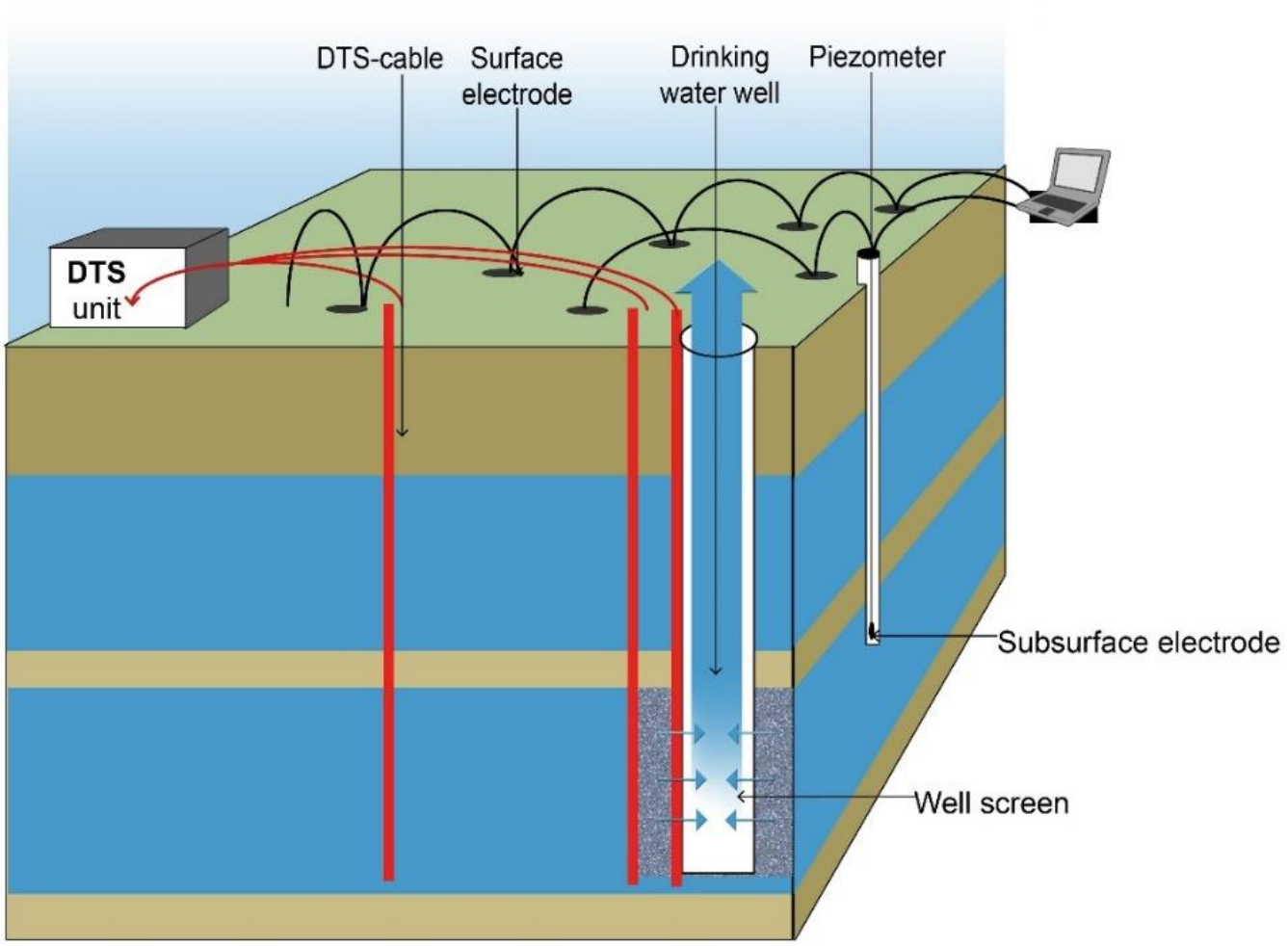
Ontwikkeling van een geautomatiseerd glasvezelmeetsysteem voor het monitoren van grondwaterstromingen rond grondwaterputten

Rapportage van TKI project 2019DEL002



Ontwikkeling van een geautomatiseerd glasvezelmeetsysteem voor het monitoren van grondwaterstromingen rond grondwaterputten
Rapportage van TKI project 2019DEL002

Dit rapport is gezamenlijk opgesteld door de consortiumpartners

Partners
Stchting Deltares, Delft
Aveco de Bondt BV, AMERSFOORT
Vitens Locatie Zwolle, ZWOLLE
N.V. Waterbedrijf Groningen, GRONINGEN

Auteurs
Ane Wiersma (Deltares)
Pieter Doornenbal (Deltares)
Manos Pefkos (Deltares)
Wiecher Bakx (Aveco de Bondt)
Anouk Sprong (Vitens)
Johannes Dunnewolt (Vitens)
Sjoerd Rijpkema (Waterbedrijf Groningen)

Kwaliteitsborging
Victor Hopman (Deltares)

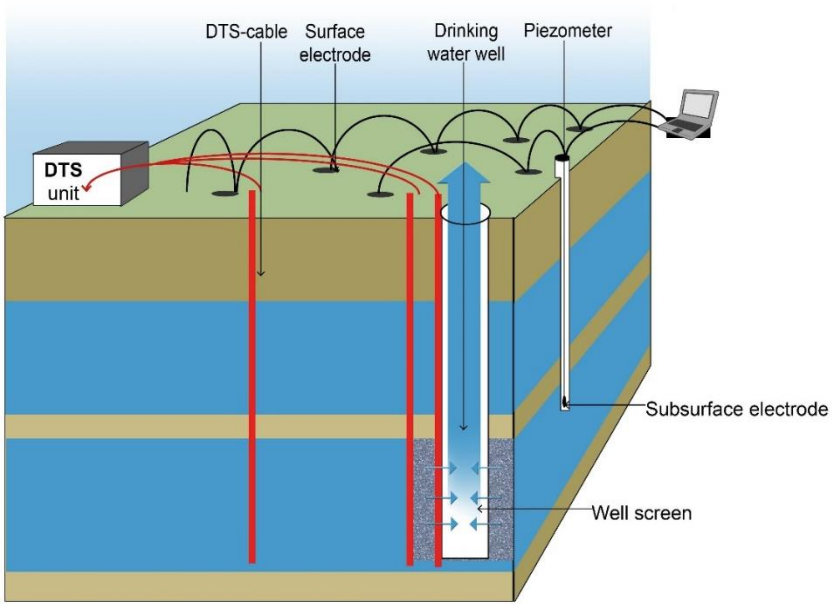
Deze activiteit is mede gefinancierd door TKI Watertechnologie uit de PPS-innovatie programmasubsidie van het Ministerie van Economische Zaken

Voorpagina: Schematische weergave van een geautomatiseerd grondwatermonitoringssysteem dat gebruik maakt van AH-DTS. De rode lijnen geven de glasvezelkabels aan (W. Bakx).

Samenvatting

Dit rapport bespreekt de resultaten van TKI watertechnologieproject 2019DEL002 *"Ontwikkeling van een geautomatiseerd glasvezelmeetsysteem voor het monitoren van grondwaterstromingen rond grondwaterputten"*. Het project richt zich op het meten van grondwaterstroming in en rondom drinkwaterputten door gebruik te maken van glasvezeltechnologie. Het project is een samenwerking tussen Deltares, Vitens NV, Waterbedrijf Groningen en Aveco de Bondt.
Glasvezelkabels kunnen als sensor worden gebruikt, onder andere voor het meten van temperatuur. Dit resulteert in een lange nauwkeurige sensor met temperatuurwaarden langs de hele kabel (Distributed Temperature Sensing: DTS). Door langs deze kabel een opwarmdraad als warmtebron te gebruiken, kan met behulp van elektriciteit deze kabel opgewarmd worden (Active Heating DTS: AH-DTS). Als zo'n glasvezelkabel met opwarmdraad verticaal in de ondergrond wordt geïnstalleerd en de kabel wordt vervolgens opgewarmd, is de snelheid van opwarming en vervolgens van afkoeling te relateren aan de stroomsnelheid van grondwater langs de kabel. Bij een hoge stroomsnelheid warmt de kabel langzamer op en koelt deze sneller af.

