul tot kleiner dan 0,5 meter aan de kant. Wanneer de amplitudeplot (figuur 6.6) wordt vergeleken met de diepteplots (figuur 6.4 en 6.5) blijkt dat de zandgolven met een grotere golflengte ook een grotere amplitude hebben.



Figuur 6.6: Gedeelte van de amplitudekaart uit figuur B-4 van Appendix B

Vectorplot

In de vectorplot is gebruik gemaakt van alle vier de voorgaande plot. In deze afbeelding zijn pijlen weergegeven met de amplitude van de zandgolven als lengte en de diepte als kleur. De pijlkop geeft de stromingsrichting aan. Hiervoor is gebruik gemaakt van de gecompenseerde diepten. In figuur 6.7 is een duidelijke hoofdstroming zichtbaar in de vaargeul, waarvandaan de stroming afbuigt naar de kust. De amplitude van de zandgolven is in de diepere gebieden gemiddeld groter dan in de ondiepe. Dit is mogelijk te verklaren door een andere soort bodem of door verschillen in getijdenstromingen voor de ondiepe en diepere gebieden.



Figuur 6.7: Gedeelte van de vectorkaart uit figuur B-5 van Appendix B

7. Conclusies en aanbevelingen

Conclusies

Het eerste doel van dit project is een JAVA programma te schrijven om de zandgolven op de bodem van een gedeelte van de Westerschelde te beschrijven. Het programma maakt gebruik van de richtingscoëfficiënten van de zandgolven op de bodem. De richtingscoëfficiënten zijn afhankelijk van de vaargeulen en de zandgolven op de bodem. De resultaten van het programma zijn beter als de algemene diepte uit de data wordt gehaald. Daarom wordt eerst de algemene trendlijn van de bodem bepaald om de richtingscoëfficiënten van alleen de zandgolven te bepalen. Vervolgens worden de dieptes van de algemene trendlijn afgehaald van de dieptes verkregen uit de data van de 'Multibeam Echo Sounder' (MBES). Hierna worden de zandgolven beschreven door berekening van hun amplitude en hun richtingscoëfficiënten. De vorm van de zandgolven bepaalt de stromingsrichting van het water. Aan de hand van deze eigenschap van de zandgolven worden de stromingsrichtingen bepaald per gebied. Vervolgens worden de amplitudes en de diepte van de zandgolven en de stromingsrichting van het water weergeven in een afbeelding.

Het tweede doel van dit project is het analyseren van de afbeeldingen verkregen uit het JAVA programma. Bij een diepte groter dan 12 meter ten opzichte van de zeespiegel, komen er minder zandgolven voor op de bodem. De amplitudes lopen op van 0.5 meter aan de randen tot 1.5 meter in de vaargeulen. Bij een kleine diepte zijn de zandgolflengten kleiner dan bij een diepte tussen de 5 en 10 meter. Voorbij 10 meter diepte worden de golflengten geleidelijk aan weer kleiner, hier zijn de golflengten wel groter dan de golflengten bij een diepte kleiner dan 5 meter. Een verklaring hiervoor is dat de richting van de stroming aan het oppervlak van het water sterk wisselt en dus veel kleine golven veroorzaakt. Bij dieper gelegen gebieden is de stroming meer constant en veroorzaakt minder zandgolven met een grotere golflengte. De algemene stromingsrichting is in richting van het noordwesten. Er is te zien aan de pijltjes dat de stroming wordt verstoord door zandbanken die boven het wateroppervlak uit steken.

Aanbevelingen

De algemene trendlijn kan nauwkeuriger berekend worden met de Fourier reeks in plaats van de bepaling van de algemene trendlijn aan de hand van de diepteverschillen op de bodem. Met de Fourier reeks kan men zelf bepalen welke functie het beste past bij de algemene trendlijn van de bodem van de Westerschelde.

Op sommige punten, aan de randen en op zandbanken, zijn geen metingen gedaan met de MBES. Hierdoor is de werking van het JAVA programma in de buurt van deze gebieden niet optimaal. JAVA technisch gezien zou dit verbeterd kunnen worden.
Een probleem bij het creëren van het vectorplaatje, was dat de datasets erg groot waren. Hierdoor kan de data niet verder gegeneraliseerd worden, zodat grotere vectoren gemaakt zouden kunnen worden. Dit zou echter wel mogelijk zijn als de gegevens van deze vectoren in een nieuw bestand weggeschreven zouden worden en van daaruit opnieuw geopend en gegeneraliseerd zouden worden.

Literatuurlijst

Google Earth. Bekeken op 6 maart 2006, op
http://www.earth.google.com

Rijkswaterstaat (2004). Kustkalender 2005. Bezocht op 27 februari 2006, op de website van het RIKZ: http://www.rikz.nl

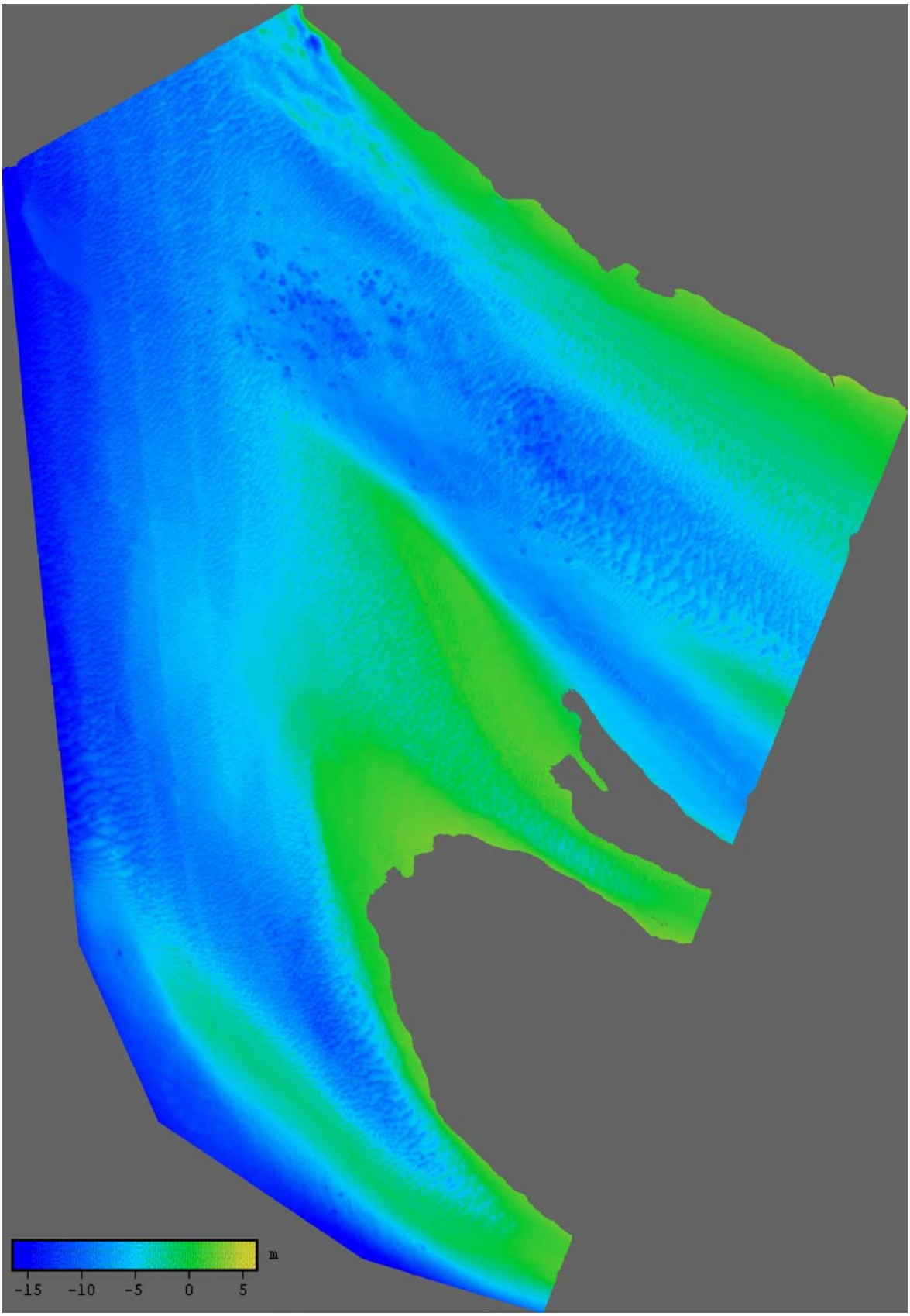
University of Rhode Island: Discovery of sound in sea, Echo-Sounder-Multibeam. Bekeken op 6 maart 2006, op
http://omp.gso.uri.edu/work1/gallery/tech/osf/esm1.htm

Nederlands tijdschrift voor Natuurkunde: Zandbanken en zandgolven. Publicatie van universiteit Twente door S. Hulscher en R van Damme.
Bekeken op 6 maart 2006, op
http://www.wem.ctw.utwente.nl/organisatie/Medewerkers/medewerkers/hulscher/Publicaties/NTV02H.pdf