Dit project heeft als doel om de AH-DTS techniek toepasbaar te maken in de praktijk voor het meten van grondwaterstroming rondom en in drinkwaterputten. Enerzijds wordt de techniek doorontwikkeld voor deze specifieke toepassing en anderzijds wordt de techniek toegepast in pilotstudies in en rondom drinkwaterputten als proof of concept (Figuur 1).



Figuur 1 Schematische weergave van een geautomatiseerd grondwatermonitoringssysteem dat gebruik maakt van AH-DTS. De rode lijnen geven de glasvezelkabels aan.

Het project is opgedeeld in 7 werkpakketten. Deze bestaan uit drie technische werkpakketten waarin benodigde meet- en verwerkingstechnieken worden ontwikkeld voor deze specifieke toepassing, en vier werkpakketten met pilotstudies waarin de potentie van de techniek wordt onderzocht en waarin onderzoeksvragen worden beantwoordt. Deze onderzoeksvragen zijn ingebracht door Vitens en Waterbedrijf Groningen. Hieronder worden de werkpakketten beschreven als ook de voornaamste resultaten.

Technische werkpakketten:

1 ***Vertaling temperatuurmeetdata naar grondwaterstroming in scripts (computercode)***
In dit werkpakket zijn vijf python scripts ontwikkeld om de ruwe temperatuurmetingen te vertalen naar stroomsnelheid. Als input voor de scripts moeten de locatie specifieke parameters van de meetopstelling worden vastgelegd in een excel bestand.

2 ***Ontwikkelen van een optimale opwarm glasvezelkabel (AH-DTS kabel)***
In FlexPDE zijn modelleringen uitgevoerd die gebruikt zijn om een optimale glasvezelkabel te ontwikkelen die nauwkeurig grondwaterstroming kan meten rondom drinkwaterputten. Met de modellering is inzicht verkregen in de optimale configuratie van opwarmdraden, glasvezelkabels en materiaaleigenschappen te bepalen om nauwkeurige grondwaterstroming te kunnen meten ongeacht de draaiing van de kabel.
De ontwikkelde opwarmkabel moet zowel praktisch toepasbaar zijn in het veld als nauwkeurig de temperatuur meten. Bij de modellering is gefocusseerd op de draaiing van de kabels. Er is een kilometer van deze kabel geproduceerd die toegepast kan worden in het veld en voor tests in het laboratorium.
In het laboratorium zijn proeven uitgevoerd om voor de nieuwe kabel een formule af te leiden voor de stroming langs de kabel.

3 ***Ontwikkelen mobiele meet- en aanstuur opstelling voor gebruik op de pilot locaties***
Er is een aansturingskast ontwikkeld die meerdere glasvezelkabels kan aansturen en kan opwarmen en uitlezen. Van deze aansturingskast is een blauwdruk opgeleverd. De aansturingskast is mobiel en kan gemakkelijk meegenomen worden in een aanhanger.

Pilotstudie werkpakketten:

4. ***Meten stroming in de drinkwaterput – 't Klooster***
Een AH-DTS kabel is in een drinkwaterput aangebracht waarin ook een reguliere flowmeterproef is uitgevoerd. Een AH-DTS kabel heeft grote voordelen omdat de kabel op ieder moment kan worden uitgelezen zonder dat de pomp eerst moet worden uitgebouwd. Een vergelijking tussen de metingen gedaan met een AH-DTS kabel en een reguliere flowmeting laat zien dat de metingen overeenkomsten vertonen. Echter, in tegenstelling tot de reguliere metingen worden de AH-DTS resultaten ook beïnvloed wordt door versnellingen in grondwaterstroming. Voordat de AH-DTS metingen een volwaardig alternatief zijn voor de regulieren flowmetermetingen moeten deze verschillen en effecten beter worden begrepen en gekwantificeerd in een laboratoriumopstelling.

5. ***Meten stroming rondom drinkwaterput – 't Klooster***
In winveld 't Klooster is gebruik gemaakt van het glasvezel meetnet van de voorloper van dit project (Deltares referentie 11201171). Getracht is om variaties in grondwaterstroming te meten bij aanpassingen in het pompregime van de verschillende grondwaterputten. Er is helaas gebleken dat in de loop der tijd een aantal kabels onbruikbaar zijn geworden. Dit is een tegenvaller, maar wel een belangrijk resultaat in het licht van robuustheid van het meetnet.

6. ***Meten van infiltratie uit een infiltratiesloot – 't Klooster***
Twee verticale meetpalen omwikkeld met glasvezelkabels zijn geplaatst in een sloot waar proceswater in wordt afgevoerd. Met de temperatuurmetingen is onderzocht of en hoe snel er water infiltreert van de sloot in de ondergrond. Ook is er gekeken naar de relatie tussen mogelijke infiltratie en de activiteit van de dichtstbijzijnde pompput. In de gemeten temperatuurprofielen zijn de overgangen tussen lucht, water en sediment duidelijk zichtbaar. Ook is er een duidelijk dag-nachtritme waarneembaar. Infiltratie van oppervlaktewater is echter niet waargenomen, ook niet bij het aanzetten van de pompput.

7. ***Meten stroming naar de drinkwaterput - Groningen***
   Op de winlocatie de Groeve van Waterbedrijf Groningen vindt putverstopping plaats doordat kleine deeltjes de toestroom rondom de filters blokkeren. Om inzicht te krijgen in dit proces zijn AH-DTS glasvezelkabels aan beide zijden van de grondwaterput geïnstalleerd om de toestroming te monitoren. Ook is er op een afstand van 5 m een derde glasvezelkabel geïnstalleerd om meer inzicht te krijgen in de toestroom van het grondwater op afstand van de put en op welke locatie de verstopping precies plaatsvindt.

   Uit de uitgevoerde metingen tijdens het pompen komen duidelijk de stukken in de put naar voren waar geen toestroming is (de blindstukken tussen de filters). De uitgevoerde flowmetingen zijn goed te relateren aan de AH-DTS resultaten. Ook kan er onderscheid gemaakt worden tussen de verschillende omstortingen die zijn toegepast. Dit geeft vertrouwen dat eventuele putverstopping die in de loop der tijd ontstaat ook waarneembaar zullen zijn. Echter, in de meetperiode zijn deze verstoppingen niet opgetreden.

Resultaten en aanbevelingen

De resultaten laten de potentie van de techniek duidelijk zien. De robuuste toepassing in het meten van putstroming vereist nader onderzoek naar de invloed en kwantificatie van de zijdelingse toestroom.

Inhoud

1 Inleiding

Dit rapport bespreekt de resultaten van TKI watertechnologieproject 2019DEL002 *"Ontwikkeling van een geautomatiseerd glasvezelmeetsysteem voor het monitoren van grondwaterstromingen rond grondwaterputten"*. Het project richtte zich op het ontwikkelen van kennis van het meten van grondwaterstroming in en rond drinkwaterputten door gebruik te maken van glasvezeltechnologie.
Over deze specifieke stroming rond drinkwaterputten en de verdeling van de stroming in de verticaal is nog veel onbekend. Het meten en begrijpen van deze stroming brengt waardevolle inzichten met betrekking tot het optimaliseren van putontwerp, putbeheer en het voorkomen van putverstoppingen. Zo kan bijvoorbeeld door regelmatig geautomatiseerd de stroming te meten in een drinkwaterput onderhoud precies op tijd uitgevoerd worden, wat kosten bespaard. Ook kan het ondergronds ontijzeringsproces beter worden begrepen en geoptimaliseerd.
Dit project richt zich op het ontwikkelen van nieuwe kennis voor de toepassing van de glasvezeltechniek, het ontwikkelen van de automatisering van metingen en omzetten van temperatuurmetingen naar grondwaterstroming, en op de toepassing van de techniek in pilotsstudies waarin specifieke onderzoeksvragen worden beantwoord.

1.1 Achtergrond

Glasvezelkabels kunnen worden gebruikt als een sensor om nauwkeurig temperatuur te meten. Hiertoe wordt een sensor aan de kabel gekoppeld die de terugkerende ruis van een lichtpuls door de kabel kan opvangen en interpreteren. Dit ruissignaal bevat een kenmerk die te relateren is aan de temperatuur van de kabel op de locatie waar de ruis door reflectie is ontstaan. Deze techniek heet 'Distributed Temperature Sensing' (DTS). De techniek kan worden gebruikt voor het in situ meten van grondwaterstroming door de DTS glasvezelkabel te combineren met een warmtebron. Deze techniek, Active Heating – DTS (AH-DTS), is in een eerder onderzoek vormgegeven en getest (Bakx et al. 2019).

Het real-time meten van grondwaterstroming met AH-DTS (zowel onder natuurlijk condities als ook condities waar actief water uit een put wordt onttrokken) kan waardevolle informatie geven bij het monitoren van verontreinigingen, de instroom van oppervlaktewater naar de ondergrond, het meten van kwel of wegzijging en stromingsbeelden bij grondwaterwinningen. Het eerdergenoemde onderzoek is gestart door in een gecontroleerde setting (laboratoriumexperimenten) de techniek toe te passen en te optimaliseren voor het meten van grondwaterstroming (Bakx et al. 2019). Op basis van de laboratoriumexperimenten is een volwaardige pilot opgezet rondom drinkwaterput 13-18 gelegen op het winveld van Hengelo 't Klooster, een productielocatie van Vitens. Voor deze pilot zijn een 16-tal glasvezelkabels rondom de drinkwaterput tot een diepte van 38 m-mv geïnstalleerd (zie Figuur 1 voor een schematische weergave van de meetopstelling). Ook is in de drinkwaterput zelf een glasvezelkabel geplaatst. Over een periode van enkele weken zijn meerdere metingen uitgevoerd. De metingen laten waardevolle inzichten zien met betrekking tot de grondwaterstroming naar de put en de verdeling van de stroomsnelheden over de diepte.

De resultaten uit dit onderzoek zijn veelbelovend, maar zijn gericht op de ontwikkeling en het testen van deze meetmethode. Hierdoor zijn de praktische onderzoeksvragen voor drinkwaterbedrijven blijven liggen. In dit project wordt naast het beantwoorden van de onderzoeksvragen toe gewerkt naar een grondwatermonitoringssysteem dat breed inzetbaar is. Daarnaast is ook de wens om de verzamelde data (temperatuur) geautomatiseerd te vertalen naar bruikbare informatie(grondwaterstroming).

1.2 Opzet van het project

Het project is opgedeeld in 7 werkpakketten. Deze bestaan uit drie technische werkpakketten waarin benodigde meettechnieken worden ontwikkeld voor deze specifieke toepassing, en drie werkpakketten met pilotstudies waarin de potentie van de techniek wordt onderzocht en waarin onderzoeksvragen worden beantwoord. Deze onderzoeksvragen zijn ingebracht door en uitgewerkt samen met Vitens en Waterbedrijf Groningen. Hieronder worden de werkpakketten beschreven en de voornaamste resultaten.

Technische werkpakketten:

1. Vastleggen en ontsluiten vertaling meetdata naar grondwaterstroming (Scripts)
2. Ontwikkelen van een optimale opwarm glasvezelkabel (AH-DTS kabel);
3. Ontwikkelen mobiele meet- en aanstuur opstelling voor gebruik op de pilot locaties.

Pilotstudie werkpakketten:

4. Meten stroming in de drinkwaterput – ’t Klooster (Vitens)
5. Meten stroming rondom drinkwaterput – ’t Klooster (Vitens)
6. Meten van infiltratie uit een infiltratiesloot – ’t Klooster (Vitens)
7. Meten stroming naar de drinkwaterput – De Groeve (Waterbedrijf Groningen)

1.3 Leeswijzer

Dit rapport bundelt de resultaten van de bovenstaande werkpakketten. Hoofdstuk 3, 4 en 5 beschrijven de resultaten van de Technische werkpakketten. Hoofdstuk 6 beschrijft de resultaten van de pilotstudies bij ’t Klooster (werkpakketten 4, 5 en 6) en Hoofdstuk 7 beschrijft de resultaten van de pilotstudie bij winveld De Groeve in Groningen (werkpakket 7). Een discussie over de resultaten en aanbevelingen worden gegeven in Hoofdstuk 8.

2 Methode

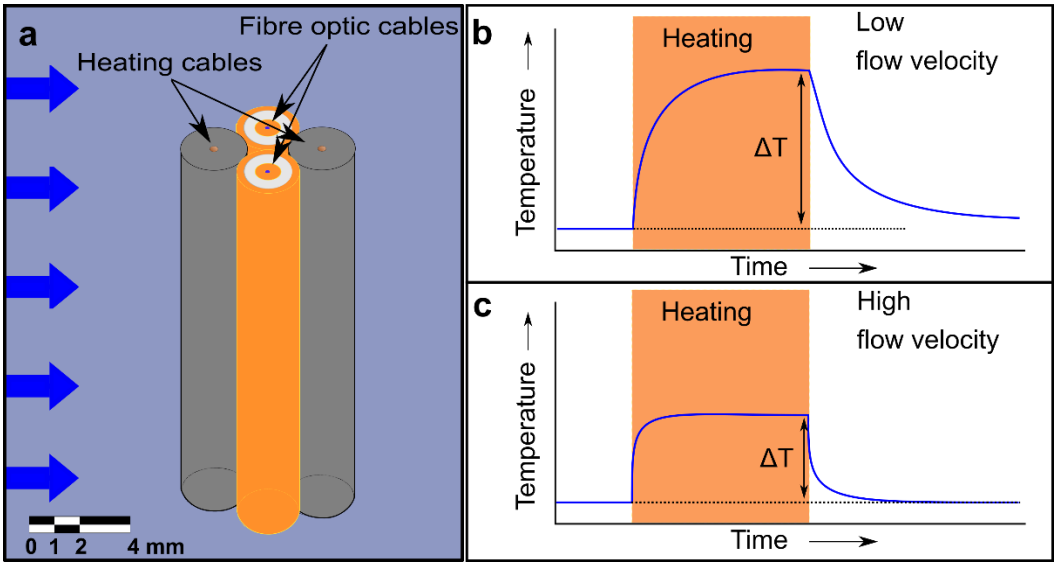
Het voorliggende onderzoek, de doorontwikkeling van de glasvezelkabels en de beschreven pilots richten zich op het meten van grondwaterstroming rond drinkwaterwinputten. Hierbij wordt gebruik gemaakt van glasvezelkabels die continue de temperatuur meten. Door deze kabels te combineren met een opwarmelement kan hieruit de stroomsnelheid worden afgeleid. Het gebruik van glasvezelkabels met een opwarmelement wordt Active Heating Distributed Temperature Sensing (AH-DTS) genoemd. In paragraaf 2.1 wordt het principe van AH-DTS en hoe hier een stroomsnelheid uit kan worden afgeleid beschreven. Paragraaf 2.2 beschrijft hoe de stroomsnelheid wordt berekend vanuit de AH-DTS meetdata.