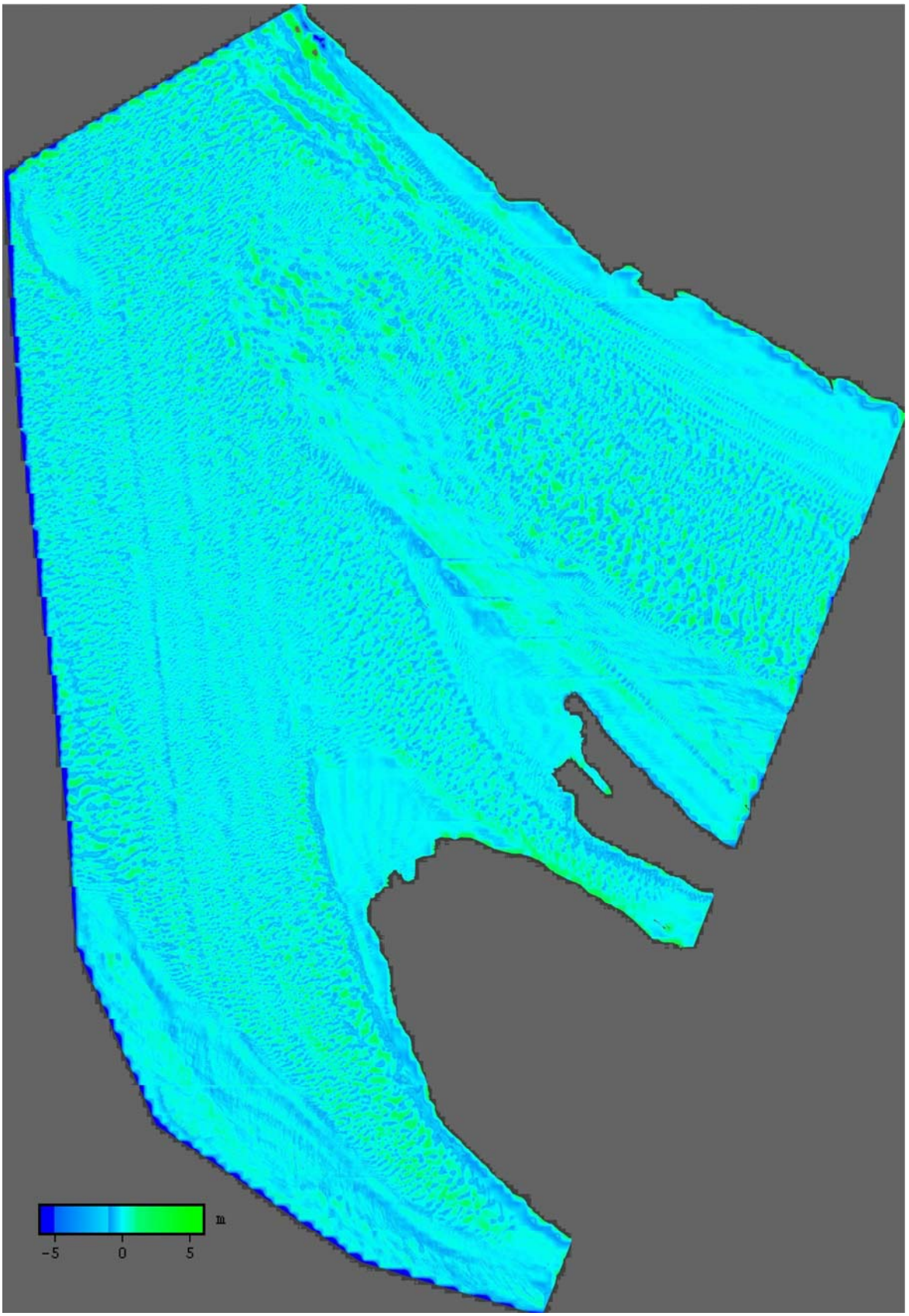
Appendix A: Gedeelte van de XYZ-data van de bodem

x-coördinaat	y-coördinaat	Diepte(m)
573999.500	5692137.500	-8.985
573995.500	5692138.500	-9.297
573996.500	5692138.500	-9.273
573997.500	5692138.500	-9.230
573998.500	5692138.500	-9.057
573999.500	5692138.500	-8.958
573992.500	5692139.500	-9.088
573993.500	5692139.500	-9.070
573994.500	5692139.500	-9.147
573995.500	5692139.500	-9.107
573996.500	5692139.500	-9.140
573997.500	5692139.500	-9.043
573998.500	5692139.500	-8.920
573999.500	5692139.500	-8.880
574000.500	5692139.500	-8.900
573989.500	5692140.500	-9.190
573990.500	5692140.500	-9.087
573991.500	5692140.500	-8.980
573992.500	5692140.500	-8.957
573993.500	5692140.500	-8.980
573994.500	5692140.500	-9.030
573995.500	5692140.500	-9.063
573996.500	5692140.500	-9.032
573997.500	5692140.500	-9.005
573998.500	5692140.500	-8.940
573999.500	5692140.500	-8.797
574000.500	5692140.500	-8.760
573985.500	5692141.500	-9.260
573986.500	5692141.500	-9.290
573987.500	5692141.500	-9.235
573988.500	5692141.500	-9.235
573989.500	5692141.500	-9.090
573990.500	5692141.500	-9.050
573991.500	5692141.500	-8.828
573992.500	5692141.500	-8.915
573993.500	5692141.500	-8.890
573994.500	5692141.500	-8.973
573995.500	5692141.500	-9.000
573996.500	5692141.500	-8.948
573997.500	5692141.500	-8.825
573998.500	5692141.500	-8.690
573999.500	5692141.500	-8.630
574000.500	5692141.500	-8.630

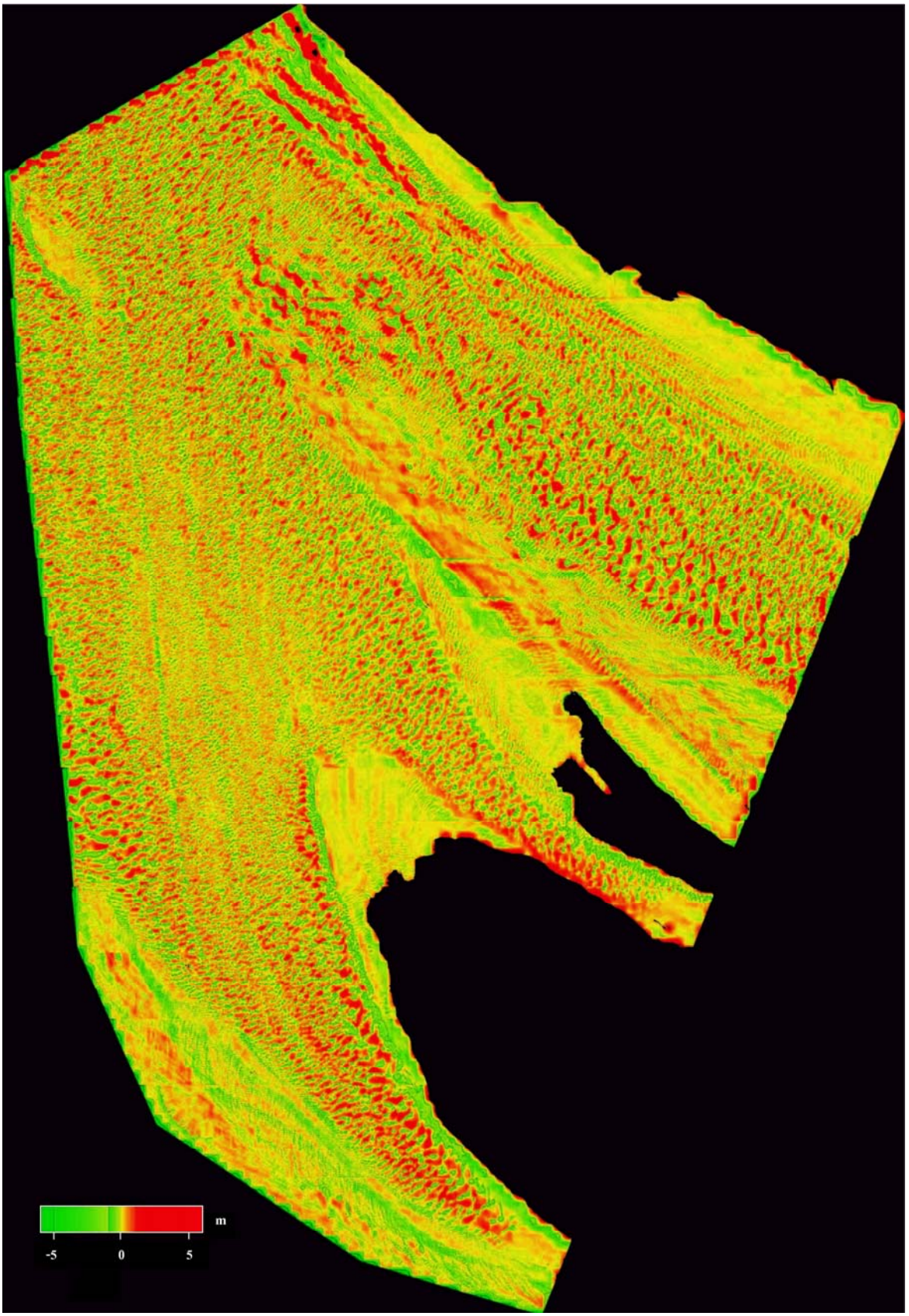
Appendix B: Verkregen reliëf- en vectorkaarten



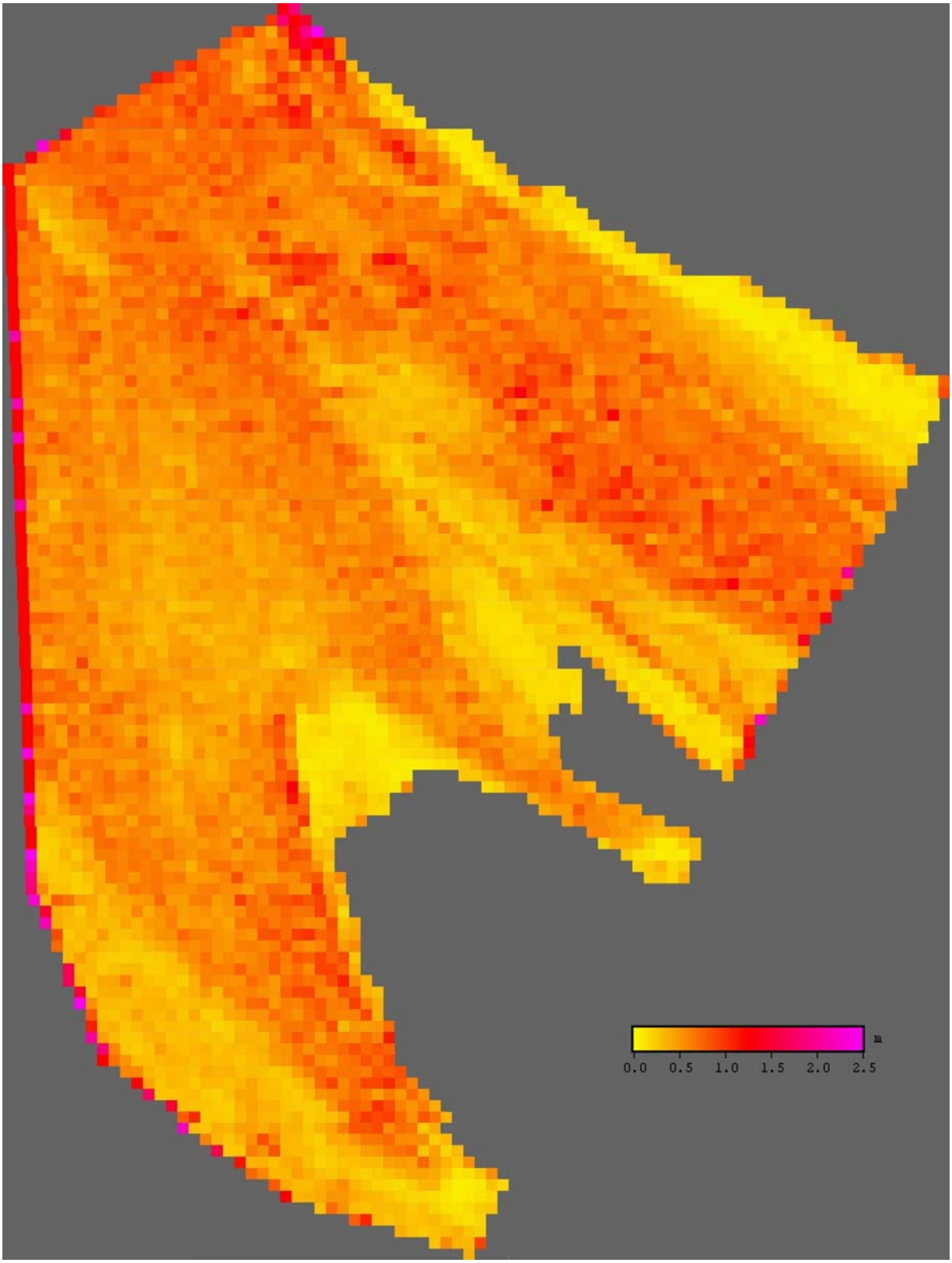
Figuur B-1: Dieptekaart van het meetgebied (4969 bij 3308 meter)



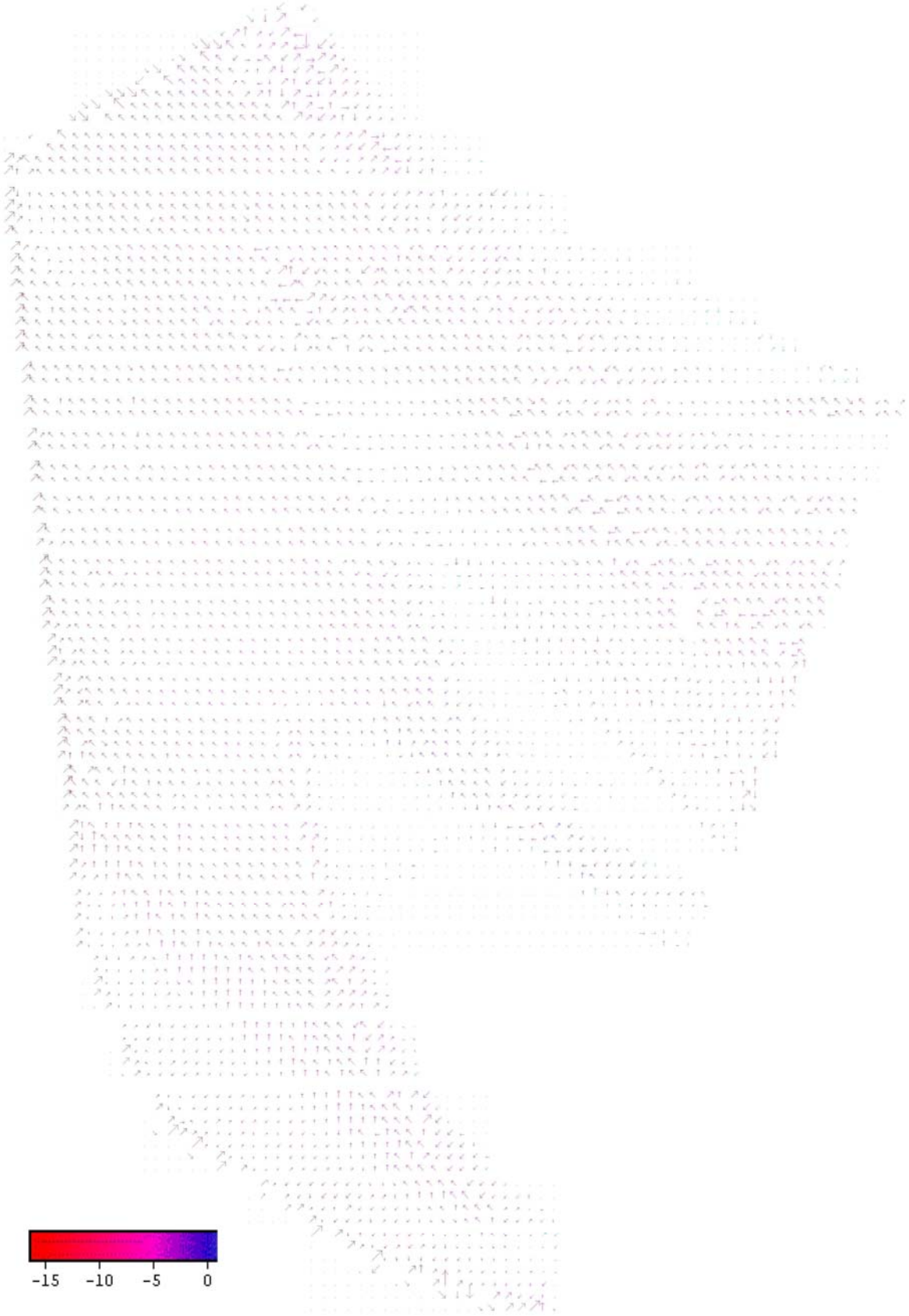
Figuur B-2: Gecompenseerde dieptekaart van het meetgebied (4969 bij 3308 meter)



Figuur B-3: Geïnverteerde gecompenseerde dieptekaart van het meetgebied (4969 bij 3308 meter)