2.1 Hoe werkt Active Heating Distributed Temperature Sensing (AH-DTS)

DTS is de techniek om met een glasvezelkabel temperatuur te meten over de hele lengte van de glasvezelkabel. Wanneer we een glasvezelkabel combineren met een opwarmelement spreken we over AH-DTS. Wanneer het opwarmelement op afstand is geplaatst van de glasvezelkabel in de ondergrond is er geen sprake van een AH-DTS setup maar is de setup te beschrijven als een tracer experiment met warmte als tracer. In AH-DTS wordt de glasvezel en het opwarmelement dus gecombineerd ingezet. Figuur 2 uit Bakx et al., 2023[1] geeft de setup van een AH-DTS met twee opwarmdraden en 2 glasvezelkabels weer die samen één kabel vormen. De opwarmdraden betreffen hoge elektrische weerstand draden. Wanneer deze onder elektrische stroom worden gebracht warmen deze gelijkmatig over de gehele lengte op. De kabel wordt voor een vaste tijdsperiode opgewarmd waarbij de glasvezelkabel de temperatuur continue monitort. In Figuur 2 b,c is weergegeven hoe het meetresultaat er hierbij komt uit te zien bij verschillende grondwaterstroomsnelheden. Het verschil tussen de achtergrond temperatuur zonder opwarming en de maximale temperatuur gedurende de opwarming (ΔT) is te relateren aan de grondwaterstroomsnelheid. Wanneer water met een hogere stroomsnelheid langs de kabel stroomt, wordt de kabel meer gekoeld en daarmee is het gemeten temperatuurverschil kleiner dan wanneer er water met een kleinere stroomsnelheid langs de kabel stroomt. Zeker bij lage stroomsnelheden zal de periode van opwarming in de praktijk te kort zijn om een evenwichtstemperatuur te bereiken. Dit kan worden opgelost door de evenwichtstemperatuur te bepalen door gebruik te maken van de gevonden relatie tussen stroomsnelheid en gemeten temperatuurverschil, die is bepaald in een lab[2].

[1] Bakx, W.; Bense, V.F.; Karaoulis, M.; Oude Essink, G.H.P.; Bierkens, M.F.P. Measuring Groundwater Flow Velocities near Drinking Water Extraction Wells in Unconsolidated Sediments. Water 2023, 15, 2167. https://doi.org/10.3390/w15122167

[2] Bakx, W.; Doornenbal, P.J.; van Weesep, R.J.; Bense, V.F.; Oude Essink, G.H.P.; Bierkens, M.F.P. Determining the Relation between Groundwater Flow Velocities and Measured Temperature Differences Using Active Heating-Distributed Temperature Sensing. Water 2019, 11, 1619. https://doi.org/10.3390/w11081619



Figuur 2 - Uit Bakx et al., 2023. Voorbeeld van een AH-DTS setup: a) setup van de kabel met 2 opwarmkabels en 2 glasvezelkabels. Die geplaatst zijn loodrecht op de stroming. b) het temperatuurverschil die ontstaat bij het tijdelijk opwarmen van de kabel bij een lage grondwaterstroming langs de kabel. C) het temperatuurverschil dat ontstaat bij een hoge stroomsnelheid langs de kabel.

2.2 Stroomsnelheid berekenen vanuit AH-DTS metingen

De berekening van de stroomsnelheid is uitgevoerd volgens de methode zoals deze is beschreven in Bakx et al., 2019. In deze paragraaf is deze werkwijze samengevat beschreven op basis van dit artikel.

Voor de basisformule is uitgegaan van een analytische oplossing. De basisformule voor de berekening van de stroomsnelheid zoals deze door Bakx et al., 2019 is voorgesteld is:

$$u = \left(\frac{p}{\frac{\Delta T \, 2\pi}{Q} + \left(\frac{1}{k_{tot}} \ln \left(\frac{r_{tot}}{r_{h1}} \right) \right)} \right)^{1/m}$$

Waarbij p een constante is die als volgt is gedefinieerd:

$$p = \frac{L \, \nu^m}{r_{h2} \, k_{eff} \, C \;\; Pr^{1/3} \;\; L^m}$$

De volgende parameters komen voor in de bovenstaande formules:

- u = stroomsnelheid in m/s;
- ΔT = gemeten temperatuurverschil tussen opwarming en achtergrondtemperatuur in °C;
- Q = warmte input die aan de kabel wordt gegeven in W/m;
- k_{tot} = gecombineerde thermische conductiviteit van de materialen van de kabel in W/m/K;
- r_{tot} = som van de radius van de opwarmkabel, de glasvezelkabel (en de afstand door het sediment als opwarmkabel en glasvezelkabel niet aangrenzend zijn) in m;
- r_{h1} = radius van het opwarmgedeelte van de opwarmkabel in m;
- L = diameter van de opwarmkabel in m;
- ν = kinematische viscositeit van water m²/s;
- r_{h2} = totale radius van de opwarmkabel in m;
- k_{eff} = gecombineerde thermische conductiviteit van water en sediment (W/m·K)
- Pr = Prandtl nummer.

C en m zijn de constanten die onder gecontroleerde omstandigheden zijn bepaald. De waarde voor C bedraagt 0.417 en voor m 0.217. Deze waardes zijn kabel specifiek. Bij een andere configuratie van de kabel moeten deze constantes opnieuw worden bepaald in een laboratoriumopstelling. De gebruikte formule is een aangepaste versie van de formule zoals deze is voorgesteld door Read et al.[3].

[3] Read, T.; Bour, O.; Selker, J.S.; Bense, V.; Le Borgne, T.; Hochreutener, R.; Lavenant, N. Active-distributed temperature sensing to continuously quantify vertical flow in boreholes. Water Resour. Res. 2014, 50, 3706–3713.

3 Scripts

Om de data uit de AH-DTS metingen, gemeten met een Silixa, om te zetten naar informatie over de grondwaterstroming moeten een aantal stappen worden uitgevoerd. Hiervoor zijn in dit project vijf scripts ontwikkeld die stap voor stap de data omzetten. Er wordt gewerkt met een metafile (Excel bestand) waarin de omstandigheden van de verschillende meetlocaties waarin de AH-DTS kabels worden ingezet kunnen worden weergegeven, zoals opwarmtrajecten en -duur. De scripts zijn daarna voor verschillende meetlocaties met beperkte aanpassingen in te zetten. In totaal zijn er een vijftal scripts doorontwikkeld in het project:

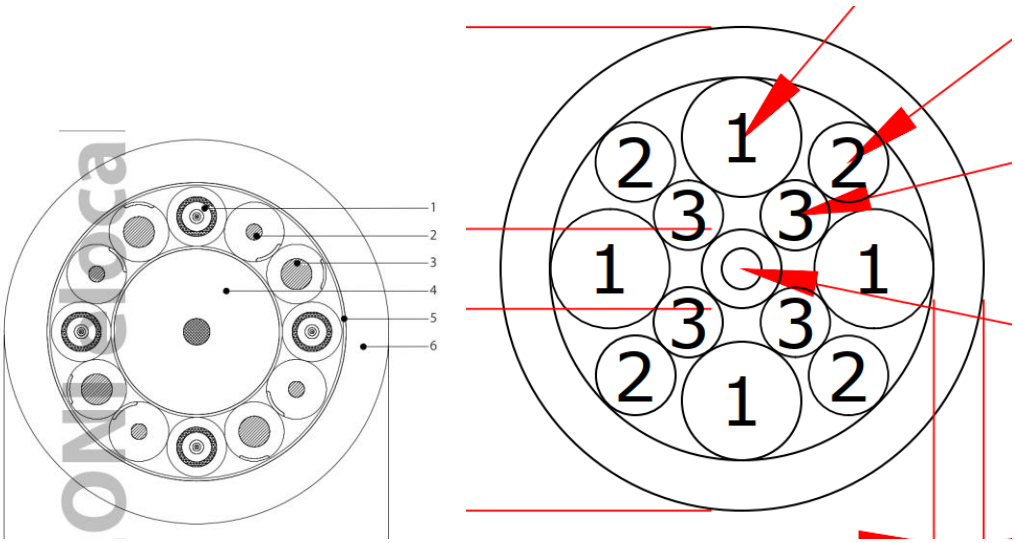
1. Script 1 – XML files van de Silixa worden omgezet naar een bruikbaar Python format
2. Script 2 – De DTS meetdata wordt gekalibreerd op basis van de kalibratiebaden
3. Script 3 – De data wordt teruggebracht tot het deel waar we werkelijk in geïnteresseerd zijn
4. Script 4 – Opdelen van de meetgegevens in data met en zonder opwarming.
5. Script 5 – Bereken ΔT en stroomsnelheid

Bijlage A geeft een toelichting op de scripts. Bijlage B1 t/m B5 geven de inhoud van de vijf ontwikkelde scripts, en Bijlage C geeft een beschrijving van de parameters in de metadata-file.