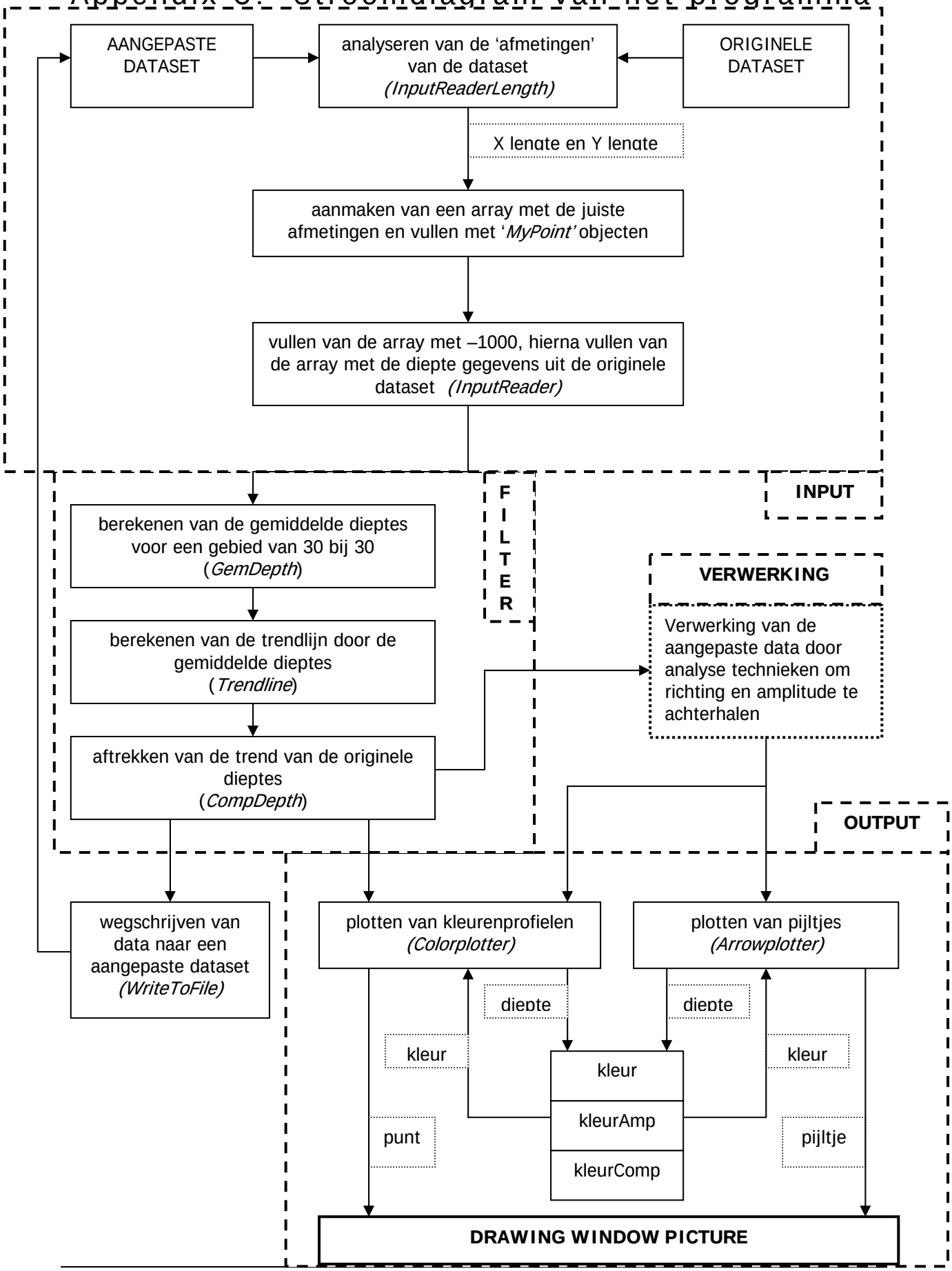


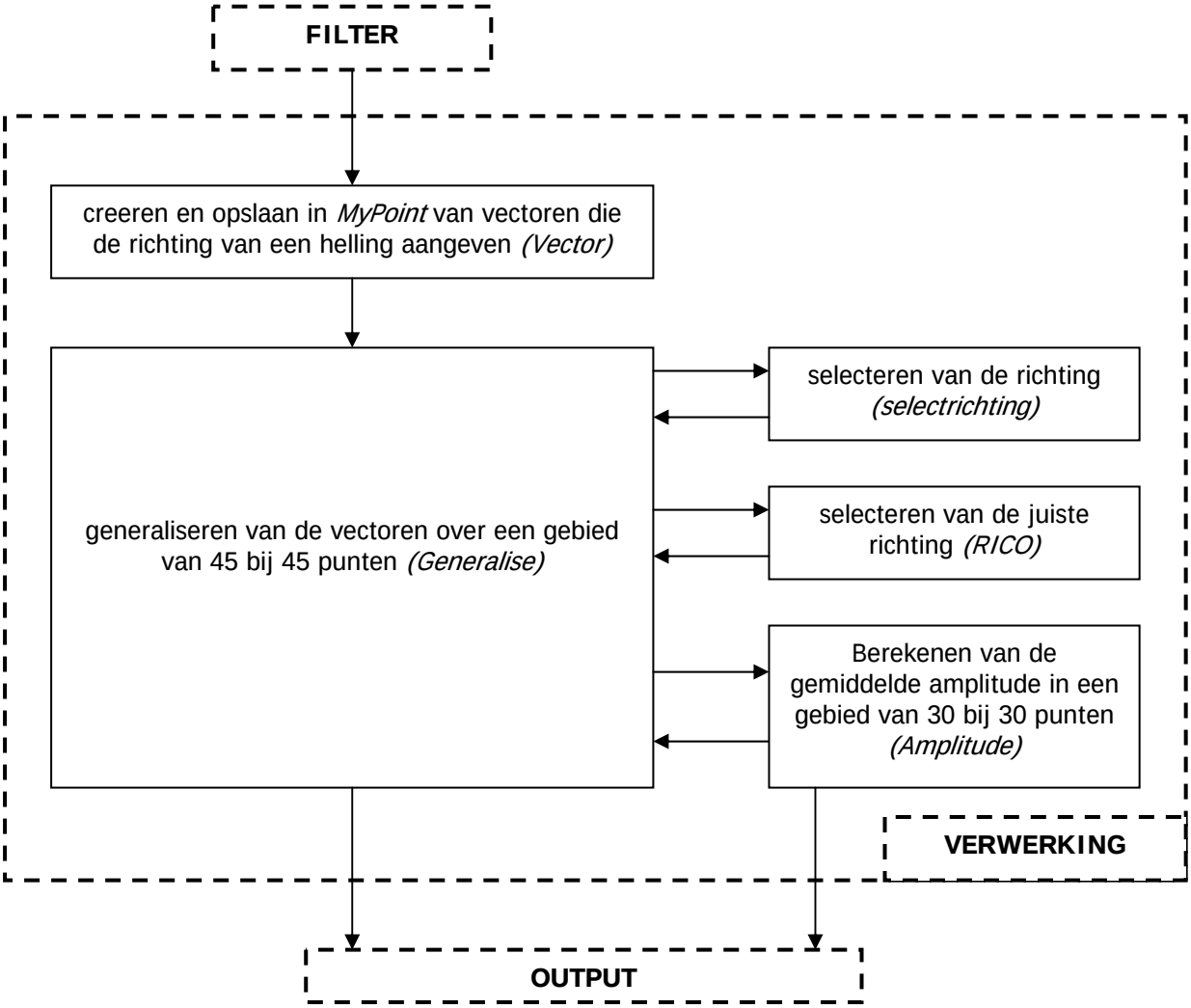
Figuur B-4: Amplitudekaart van het meetgebied (4969 bij 3308 meter)



Figuur B-5: Amplitudekaart van het meetgebied (4969 bij 3308 meter)

Appendix C: Stroomdiagram van het programma




FILTER
creeren en opslaan in *MyPoint* van vectoren die de richting van een helling aangeven *(Vector)*
generaliseren van de vectoren over een gebied van 45 bij 45 punten *(Generalise)*
selecteren van de richting *(selectrichting)*
selecteren van de juiste richting *(RICO)*
Berekenen van de gemiddelde amplitude in een gebied van 30 bij 30 punten *(Amplitude)*
VERWERKING
OUTPUT

Appendix D:

Aanroepprogramma

```
import java.util.*;
import java.awt.Color;
import java.io.*;

public class aanroep
{
  public static void main(String [] args)
            {
                        String fileName = "../Java/3850-4000.pts"; //input bestand
                        String fileNameOutput = "output.pts"; //output bestand
                        int Xmin = 0; //minimale x voor samplecreator
                        int Xmax = 50; //maximale x voor samplecreator
                        int Ymin = 0; //minimale y voor samplecreator
                        int Ymax = 50; //maximale y voor samplecreator
                        int Straal = 30; //straal voor amplitudes moet >15
                        boolean met_kaartje = false; //als true print diepteplaatje achter gegeneraliseerde pijltjes
    int resolutie = 45;  //resolutie voor generaliseren
    int zoomfactor = 1;  //zoomfactor voor pijltjes plotten
    //inlezen begint hier
    double gegevens[] = Methodes.inputReaderLength(fileName);
    int xLength = (int)gegevens[0];
    int yLength = (int)gegevens[1];
    double MinX = gegevens[2];
    double MinY = gegevens[3];
    MyPoint MyArray[][] = new MyPoint[xLength+1][yLength+1];
    for(int x=0; x<(xLength+1); x++){
      for(int y=0; y<(yLength+1); y++){
        MyArray [x][y] = new MyPoint();
      }
    }
    MyArray = Methodes.inputReader(fileName, MyArray, xLength, yLength, MinX, MinY);
    //inlezen eindigt hier
    //MyArray = Methodes.sample(MyArray,Xmin,Xmax,Ymin,Ymax); //creeert kleinere delen
    //Methodes.AmplitudeTest(MyArray, Straal, xLength, yLength);

    MyArray = Methodes.GemDiepte(MyArray,xLength, yLength,30);
    MyArray = Methodes.TrendLine(MyArray,xLength, yLength,30);
    MyArray = Methodes.CompDepth(MyArray,xLength, yLength);
    MyArray = Methodes.Vector(MyArray, xLength, yLength); //creeeren en opslaan van kleine vectoren
    MyArray = Methodes.Generalise(MyArray, xLength, yLength, resolutie); //creeeren en opslaan van grote vectoren
inclusief juiste richting
    Methodes.DrawVector(MyArray,xLength,yLength,true, met_kaartje, fileName, resolutie, zoomfactor); //printen van
pijltjes


    //Methodes.WriteToFile(MyArray,2, xL-ength, yLength, fileNameOutput); //getal kan 0 worden voor schrijven van
dieptes, 1 voor gemiddelde dieptes en 2 voor gecompenseerde dieptes
    //Methodes.colorplot(MyArray,2, xLength,yLength); //plotter



  }
}
```

Object class

```
public class MyPoint
{
public double depth;
public double Direction;
public double Length;
public double AvX;
public double AvY;
public double alpha;
public double AvDepth;
public double CompDepth;
public double Amplitude;
public boolean isAvPoint;


 public void MyPoint()
 {
  Direction = 0;
  Length = 0;
 }

 public void SetDepth(double diepte)
 {
  depth = diepte;
 }

 public void SetVector(double richting, double lengte)
 {
  Direction = richting;
  Length = lengte;
 }

 public double GetDepth()
 {
  return depth;
 }
 public void SetAverageVector(double gemx, double gemy, double alfa, boolean puntje)
 {
  AvX = gemx;
  AvY = gemy;
  alpha = alfa;
  isAvPoint = puntje;
 }
 public void SetDir(double dir, double amp)
 {
  if(dir<0)
  {
    alpha += Math.PI;
  }
  Amplitude = amp;
 }
 public void SetAverageDepth(double gemDepth)
 {
  AvDepth = gemDepth;
 }
 public double GetAverageDepth()
 {
  return AvDepth;
 }
 public void SetCompDepth(double gemDepth)
 {
  CompDepth = gemDepth;
 }
 public double GetCompDepth()
 {
  return CompDepth;
 }
}
```