4 Kabelontwikkeling

4.1 Inleiding

Als onderdeel van het project GroFloMo is een nieuwe AH-DTS meetkabel ontwikkeld. Het doel van deze kabel is om zo nauwkeurig mogelijk grondwaterstroming te kunnen meten. Hiertoe moet de kabel beschikken over glasvezelcomponenten om de temperatuur met DTS te kunnen meten en een opwarmelement. Ook wordt gestreefd naar een zo klein mogelijke invloed van rotatie van de kabel op de meetresultaten, en voldoende sterkte voor de installatie in het veld.
In samenspraak met Felthen Wire & Cable solutions B.V. zijn 2 concepten uitgedacht van een geschikte kabel. Deze twee varianten zijn in Figuur 3 weergegeven.



Figuur 3 Kabelontwerpen optie 1 (links) en optie 2 (rechts). Ontwerpschetsen zoals opgesteld door Felthen

Om te bepalen welke variant het beste geschikt is voor de toepassing voor ogen, is een modellering uitgevoerd waarbij de kabel is gesimuleerd in een grondwaterstromingsomgeving. Hiervoor is gebruik gemaakt van de software FlexPDE.
In de volgende paragrafen wordt de werkwijze en de resultaten van deze modellering bondig beschreven.

4.2 Werkwijze

De twee kabelontwerpen (optie 1 en optie 2) zijn in de software FlexPDE geschematiseerd. Naast de twee kabelontwerpen is ook de reeds toegepaste kabel (originele kabel) en een variant van optie 2 (optie 3) doorgerekend. In optie 3 is de dikte van de buitenste mantel van kabelontwerp optie 2 met 40% verkleind.
Het FlexPDE model betreft een stroming- en warmtetransport model. De geschematiseerde kabel is gesitueerd in direct contact met een zandige ondergrond waarin stroming van grondwater plaatsvindt. Elk los component van de kabel is in het model apart gedefinieerd. De benodigde parameters (thermische conductiviteit, soortelijke warmte heat, dichtheid) zijn door Felthen aangeleverd en waar onbekend ingeschat op basis van literatuur.
Van elke kabelvariant (origineel, optie 1-3) zijn drie rotaties van de kabel in het model ingebouwd. Daarmee kan worden onderzocht in welke mate de rotatie van de kabel bij plaatsing in de ondergrond de nauwkeurigheid van het meetresultaat beïnvloed.

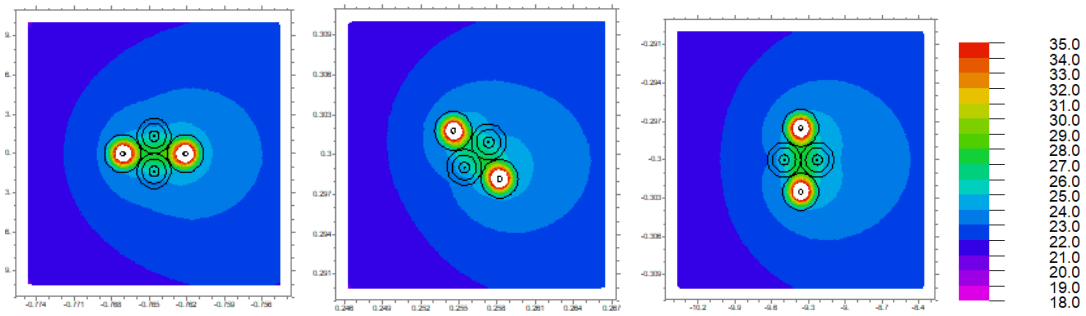
De opgestelde modellen zijn voor 6 verschillende grondwaterstroomsnelheden (0.001-2-5-7-9-40 m/d) doorgerekend. Met het doorrekenen van verschillende stroomsnelheden is de gevoeligheid van het meetresultaat voor de veranderingen in de grondwaterstroming in beeld gebracht.

4.3 Resultaten

Onderstaand worden de resultaten voor de 4 doorgerekende kabelvarianten weergegeven.

4.3.1 Originele kabel

In onderstaand figuur is de schematisatie van de originele kabel in drie verschillende rotatievarianten in het flexPDE model weergegeven.



Figuur 4 Schematisatie van de originele kabel in het FlexPDE model

In onderstaande tabel is het resultaat van de modellering beschreven aan de hand van de resulterende verschiltemperatuur (DeltaT) tussen stand: ’0 en 90 graden rotatie bij verschillende snelheden.

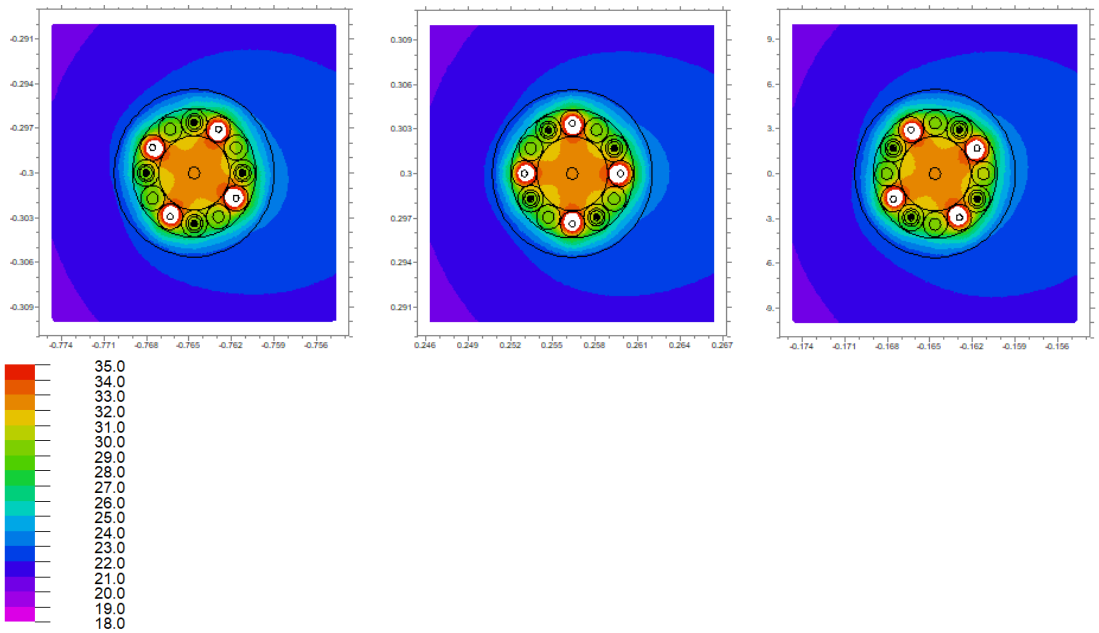
Tabel 1 Originele kabel (proeven 2021)

Snelheid (m/d)	Minimale DeltaT (℃)	Maximale DeltaT (℃)	Verschil DeltaT (℃)	Gemiddelde DeltaT (℃)
0.01	7.81	8.13	0.33	8.02
2	6.93	7.12	0.19	7.05
5	5.56	5.66	0.09	5.62
7	5.07	5.13	0.06	5.11
9	4.72	4.74	0.02	4.74
40	2.70	3.02	0.32	2.88
Maximale variatie DeltaT als gevolg van draaiing kabel			0.33	
Verschil als gevolg van variatie in stroomsnelheid				5.15

Als gevolg van de variatie in stroomsnelheid is er een verschil in de gemiddelde DeltaT van maximaal 5,15 ℃. Dit is een maat voor de gevoeligheid van de kabel voor grondwaterstroming. De maximale variatie in DeltaT als gevolg van de rotatie (draaiing van de kabel) bedraagt 0,33 ℃. Dit is een maat voor de nauwkeurigheid van de kabel aangezien de draaiing van de kabel bij plaatsing onbekend is.

4.3.2 Kabelontwerp optie 1

In onderstaand figuur is de schematisatie van het kabelontwerp optie 1 in drie verschillende rotatievarianten in het flexPDE model weergegeven.