Methodes

```
import java.util.*;
import java.awt.Color;
import java.io.*;
import element.*;
public class Methodes
{
  public static double[] inputReaderLength(String fileName)
            {
                        double gegevens[] = new double[4];

    double tempMinX = 9999999.0;
                        double tempMaxX = 0.0;
    double tempMinY = 9999999.0;
                        double tempMaxY = 0.0;
                        double depthMax = -1000.0;
                        double depthMin = 10000.0;
    String inLine;
                        StringTokenizer st;

    try
    {
                        // set up buffered input stream
                        BufferedReader br = new BufferedReader(new FileReader(fileName));

                        // read and display three lines; this is just an example
                        // you will have to store the data into some array
                        //while(doorgaan)
                        while(true)
                        {
                          inLine = br.readLine();
                          if(inLine==null)break;

     st = new StringTokenizer(inLine); // here is were the string is

     double Xcoord = Double.parseDouble(st.nextToken()); //get the data item
     double Ycoord = Double.parseDouble(st.nextToken()); //get the data item
                          double Depth = Double.parseDouble(st.nextToken()); //get the data item

     if(tempMinX > Xcoord)
               {
      tempMinX=Xcoord;
     }
      if(tempMaxX < Xcoord)
               {
      tempMaxX=Xcoord;
     }
     if(tempMinY > Ycoord)
               {
      tempMinY=Ycoord;
     }
     if(tempMaxY < Ycoord)
               {
      tempMaxY=Ycoord;
     }
     if(depthMax < Depth)
               {
      depthMax = Depth;
     }
     if(depthMin > Depth)
               {
      depthMin = Depth;
     }
   }
   br.close();
   }

   catch (Exception e)
      {
```

```
        System.err.println ("Error writing to file");
      }

   double depthDiff = depthMax - depthMin;
   double xLength = (int)(tempMaxX-tempMinX);
   double yLength = (int)(tempMaxY-tempMinY);
   double MinX = tempMinX;
   double MinY = tempMinY;
   gegevens[0] = xLength;
   gegevens[1] = yLength;
   gegevens[2] = MinX;
   gegevens[3] = MinY;
   return gegevens;
 }


 public static MyPoint[][] inputReader(String fileName, MyPoint MyArray[][], int xLength, int yLength, double
tempMinX,double tempMinY)
          {
   try
   {
     for (int i=0;i<xLength;i++)
             {
                      for (int j=0;j<yLength;j++)
                      {
                               MyArray[i][j].SetDepth(-1000);
                      }
     }
     String inLine2;
                      StringTokenizer st2;
            // set up buffered input stream
                      BufferedReader br2 = new BufferedReader(new FileReader(fileName));
     // read and display three lines; this is just an example
                      // you will have to store the data into some array

                      while(true)
                      {
                        inLine2 = br2.readLine();
             if(inLine2==null)break;
       st2 = new StringTokenizer(inLine2); // here is were the string is

       double Xcoord = Double.parseDouble(st2.nextToken()); //get the data item
       double Ycoord = Double.parseDouble(st2.nextToken()); //get the data item
                        double heigth = Double.parseDouble(st2.nextToken()); //get the data item

       MyArray [(int)(Xcoord-tempMinX)][(int)(Ycoord-tempMinY)].SetDepth(heigth);
            //System.out.println(); // create newline
                      }
     br2.close();                                                                    // DON'T FORGET
THIS
    }
    catch (Exception e)
    {
      System.err.println ("Error reading file");
    }
    return MyArray;
  }


   public static MyPoint[][] Vector(MyPoint amypoint[][], int i, int j)
   {
      for(int k = 1; k < i; k++)
      {
         for(int l = 1; l < j; l++)
         {
            if(amypoint[k][l].CompDepth == -1000D)
               continue;
            double d = amypoint[k - 1][l + 1].CompDepth;
            double d1 = 1.0D;
            double d2 = Math.abs((amypoint[k - 1][l + 1].CompDepth - amypoint[k][l].CompDepth) / Math.sqrt(2D));
            if(amypoint[k - 1][l].CompDepth > d)
```

```
            {
                d = amypoint[k - 1][l].CompDepth;
                d1 = -4D;
                d2 = Math.abs(amypoint[k - 1][l].CompDepth - amypoint[k][l].CompDepth);
            }
            if(amypoint[k - 1][l - 1].CompDepth / Math.sqrt(2D) > d)
            {
                d = amypoint[k - 1][l - 1].CompDepth;
                d1 = -3D;
                d2 = Math.abs((amypoint[k - 1][l - 1].CompDepth - amypoint[k][l].CompDepth) / Math.sqrt(2D));
            }
            if(amypoint[k][l - 1].CompDepth > d)
            {
                d = amypoint[k][l - 1].CompDepth;
                d1 = -2D;
                d2 = Math.abs(amypoint[k][l - 1].CompDepth - amypoint[k][l].CompDepth);
            }
            if(amypoint[k + 1][l - 1].CompDepth / Math.sqrt(2D) > d)
            {
                d = amypoint[k + 1][l - 1].CompDepth;
                d1 = -1D;
                d2 = Math.abs((amypoint[k + 1][l - 1].CompDepth - amypoint[k][l].CompDepth) / Math.sqrt(2D));
            }
            if(amypoint[k + 1][l].CompDepth > d)
            {
                d = amypoint[k + 1][l].CompDepth;
                d1 = 4D;
                d2 = Math.abs(amypoint[k + 1][l].CompDepth - amypoint[k][l].CompDepth);
            }
            if(amypoint[k + 1][l + 1].CompDepth / Math.sqrt(2D) > d)
            {
                d = amypoint[k + 1][l + 1].CompDepth;
                d1 = 3D;
                d2 = Math.abs((amypoint[k + 1][l + 1].CompDepth - amypoint[k][l].CompDepth) / Math.sqrt(2D));
            }
            if(amypoint[k][l + 1].CompDepth > d)
            {
                d = amypoint[k][l + 1].CompDepth;
                d1 = 2D;
                d2 = Math.abs(amypoint[k][l + 1].CompDepth - amypoint[k][l].CompDepth);
            }
            if(d < amypoint[k][l].CompDepth)
            {
                d1 = 0.0D;
                d2 = 0.0D;
            }
            amypoint[k][l].SetVector(d1, d2);
        }

    }

    return amypoint;
}
public static MyPoint[][] Generalise(MyPoint amypoint[][], int i, int j, int resolution)
{
    double d = 0.0D;
    double d2 = 0.0D;
    double d4 = 0.0D;
    double d6 = 0.0D;
    double d7 = 0.0D;
    double d8 = 0.0D;
    double d9 = 0.0D;
    for(int i1 = 0; i1 < (i - resolution) + 1; i1 += resolution)
    {
        for(int j1 = 0; j1 < (j - resolution) + 1; j1 += resolution)
        {
            for(int k1 = 0; k1 < resolution; k1++)
            {
                for(int l1 = 0; l1 < resolution; l1++)
                {
                    if(Math.abs(amypoint[i1 + k1][j1 + l1].Direction) == 1.0D)
```

```
                    d6 += amypoint[i1 + k1][j1 + l1].Length;
                if(Math.abs(amypoint[i1 + k1][j1 + l1].Direction) == 2D)
                    d7 += amypoint[i1 + k1][j1 + l1].Length;
                if(Math.abs(amypoint[i1 + k1][j1 + l1].Direction) == 3D)
                    d8 += amypoint[i1 + k1][j1 + l1].Length;
                if(Math.abs(amypoint[i1 + k1][j1 + l1].Direction) == 4D)
                    d9 += amypoint[i1 + k1][j1 + l1].Length;
            }

        }

        int k = 1;
        double d1 = -0.5D * Math.sqrt(2D) * d6;
        double d3 = 0.5D * Math.sqrt(2D) * d6;
        if(d7 > d6)
        {
            k = 2;
            d1 = 0.0D;
            d3 = d7;
        }
        if(d8 > d7 && d8 > d6)
        {
            k = 3;
            d1 = 0.5D * Math.sqrt(2D) * d8;
            d3 = 0.5D * Math.sqrt(2D) * d8;
        }
        if(d9 > d8 && d9 > d7 && d9 > d6)
        {
            k = 4;
            d1 = d9;
            d3 = 0.0D;
        }
        d1 = (d1 / (double)resolution) * 2D;
        d3 = (d3 / (double)resolution) * 2D;
        double d5 = selectrichting(k);
        amypoint[i1 + resolution / 2][j1 + resolution / 2].SetAverageVector(d1, d3, d5, true);
        amypoint[i1 + resolution / 2][j1 + resolution / 2].SetDir(RiCo(amypoint, k, i1 + resolution / 2, j1 + resolution /
2, resolution / 2), Amplitude(amypoint, k, i1 + resolution / 2, j1 + resolution / 2, resolution / 2));

        d1 = 0.0D;
        d3 = 0.0D;
        d6 = 0.0D;
        d7 = 0.0D;
        d8 = 0.0D;