Figuur 5 Schematisatie van kabelontwerp optie 1 in het FlexPDE model

In onderstaande tabel is het resultaat van de modellering beschreven aan de hand van de resulterende verschiltemperatuur (DeltaT) bij verschillende snelheden.

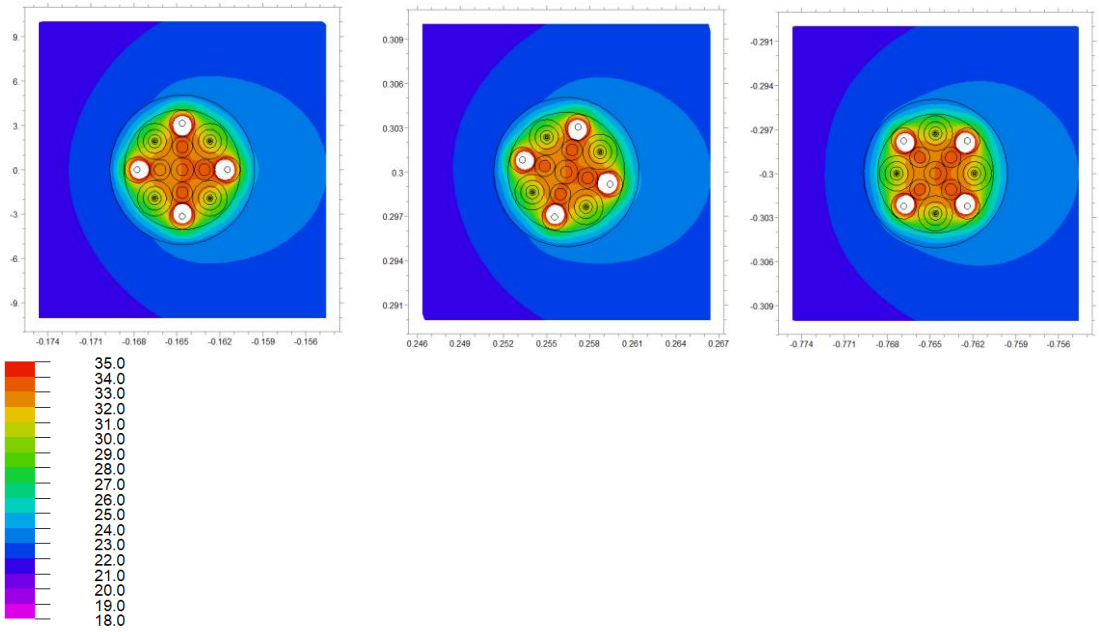
Tabel 2 Kabelontwerp optie 1

Snelheid (m/d)	Minimale DeltaT (℃)	Maximale DeltaT (℃)	Verschil DeltaT (℃)	Gemiddelde DeltaT (℃)
0.01	12.56	12.59	0.03	12.57
2	11.60	11.63	0.03	11.61
5	10.27	10.35	0.08	10.30
7	9.81	9.84	0.03	9.82
9	9.51	9.54	0.03	9.52
40	8.24	8.28	0.04	8.25
Gemiddelde variatie DeltaT als gevolg van draaiing kabel			0.08	
Verschil als gevolg van variatie in snelheid				4.31

Als gevolg van de variatie in snelheid is er een variatie in de gemiddelde DeltaT van 4,31 ℃. De maximale variatie in DeltaT als gevolg van de rotatie (draaiing van de kabel) bedraagt 0,08 ℃.

4.3.3 Kabelontwerp optie 2

In onderstaand figuur is de schematisatie van het kabelontwerp optie 2 in drie verschillende rotatievarianten in het flexPDE model weergegeven.



Figuur 6 Schematisatie van kabelontwerp optie 2 in het FlexPDE model

In onderstaande tabel is het resultaat van de modellering beschreven aan de hand van de resulterende verschiltemperatuur (DeltaT) bij verschillende snelheden.

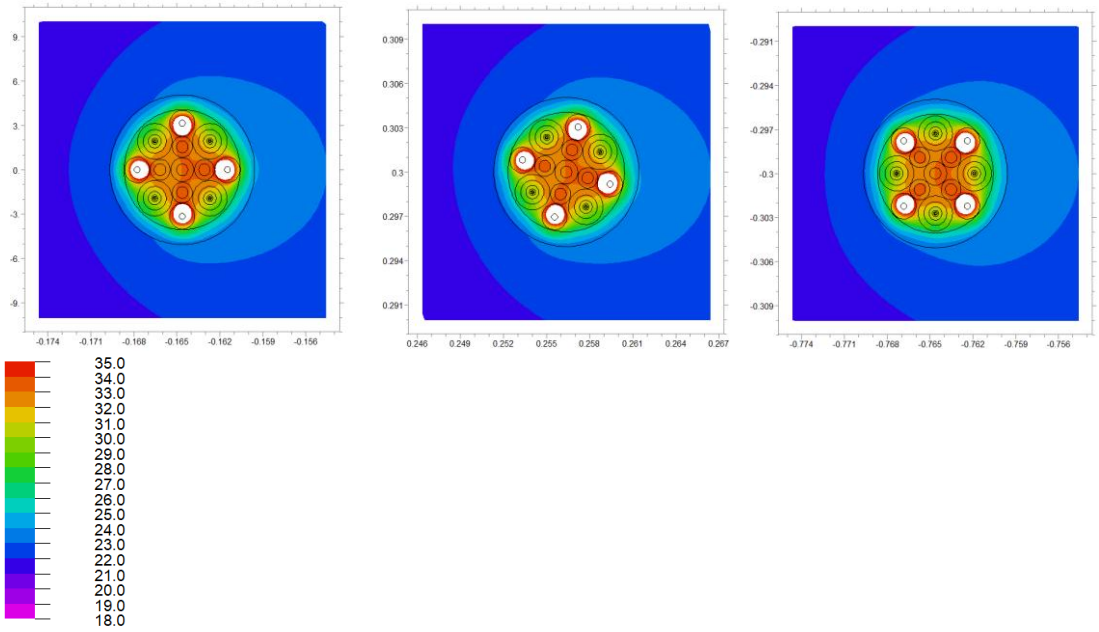
Tabel 3 Kabelontwerp optie 2

Snelheid (m/d)	Minimale DeltaT (℃)	Maximale DeltaT (℃)	Verschil DeltaT (℃)	Gemiddelde DeltaT (℃)
0.01	12.95	12.95	0.00	12.95
2	11.97	11.97	0.00	11.97
5	10.61	10.62	0.01	10.62
7	10.15	10.15	0.01	10.15
9	9.83	9.83	0.00	9.83
40	8.50	8.51	0.00	8.50
Gemiddelde variatie DeltaT als gevolg van draaiing kabel			0.01	
Verschil als gevolg van variatie in snelheid				4.44

Als gevolg van de variatie in snelheid is er een variatie in de gemiddelde DeltaT van 4,44 ℃. De maximale variatie in DeltaT als gevolg van de rotatie (draaiing van de kabel) bedraagt 0,008 ℃.

4.3.4 Kabelontwerp optie 3

In onderstaand figuur is de schematisatie van het kabelontwerp optie 3 in drie verschillende rotatievarianten in het flexPDE model weergegeven.



Figuur 7 Schematisatie van kabelontwerp optie 3 in het FlexPDE model

In onderstaande tabel is het resultaat van de modellering beschreven aan de hand van de resulterende DeltaT bij verschillende snelheden.

Tabel 4 Kabelontwerp optie 3

Snelheid (m/d)	Minimale DeltaT (℃)	Maximale DeltaT (℃)	Verschil DeltaT (℃)	Gemiddelde DeltaT (℃)
0.01	11.83	11.83	0.00	11.83
2	10.84	10.85	0.00	10.84
5	9.46	9.47	0.01	9.47
7	8.99	9.00	0.01	8.99
9	8.67	8.67	0.00	8.67
40	7.29	7.29	0.00	7.29
Maximale variatie DeltaT als gevolg van draaiing kabel			0.01	
Verschil als gevolg van variatie in snelheid				4.53

Als gevolg van de variatie in snelheid is er een variatie in de gemiddelde DeltaT van 4,53 ℃. De maximale variatie in DeltaT als gevolg van de rotatie (draaiing van de kabel) bedraagt 0,008 ℃.