        d9 = 0.0D;
      }

    }

    return amypoint;
  }

  public static double selectrichting (double richting)
  {
   double alfa=0;
            if (richting==1) alfa=(3*Math.PI)/4;
   if (richting==2) alfa=(Math.PI)/2;
   if (richting==3) alfa=(Math.PI)/4;
   if (richting==4) alfa=0;
            if (richting==-1) alfa=(7*Math.PI)/4;
            if (richting==-2) alfa=(3*Math.PI)/2;
            if (richting==-3) alfa=(5*Math.PI)/4;
   if (richting==-4) alfa=Math.PI;
            return alfa;
  }

    public static void DrawVector(MyPoint amypoint[][], int i, int j, boolean flag, boolean flag1, String s, int resolutie, int
k2)
  {
    int counter=-1;
    int k;
```

```
int l;
double d;
double d1;
int i1;
int j1;
int i2 = 0;    //starting at
int j2 = i;   //stopping at i
DrawingWindow drawingwindow = new DrawingWindow(1500, 800, s);
float f = 0.0F;
if(flag1)
{
   for(int l2 = i2; l2 < j2 - k2; l2 += k2)
   {
      for(int j3 = 0; j3 < j - k2; j3 += k2)
      {
         drawingwindow.setForeground(KleurComp(amypoint[l2][j3].CompDepth));
         Pt pt = new Pt((l2 - i2) / k2, (j - j3) / k2);
         drawingwindow.draw(pt);
      }

   }

}
for (int jopie=0; jopie<=2400;jopie+=1200)
{
        counter++;

for(int i3 = jopie; i3 < jopie+1200; i3++)
{
   for(int k3 = 0; k3 < j; k3++)
   {
      if(i3>i) break;
      if(flag)
      {
         k = (i3 - i2) / k2;    //x0
         l = k3 / k2;           //y0
         int k1 = 14 / k2;        //magFactor
         if(!amypoint[i3][k3].isAvPoint)
            continue;
         i1 = (int)(amypoint[i3][k3].Amplitude* Math.cos(amypoint[i3][k3].alpha) * (double)k1);
         if (amypoint[i3][k3].alpha>=Math.PI) i1 = -i1;
         j1 = Math.abs((int)(amypoint[i3][k3].Amplitude* Math.sin(amypoint[i3][k3].alpha) * (double)k1));
         d1 = amypoint[i3][k3].alpha;       //alfa
         d = amypoint[i3][k3].Amplitude*k1;          //amplitude
      } else
      {
         k = i3 * 10;
         l = k3 * 10;
         int l1 = 50;
         d = amypoint[i3][k3].Length * (double)l1;
         d1 = selectrichting(amypoint[i3][k3].Direction);
         i1 = (int)(d * Math.cos(d1));
         j1 = (int)(d * Math.sin(d1));
      }
      if(d > resolutie*1.2)
         continue;
      drawingwindow.setForeground(Kleur(amypoint[i3][k3].depth));
      Line line = new Line(k-jopie, j / k2 - l+ counter*(j+10), k + i1-jopie, j / k2 - l - j1+ counter*(j+10));

      if(amypoint[i3][k3].alpha == 0.0D)
      {
         Line line1 = new Line(k + i1-jopie, (j / k2 - l) + j1+ counter*(j+10), k + (int)(0.80000000000000004D * d *
Math.cos(d1 + 0.29999999999999999D))-jopie, (j / k2 - l) + (int)(0.80000000000000004D * d * Math.sin(d1 +
0.29999999999999999D))+ counter*(j+10));
         Line line5 = new Line(k + i1-jopie, (j / k2 - l) + j1+ counter*(j+10), k + (int)(0.80000000000000004D * d *
Math.cos(d1 + 0.29999999999999999D))-jopie, j / k2 - l - (int)(0.80000000000000004D * d * Math.sin(d1 +
0.29999999999999999D))+ counter*(j+10));
         drawingwindow.draw(line1);
         drawingwindow.draw(line5);
      }
```

```
            else if(amypoint[i3][k3].alpha >= 0.78539816339744828D && amypoint[i3][k3].alpha <
3.1415926535897931D)
            {
               Line line2 = new Line(k + i1-jopie, j / k2 - l - j1+ counter*(j+10), k + (int)(0.80000000000000004D * d *
Math.cos(d1 + 0.29999999999999999D))-jopie, j / k2 - l - (int)(0.80000000000000004D * d * Math.sin(d1 +
0.29999999999999999D))+ counter*(j+10));
               Line line6 = new Line(k + i1-jopie, j / k2 - l - j1+ counter*(j+10), k + (int)(0.80000000000000004D * d *
Math.cos(d1 - 0.29999999999999999D))-jopie, j / k2 - l - (int)(0.80000000000000004D * d * Math.sin(d1 -
0.29999999999999999D))+ counter*(j+10));
               drawingwindow.draw(line2);
               drawingwindow.draw(line6);
            }
            else if(amypoint[i3][k3].alpha == 3.1415926535897931D)
            {
               Line line3 = new Line(k-jopie, j / k2 - l+ counter*(j+10), k + i1 + (int)(0.80000000000000004D * d *
Math.cos(d1 + 0.29999999999999999D))-jopie, (j / k2 - l - j1) + (int)(0.80000000000000004D * d * Math.sin(d1 +
0.29999999999999999D))+ counter*(j+10));
               Line line7 = new Line(k-jopie, j / k2 - l+ counter*(j+10), k + i1 + (int)(0.80000000000000004D * d *
Math.cos(d1 - 0.29999999999999999D))-jopie, (j / k2 - l - j1) + (int)(0.80000000000000004D * d * Math.sin(d1 -
0.29999999999999999D))+ counter*(j+10));
               drawingwindow.draw(line3);
               drawingwindow.draw(line7);
            } else
            if(amypoint[i3][k3].alpha >= 3.9269908169872414D && amypoint[i3][k3].alpha < 6.2831853071795862D)
            {
               Line line4 = new Line(k-jopie, j / k2 - l+ counter*(j+10), k + i1 + (int)(0.80000000000000004D * d *
Math.cos(d1 + 0.29999999999999999D))-jopie, j / k2 - l - j1 - (int)(0.80000000000000004D * d * Math.sin(d1 +
0.29999999999999999D))+ counter*(j+10));
               Line line8 = new Line(k-jopie, j / k2 - l+ counter*(j+10), k + i1 + (int)(0.80000000000000004D * d *
Math.cos(d1 - 0.29999999999999999D))-jopie, j / k2 - l - j1 - (int)(0.80000000000000004D * d * Math.sin(d1 -
0.29999999999999999D))+ counter*(j+10));
               drawingwindow.draw(line4);
               drawingwindow.draw(line8);
            }
            drawingwindow.draw(line);
         }

      }
    }
  }


 public static double RiCo(MyPoint ari[][],int richting,int xco,int yco, int straal)
  // pre : richting!=0 en lokatie en grote van scanstraal
  // post: een gemidelde amplitude voor de locatie
  {

    int x=xco;
    int y=yco;
    int xlength=ari.length-1;
    int ylength=ari[1].length-1;

    double ricoplus=0;
    int tellerplus=0;
    double ricomin=0;
    int tellermin=0;

    // Scan in noord-westelijke richting
    if (richting==1)
    {
      x=xco-straal;
      y=yco-straal;

      while(true)
      {
       if (x<xco+straal){
          x=x+10;}
       else if (y<yco+straal){
          y=y+10;}
       else
          break;
```

```
        int i=x;
        int j=y;

        while((i>xco-straal) && (j<yco+straal))
        {
          if ((i-1)<0 || (i+1)>xlength || j<0 || (j+1)>ylength){}// Doe niets
          else{
            if ((ari[i][j].CompDepth > (-100))&&(ari[i][j].CompDepth < 100)&&(ari[i-1][j+1].CompDepth > (-100))&&(ari[i-
1][j+1].CompDepth < 100))
            {
                    if (ari[i-1][j+1].CompDepth > ari[i][j].CompDepth){
                      ricoplus = ricoplus + ari[i-1][j+1].CompDepth - ari[i][j].CompDepth;
                tellerplus++;}
                    if (ari[i-1][j+1].CompDepth < ari[i][j].CompDepth){
                      ricomin = ricomin + ari[i][j].CompDepth - ari[i-1][j+1].CompDepth;
                tellermin++;}
            }
            }
            j++;
            i--;
          }
        }

        if (ricoplus/(tellerplus) > ricomin/(tellermin)){
          return 1;}
        else{
          return -1;}
      }

      // Scan in noordelijke richting
      else if (richting==2)
      {
        x=xco-straal;
        y=yco-straal;

        while(x<xco+straal)
        {
          x=x+10;

          int i=x;
          int j=y;

          while(j<yco+straal)
          {
            if ((i-1)<0 || (i+1)>xlength || j<0 || (j+1)>ylength){}// Doe niets
            else{
              if ((ari[i][j].CompDepth > (-100))||(ari[i][j].CompDepth < 100)||(ari[i][j+1].CompDepth > (-
100))||(ari[i][j+1].CompDepth < 100))
              {
                      if (ari[i][j+1].CompDepth > ari[i][j].CompDepth){
                        ricoplus = ricoplus + ari[i][j+1].CompDepth - ari[i][j].CompDepth;
                tellerplus++;}
                      if (ari[i][j+1].CompDepth < ari[i][j].CompDepth){
                        ricomin = ricomin + ari[i][j].CompDepth - ari[i][j+1].CompDepth;
                tellermin++;}
              }
              }
              j++;
          }
        }
        if  (ricoplus/(tellerplus) > ricomin/(tellermin)){
          return 2;}
        else{
          return -2;}
      }

      // Scan in noord-oostelijke richting
      else if (richting==3)
      {
```

```
        x=xco+straal;
        y=yco-straal;

        while(true)
        {
          if (x>xco-straal){
            x=x-10;}
          else if (y<yco+straal){
            y=y+10;}
          else
            break;

          int i=x;
          int j=y;

          while((i>xco-straal) && (j<yco+straal))
          {
            if ((i-1)<0 || (i+1)>xlength || j<0 || (j+1)>ylength){}// Doe niets