4.4 Conclusie en advies

De resultaten van de doorgerekende varianten zijn samengebracht in onderstaande tabel. In deze tabel is de afhankelijkheid van de DeltaT voor de grondwatersnelheid voor elke variant weergegeven. Ook is in beeld gebracht in welke mate de kabel gevoelig is voor rotatie.
Wanneer we kijken naar de gevoeligheid van de DeltaT voor de grondwatersnelheid dan zien we dat de originele kabel hierop het beste scoort. Dit ligt in lijn met de verwachting aangezien dit een ‘open’ kabel betreft zonder een buitenmantel. Ontwerp optie 1 laat ca 16% minder variatie in DeltaT zien. Bij optie 2 is dit 14% en optie 3 12%. Voor het meten van stroming is de originele kabel op dit punt dus het beste. Van de nieuwe ontwerp opties is kabeloptie 2 het beste. Zeker wanneer de buiten mantel nog verder kan worden verdund (dus optie 3).

Wanneer we kijken naar de gevoeligheid van de kabel voor rotatie dan scoort de originele kabel veruit het slechts. Als gevolg van de rotatie van de kabel ontstaat er een verschil van 0.169℃. De ontwerp optie 1 laat een grotere nauwkeurigheid zien. De variatie als gevolg van de rotatie is een factor 4 kleiner. Kabel ontwerp optie 2 en 3 zijn echter nog aanzienlijk nauwkeuriger. De variatie als gevolg van de rotatie is voor deze opties tot 40 x kleiner dan bij de originele kabel.

Tabel 5 Overzicht van de resultaten van de verschillende varianten

Variant	Variatie DeltaT als gevolg van snelheid	Percentage t.o.v. origineel	Nauwkeurigheid i.r.t rotatie	Nauwkeurigheids factor t.o.v. origineel
Originele kabel	5.15 ℃	100 %	0.33 ℃	1 x
Ontwerp optie 1	4.31 ℃	83.78 %	0.08 ℃	3.9 x
Ontwerp optie 2	4.44 ℃	86.30 %	0.01 ℃	40.8 x
Ontwerp optie 3	4.53 ℃	88.08 %	0.01 ℃	40.1 x

Samenvattend kan het volgende worden geconcludeerd:

- Alle ontwerpvarianten hebben een aanzienlijke toename in nauwkeurigheid ten opzichte van het origineel. Het effect van de rotatie op de deltaT wordt zeker bij ontwerpen 2 en 3 tot een verwaarloosbaar niveau verlaagd.
- Ten opzichte van de originele kabel zijn de ontwerp opties minder gevoelig voor de grondwatersnelheid. Dit is het gevolg van de al omsluitende mantel. De ontwerp opties 2 en 3 laten hierbij het beste resultaat zien. Het beperken van de dikte van de buitenmantel met 40% (optie 3) zorgt voor een 0.1℃ toename in de gevoeligheid voor de snelheid.
- Ontwerp optie 2 (en 3) zijn op beide fronten sterker dan ontwerp optie 1. Daarmee heeft deze ontwerpoptie de voorkeur.
- Ook vanuit een praktische overweging heeft optie 2 (en 3) een voorkeur ten opzichte van ontwerp optie 1. De glasvezelkabels zijn meer beschermd als losse eenheden in de kabel. Bij het openhalen van de kabel om de weerstand en elektra kabels te verbinden is er minder risico dat de glasvezelkabels worden beschadigd.

Het is echter wenselijk de gevoeligheid voor de grondwatersnelheid zoveel mogelijk te verhogen. Het is positief dat de onzekerheid als gevolg van de rotatie wordt gereduceerd, maar dit compenseert nog niet het verlies in de gevoeligheid i.r.t. de grondwatersnelheid. Een verdere verdunning, of beter nog een open structuur, voor de buitenmantel kan op dit punt tot een beter resultaat leiden. Kabel ontwerp optie 2 is hierbij het vertrekpunt.

4.5 Aanvullende varianten optie 1

Na overleg met de fabrikant blijkt kabel optie 2 moeilijk te fabriceren. Het is in dit ontwerp niet te realiseren dat de kabelonderdelen met de gewenste indeling in de kabel komen. Dit kan dan gaan variëren over de lengte van de kabel. Wel blijkt na overleg met de fabrikant dat de buitenmantel van kabeloptie 1 aanzienlijk dunner kan. Deze is nu 1.3 mm dik. Een dikte van 0.5 mm is zeker haalbaar. Mogelijk is een verkleining naar een dikte van 0.2 mm ook nog haalbaar. Daarom zijn er twee nadere varianten van kabeloptie 1 doorgerekend.
Bij de 1e variant (optie 1_05) is de dikte van de buitenmantel teruggebracht naar 0.5 mm en bij de 2e variant (optie 1_02) is de dikte van de buitenmantel teruggebracht naar 0.2 mm. In onderstaande tabellen is het resultaat beschreven.

Tabel 6 Kabelontwerp optie 1_05

Snelheid (m/d)	Minimale DeltaT (℃)	Maximale DeltaT (℃)	Verschil DeltaT (℃)	Gemiddelde DeltaT (℃)
0.01	10.43	10.46	0.03	10.44
2	9.46	9.49	0.03	9.47
5	8.09	8.11	0.02	8.10
7	7.62	7.65	0.03	7.63
9*				
40	5.94	5.98	0.05	5.96
Maximale variatie DeltaT als gevolg van draaiing kabel			0.05	
Verschil als gevolg van variatie in snelheid				4.48

* Berekening mislukt

Tabel 7 Kabelontwerp optie 1_02

Snelheid (m/d)	Minimale DeltaT (℃)	Maximale DeltaT (℃)	Verschil DeltaT (℃)	Gemiddelde DeltaT (℃)
0.01	9.56	9.59	0.03	9.57
2	8.58	8.61	0.03	8.59
5	7.20	7.22	0.03	7.21
7	6.72	6.75	0.03	6.73
9	6.39	6.42	0.03	6.40
40	4.99	5.04	0.05	5.02
Maximale variatie DeltaT als gevolg van draaiing kabel			0.05	
Verschil als gevolg van variatie in snelheid				4.56

Wanneer we deze varianten naast de eerdere opties zetten, komen we op het volgende overzicht:

Tabel 8 Overzicht van de resultaten van de verschillende varianten

Variant	Variatie DeltaT als gevolg van snelheid	Percentage t.o.v. origineel	Nauwkeurigheid i.r.t rotatie	Nauwkeurigheids factor t.o.v. origineel
Originele kabel	5.15 ℃	100 %	0.325 ℃	1 x
Ontwerp optie 1 – mantel 1.3mm	4.31 ℃	83.78 %	0.083 ℃	3.9 x
Ontwerp optie 2	4.44 ℃	86.30 %	0.008 ℃	40.8 x
Ontwerp optie 3	4.53 ℃	88.08 %	0.008 ℃	40.1 x
Ontwerp optie 1 – mantel 0.5mm	4.49 ℃	87.18 %	0.046 ℃*	7 x*
Ontwerp optie 1 – mantel 0.2 mm	4.56 ℃	88.55 %	0.049 ℃	6.6 x

* Door ontbreken van 1 berekening is deze waarde minder betrouwbaar. Deze ligt naar verwachting hoger