            else{
              if ((ari[i][j].CompDepth > (-100))||(ari[i][j].CompDepth < 100)||(ari[i+1][j+1].CompDepth > (-
100))||(ari[i+1][j+1].CompDepth < 100))
              {
                        if (ari[i+1][j+1].CompDepth > ari[i][j].CompDepth){
                          ricoplus = ricoplus + ari[i+1][j+1].CompDepth - ari[i][j].CompDepth;
                    tellerplus++;}
                        if (ari[i+1][j+1].CompDepth < ari[i][j].CompDepth){
                          ricomin = ricomin + ari[i][j].CompDepth - ari[i+1][j+1].CompDepth;
                    tellermin++;}
              }
              }
              i++;
              j++;
            }
          }
          if (ricoplus/(tellerplus) > ricomin/(tellermin)){
            return 3;}
          else{
            return -3;}
        }

        // Scan in oostelijke richting
        else if (richting==4)
        {
          x=xco-straal;
          y=yco-straal;

          while(y<yco+straal)
          {
            y=y+10;

            int i=x;
            int j=y;

            while(i<xco+straal)
            {
              if ((i-1)<0 || (i+1)>xlength || j<0 || (j+1)>ylength){}// Doe niets
              else{
                if ((ari[i][j].CompDepth > (-100))||(ari[i][j].CompDepth < 100)||(ari[i+1][j].CompDepth > (-
100))||(ari[i+1][j].CompDepth < 100))
                {
                        if (ari[i+1][j].CompDepth > ari[i][j].CompDepth){
                          ricoplus = ricoplus + ari[i+1][j].CompDepth - ari[i][j].CompDepth;
                    tellerplus++;}
                        if (ari[i+1][j].CompDepth < ari[i][j].CompDepth){
                          ricomin = ricomin + ari[i][j].CompDepth - ari[i+1][j].CompDepth;
                    tellermin++;}
                }
                }
                i++;
            }
          }
```

```
      if (ricoplus/(tellerplus) > ricomin/(tellermin)){
         return 4;}
      else{
         return -4;}
   }
   else{return 0;}
}


public static void WriteToFile(MyPoint ar[][],int choice, int xLength, int yLength, String FileName)
 //pre: get's an array with points, it's width and length and a FileName
 //post: writes the array to a .pts file with as filename FileName
 {
   FileOutputStream output;
   PrintStream textfile;

    try
    {
          output = new FileOutputStream(FileName);
     textfile = new PrintStream(output);

     for (int i=0;i<=xLength;i++)
     {
       for (int j=0;j<=yLength;j++)
       {
         //writer for Depths
         if(choice ==0)
         {
           textfile.print((double)i+"\t"+(double)j+"\t"+ar[i][j].depth+"\t");
           textfile.println();
         }

         //writer for AverageDepths
         if(choice ==1)
         {
           textfile.print((double)i+"\t"+(double)j+"\t"+ar[i][j].AvDepth+"\t");
           textfile.println();
         }

         //writer for Compensated Depths
         if(choice ==2)
         {
           textfile.print((double)i+"\t"+(double)j+"\t"+ar[i][j].CompDepth+"\t");
           textfile.println();
         }

       }
     }
     textfile.close();
    }

    catch (Exception e)
    {
         System.err.println ("Error writing to file");
    }
}

public static MyPoint[][] GemDiepte(MyPoint ar[][],int xlength, int ylength, int resolutie)
 {
           int counter =0;
   double diepte=0;

   for(int k = 0; k <=xlength; k+=resolutie)
   {
         for(int l = 0; l <=ylength; l+=resolutie)
         {
                  if (ar[k][l].depth < -100)continue;
      for(int i = 0; i<resolutie; i++)
      {
         if ((k+i)>xlength) break;
       if (ar[k+i][l].depth < -100)continue;
```

```
            for(int j = 0; j<resolutie; j++)
        {

         if ((l+j)> ylength) break;
         if  (ar[k+i][l+j].depth < -100)continue;
         diepte += ar[k+i][l+j].depth;
         counter++;
         }
       }
        diepte = diepte/counter;
        counter = 0;

      for(int i = 0; i<resolutie; i++)
      {
          if ((k+i)>xlength) continue;
            for(int j = 0; j<resolutie; j++)
       {
         if ((l+j)> ylength) continue;
        ar[i+k][j+l].SetAverageDepth(diepte);
         }
       }
      diepte = 0;
    }
   }
  return ar;
}

public static MyPoint[][] TrendLine(MyPoint ar[][],int xlength, int ylength, int resolutie)
{
          int counter =0;
  double diepteDiff=0;
  int DifferenceX=0;
  int DifferenceY=0;

  for(int k = 0; k <= xlength; k+=resolutie)
  {
    for(int l = 0; l <= ylength; l+=resolutie)
        {
     DifferenceX = xlength-k;
     DifferenceY = ylength-l;
     if (DifferenceX<resolutie)
     {
      if (DifferenceY>resolutie)
      {
       diepteDiff = (ar[k][l].AvDepth - ar[k+DifferenceX][l].AvDepth)/DifferenceX;
       for(int j = 0; j<resolutie; j++)
       {
        for(int i = 0; i<DifferenceX; i++)
        {
                ar[i+k][j+l].SetAverageDepth(ar[i+k][j+l].AvDepth - diepteDiff*i);
        }
       }
      }
      else
      {
       diepteDiff = (ar[k][l].AvDepth - ar[k+DifferenceX][l].AvDepth)/DifferenceX;
       for(int j = 0; j<DifferenceY; j++)
       {
        for(int i = 0; i<DifferenceX; i++)
        {
         ar[i+k][j+l].SetAverageDepth(ar[i+k][j+l].AvDepth - diepteDiff*i);
        }
       }
      }
     }
     if (DifferenceY<resolutie && DifferenceX>resolutie)
     {
        diepteDiff = (ar[k][l].AvDepth - ar[k+resolutie][l].AvDepth)/resolutie;
        for(int j = 0; j<DifferenceY; j++)
        {
         for(int i = 0; i<resolutie; i++)
```

```
            {
              ar[i+k][j+l].SetAverageDepth(ar[i+k][j+l].AvDepth - diepteDiff*i);
            }
           }
        }
        if (DifferenceY>resolutie && DifferenceX>resolutie)
        {
          diepteDiff = (ar[k][l].AvDepth - ar[k+resolutie][l].AvDepth)/resolutie;
          for(int j = 0; j<resolutie; j++)
          {
            for(int i = 0; i<resolutie; i++)
            {
              ar[i+k][j+l].SetAverageDepth(ar[i+k][j+l].AvDepth - diepteDiff*i);
            }
          }
        }
     }
    }

    diepteDiff=0;
    for(int l = 0; l <= ylength-resolutie; l+=resolutie)
    {
      for(int k = 0; k <= xlength; k++)
           {
        DifferenceY = ylength-l;
        if(DifferenceY < resolutie)
        {
          diepteDiff = (ar[k][l].AvDepth - ar[k][l+DifferenceY].AvDepth)/DifferenceY;
          for(int j = 0; j<DifferenceY; j++)
          {
            if ((l+j)>ylength) break;
            ar[k][j+l].SetAverageDepth(ar[k][j+l].AvDepth - diepteDiff*j);
          }
        }
        else
        {
          diepteDiff = (ar[k][l].AvDepth - ar[k][l+resolutie].AvDepth)/resolutie;
          for(int j = 0; j<resolutie; j++)
          {
            if ((l+j)>ylength) break;
            ar[k][j+l].SetAverageDepth(ar[k][j+l].AvDepth - diepteDiff*j);
          }
        }
      }
    }
   return ar;
  }

 public static MyPoint[][] CompDepth (MyPoint ar[][],int xlength, int ylength)
  {
    for(int j = 0; j<=ylength; j++)
    {
      for(int i = 0; i<=xlength; i++)
      {
       if (ar[i][j].depth < -100)ar[i][j].CompDepth = -1000;
       else ar[i][j].SetCompDepth(ar[i][j].depth - ar[i][j].AvDepth);
      }
    }
      return ar;
   }
   public static void colorplot(MyPoint a[][],int choice, int xlength, int ylength)
 //pre: get's an array with points, it's width and length and a choice
 //post: plots a color picture of the array
 {
  int counter= -1;
  DrawingWindow d = new DrawingWindow(1220,(3*ylength)+40);

  for (int k=0; k<=2400;k=k+1200)
  {
            counter++;
```

```
    for (int i=k; i<(k+1200); i++)
            {

     for (int j=0; j<ylength; j++)
                        {
        if(i>xlength) break;
        if (a[i][j].GetDepth()<-100)
        {
           d.setForeground(Color.darkGray);

        }
        else
        {
            //plotter for Depths
            if(choice==0)
            {
                        //if ((k+i) > xlength) break;
              d.setForeground(Kleur(a[i][j].depth));
            }
            //plotter for Average depths
            if(choice==1)
            {
                        //if ((k+i) > xlength) break;
              d.setForeground(Kleur(a[i][j].AvDepth));
            }
            //plotter for Compensated depths
            if(choice==2)
            {
                        //if ((k+i) > xlength) break;
              d.setForeground(KleurComp(a[i][j].CompDepth));
            }
        }

        Pt pixel = new Pt((i-k),(ylength-j)+ counter*(ylength+10));
        d.draw(pixel);
      }
    }
  }
}

 public static Color KleurComp(double diepte)
 {
    int kleur1;
    int kleur2;
    int kleur3;

    //DrawingWindow d = new DrawingWindow(600,600);


      if (diepte<-100)
      {
        kleur1=100;
        kleur2=100;
        kleur3=100;
      }
      else if (diepte<-5)
      {
        kleur1=0;
        kleur2=0;
        kleur3=250;
      }
      else if (diepte<-1)
      {
            kleur1=0;
        kleur2=(int)(30*(diepte)+250);
            kleur3=250;
      }
      else if (diepte<0)
      {
            kleur1=0;
        kleur2=(int)(100*(diepte)+250);
```

```
      kleur3=250;
    }
    else if (diepte<1)
    {
      kleur1=0;
      kleur2=250;
      kleur3=(int)(250-(diepte*100));
    }
    else if (diepte<5)
    {
      kleur1=0;
      kleur2=250;
      kleur3=(int)(150-(diepte*30));
    }
    else
    {
      kleur1=100;
      kleur2=100;
      kleur3=100;
    }

    return new Color(kleur1,kleur2,kleur3);
}

public static Color Kleur(double diepte)
{
  int kleur1;
  int kleur2;
  int kleur3;

  //DrawingWindow d = new DrawingWindow(600,600);


    if (diepte<-100)
    {
      kleur1=100;
      kleur2=100;
      kleur3=100;
    }
    else if (diepte<-15)
    {
      kleur1=0;
      kleur2=0;
      kleur3=250;
    }
    else if (diepte<-5)
    {
        kleur1=0;
      kleur2=(int)(20*(15+(diepte)));
        kleur3=250;
    }
    else if (diepte<0)
    {
        kleur1=0;
      kleur2=200;
      kleur3=(int)(250-40*(5+(diepte)));
    }
    else if (diepte<1)
    {
      kleur1=(int)(40*diepte);
      kleur2=200;
      kleur3=50;
    }
    else if (diepte<25)
    {
      kleur1=200;
      kleur2=200;
      kleur3=50;
    }
    else
    {
```

```
        kleur1=100;
        kleur2=100;
        kleur3=100;
      }

      return new Color(kleur1,kleur2,kleur3);
  }
 public static MyPoint[][] sample(MyPoint a[][],int Xmin,int Xmax,int Ymin,int Ymax)
   // Pre: array MyArray
   // Post: sample array
   {
            MyPoint MyArray[][] = new MyPoint[Xmax-Xmin][Ymax-Ymin];

  for (int i=Xmin;i<Xmax;i++)
            {
                     for (int x=Ymin;x<Ymax;x++)
    {
      MyArray[i-Xmin][x-Ymin] = a[i][x];
                     }
  }
  return MyArray;
 }

  public static void AmplitudeTest(MyPoint MyArray[][],int straal, int xLength, int yLength)
  {
   DrawingWindow d = new DrawingWindow(1200,yLength);
   DrawingWindow e = new DrawingWindow(1200,yLength);
   DrawingWindow f = new DrawingWindow(1200,yLength);
   double amplitude;
   for (int k=straal; k<xLength; k=k+2*straal)
   {
     for (int l=straal; l<yLength; l=l+2*straal)
     {
       amplitude=0;
       for (int i=1; i<5; i++)
       {
            if(amplitude<Amplitude(MyArray,i,k,l,straal));{                    // In een loop wanneer
           amplitude=Amplitude(MyArray,i,k,l,straal);}                        // scanrichting onbekend
       }
       if (k<1200)
       {
         d.setForeground(KleurAmp(amplitude));                                 // Het Plotten van kleur
         Rect vierkant = new Rect(k,yLength-l,straal);
              d.fill(vierkant);
            }
            if (k>1100)
       {
         e.setForeground(KleurAmp(amplitude));                                 // Het Plotten van kleur
         Rect vierkant = new Rect(k-1100,yLength-l,straal);
              e.fill(vierkant);
            }
       if (k>2200)
       {
         f.setForeground(KleurAmp(amplitude));                                 // Het Plotten van kleur
         Rect vierkant = new Rect(k-2200,yLength-l,straal);
              f.fill(vierkant);
            }
            //String t = Integer.toString((int)(10*Amplitude(MyArray,2,k,l,straal)));  // Het Plotten van tekst
            //d.draw(new Text(t,k,l));
     }
   }
  }

  public static double Amplitude(MyPoint ari[][],int richting,int xco,int yco, int straal)
  // pre : ricthting!=0 en lokatie en grote van scanstraal
  // post: een gemidelde amplitude voor de locatie
  {
    double amp=0;
    double hoogtemax=-100;
    double hoogtemin=100;
    int xlength=ari.length-1;
```

```
int ylength=ari[1].length-1;
int x=xco;
int y=yco;
int teller=0;

// Scan in noord-westelijke richting
if (richting==1)
{
  x=xco-straal;
  y=yco-straal;

  while(true)
  {
    if (x<xco+straal){
      x=x+10;}
    else if (y<yco+straal){
      y=y+10;}
    else
      break;

    int i=x;
    int j=y;

    while((i>xco-straal) && (j<yco+straal))
    {
      hoogtemax=-100;
      hoogtemin=100;
      for(int k=0; k<30; k++)
      {
        if ((i-k)<0 || (i-k)>xlength || (j+k)<0 || (j+k)>ylength){}// Doe niets
        else{
        if ((ari[i-k][j+k].CompDepth < hoogtemin) && (ari[i-k][j+k].CompDepth > (-100))){
          hoogtemin=ari[i-k][j+k].CompDepth;}
         if ((ari[i-k][j+k].CompDepth > hoogtemax) && (ari[i-k][j+k].CompDepth < (100))){
          hoogtemax=ari[i-k][j+k].CompDepth;}}
      }
      if(hoogtemax-hoogtemin!=-200){
        amp=amp+hoogtemax-hoogtemin;
        teller++;}
      i=i-10;
      j=j+10;
    }
  }
}

// Scan in noordelijke richting
else if (richting==2)
{
  x=xco-straal;
  y=yco-straal;

  while(x<xco+straal)
  {
    x=x+10;

    int i=x;
    int j=y;

    while(j<yco+straal)
    {
      hoogtemax=-100;
      hoogtemin=100;
      for(int k=0; k<30; k++)
      {
        if ((i)<0 || (i)>xlength || (j+k)<0 || (j+k)>ylength){}// Doe niets
        else{
        if ((ari[i][j+k].CompDepth < hoogtemin) && (ari[i][j+k].CompDepth > (-100))){
          hoogtemin=ari[i][j+k].CompDepth;}
         if ((ari[i][j+k].CompDepth > hoogtemax) && (ari[i][j+k].CompDepth < (100))){
          hoogtemax=ari[i][j+k].CompDepth;}}
      }
```

```
      if(hoogtemax-hoogtemin!=-200){
        amp=amp+hoogtemax-hoogtemin;
        teller++;}

      j=j+10;
    }

  }
}

// Scan in noord-oostelijke richting
else if (richting==3)
{
  x=xco+straal;
  y=yco-straal;

  while(true)
  {
    if (x>xco-straal){
      x=x-10;}
    else if (y<yco+straal){
      y=y+10;}
    else
      break;

    int i=x;
    int j=y;

    while((i>xco-straal) && (j<yco+straal))
    {
       hoogtemax=-100;
      hoogtemin=100;
      for(int k=0; k<30; k++)
      {
        if ((i+k)<0 || (i+k)>xlength || (j+k)<0 || (j+k)>ylength){}// Doe niets
        else{
        if ((ari[i+k][j+k].CompDepth < hoogtemin) && (ari[i+k][j+k].CompDepth > (-100))){
          hoogtemin=ari[i+k][j+k].CompDepth;}
        if ((ari[i+k][j+k].CompDepth > hoogtemax) && (ari[i+k][j+k].CompDepth < (100))){
          hoogtemax=ari[i+k][j+k].CompDepth;}}
      }
      if(hoogtemax-hoogtemin!=-200){
        amp=amp+hoogtemax-hoogtemin;
        teller++;}

      i=i+10;
      j=j+10;
    }

  }
}

// Scan in oostelijke richting
else if (richting==4)
{
  x=xco-straal;
  y=yco-straal;

  while(y<yco+straal)
  {
    y=y+10;

    int i=x;
    int j=y;

    while(i<xco+straal)
    {
      hoogtemax=-100;
      hoogtemin=100;
      for(int k=0; k<30; k++)
      {
```

```
            if ((i+k)<0 || (i+k)>xlength || (j)<0 || (j)>ylength){}// Doe niets
            else{
            if ((ari[i+k][j].CompDepth < hoogtemin) && (ari[i+k][j].CompDepth > (-100))){
               hoogtemin=ari[i+k][j].CompDepth;}
            if ((ari[i+k][j].CompDepth > hoogtemax) && (ari[i+k][j].CompDepth < (100))){
               hoogtemax=ari[i+k][j].CompDepth;}}
            //System.out.println(hoogtemax-hoogtemin);
          }
          if(hoogtemax-hoogtemin!=-200){
            amp=amp+hoogtemax-hoogtemin;
            teller++;}

          i=i+10;
        }
      }
    }

    else{
      amp=0;teller=1;}
    if(teller==0){teller=1;}

    return amp/teller;
}

 public static Color KleurAmp(double amp)
{
    int kleur1;
    int kleur2;
    int kleur3;

    //DrawingWindow d = new DrawingWindow(600,600);


      if (amp<0)
      {
        kleur1=100;
        kleur2=100;
        kleur3=100;
      }
      else if (amp<2.5)
      {
        kleur1=250;
        kleur2=250-(int)(100*amp);
        kleur3=0;
      }
      else if (amp<5)
      {
        kleur1=250;
        kleur2=0;
        kleur3=(int)(100*(amp-2.5));
      }
      else if (amp<25)
      {
        kleur1=250;
        kleur2=0;
        kleur3=0;
      }
      else
      {
        kleur1=100;
        kleur2=100;
        kleur3=100;
      }

      return new Color(kleur1,kleur2,kleur3);

      //d.setForeground(new Color(kleur1,kleur2,kleur3));
      //Circle rond1 = new Circle(50+i,150,5);
            //d.fill(rond1);
}
```

```
}
```