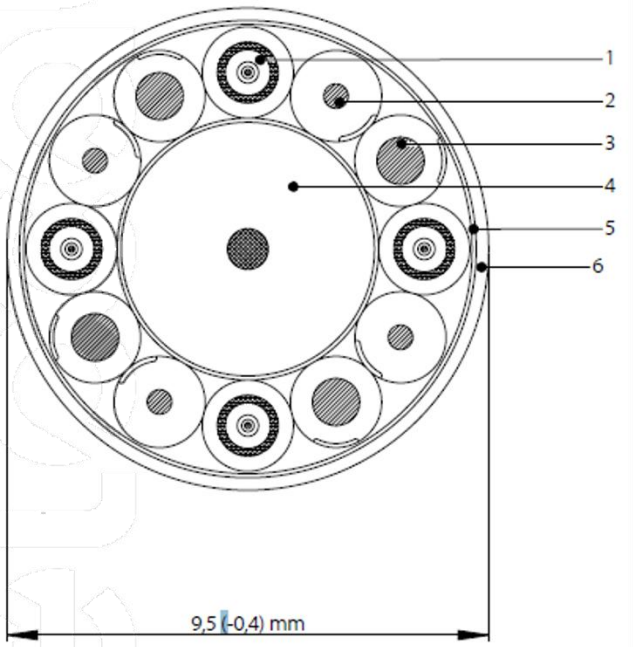
4.5.1 Conclusie ontwerpvarianten

Een dunnere buitenmantel zorgt voor een grotere nauwkeurigheid i.r.t. rotatie dan de originele kabeloptie 1. Kabelontwerp optie 2 blijft hier echter nog een factor 10 beter. Wel kan gesteld worden dat met deze ontwerpen de nauwkeurigheid op dit punt dusdanig verbetert, dat de rotatie van de kabel geen significante rol meer speelt.
Wanneer we kijken naar de gevoeligheid voor de grondwatersnelheid zien we ten opzichte van de basis kabeloptie 1 ook een verbetering. De gevoeligheid neemt toe. Met een mantel van 0.2 mm wordt deze zelfs beter dan het kabelontwerp 3. De gevoeligheid bereikt nog niet de mate van de originele kabel. Om hier dichter bij in de buurt te komen zal naar verwachting een ontwerp zonder een buitenmantel nodig zijn.
Felten heeft nagevraagd of een buitenmantel van gevlochten kevlar een optie kan zijn. Dit is technisch mogelijk, maar brengt de kabel wel in een andere (duurdere) prijsklasse.

4.5.2 Uiteindelijke keuze kabel

Op basis van deze analyse en praktische uitgangspunten is ervoor gekozen om kabelvariant 1 te ontwikkelen, maar dan met een zo dun mogelijke buitenmantel. Deze kabel heeft de beste combinatie van gevoeligheid voor rotatie, gevoeligheid voor het meten van grondwaterstroming, sterkte en handelbaarheid in het veld en gecontroleerde fabricage.

Het ontwerp en de eigenschappen van de uiteindelijk ontwikkelde kabel zijn weergegeven in Figuur 8. De uiteindelijk dikte van de buitenmantel is ca 0.3 mm. De eigenschappen hiervan komen het dichts overeen met doorgerekende ontwerp optie 1 mantel 0.5 mm. Van de kabel is 1000 m besteld en geleverd voor de installatie in drinkwaterputten.

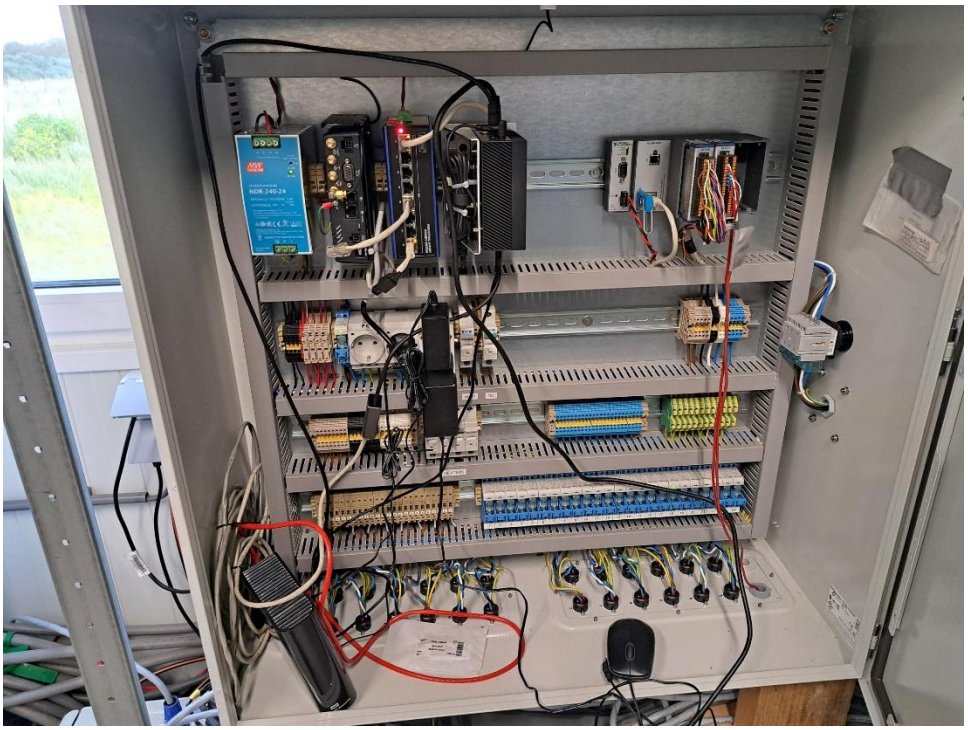


pos. 1 4 optical fibre G50/125 OM3 Bend Intensive I-V(ZN)H1
colour orange
pos. 2 4 insulated wires 0.14 mm²
conductor (7 x 0.16 mm) CuNi15
insulation Specialpolymer
colour black; printed with numbers 1 – 4
pos. 3 4 insulated wires 0.5 mm²
conductor (19 x 0.185 mm) tin plated copper
insulation Specialpolymer
colour black; printed with numbers 5 – 8
pos. 4 strength member Kevlar,
min. 500 N
sheathing TPU
colour black
binder
pos. 5 binder
pos. 6 outer jacket
material LD-PE
colour black
wall thickness 0.3 ± 0.15mm
printing white;
elocab EHRK 200936 REV.00 FA_XXXXXXX

Figuur 8 Ontwerp en eigenschappen van de gefabriceerde kabelvariant voor het meten van grondwaterstroming

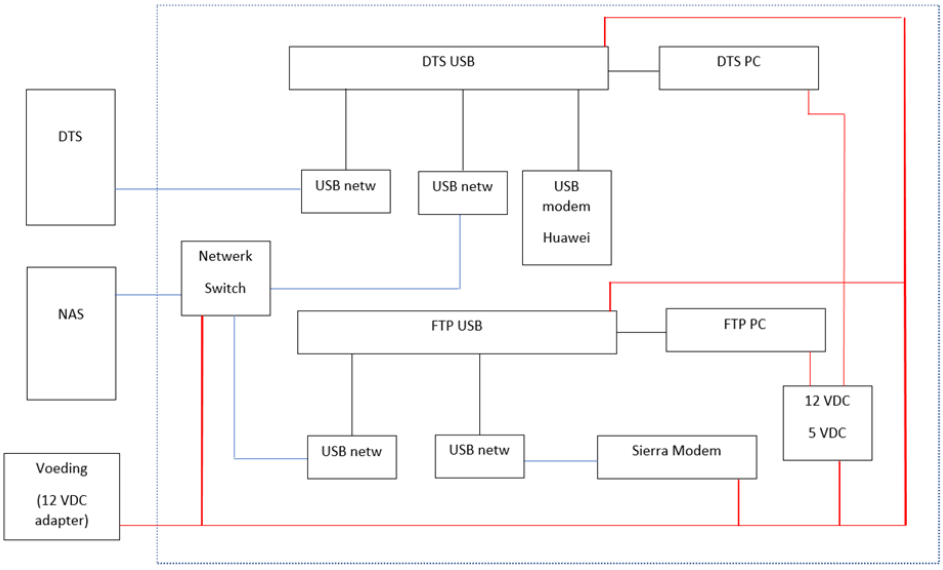
5 Opwarmkast

Voor dit project is een opwarmkast ontworpen en gebouwd om het elektrische deel van de AH-DTS kabel aan te sturen en op te warmen. De kast heeft een onboard computer die op afstand bediend kan worden waarmee de duur, de hoeveelheid stroom en de frequentie digitaal aangestuurd kan worden. In Figuur 9 is de opwarmkast getoond.

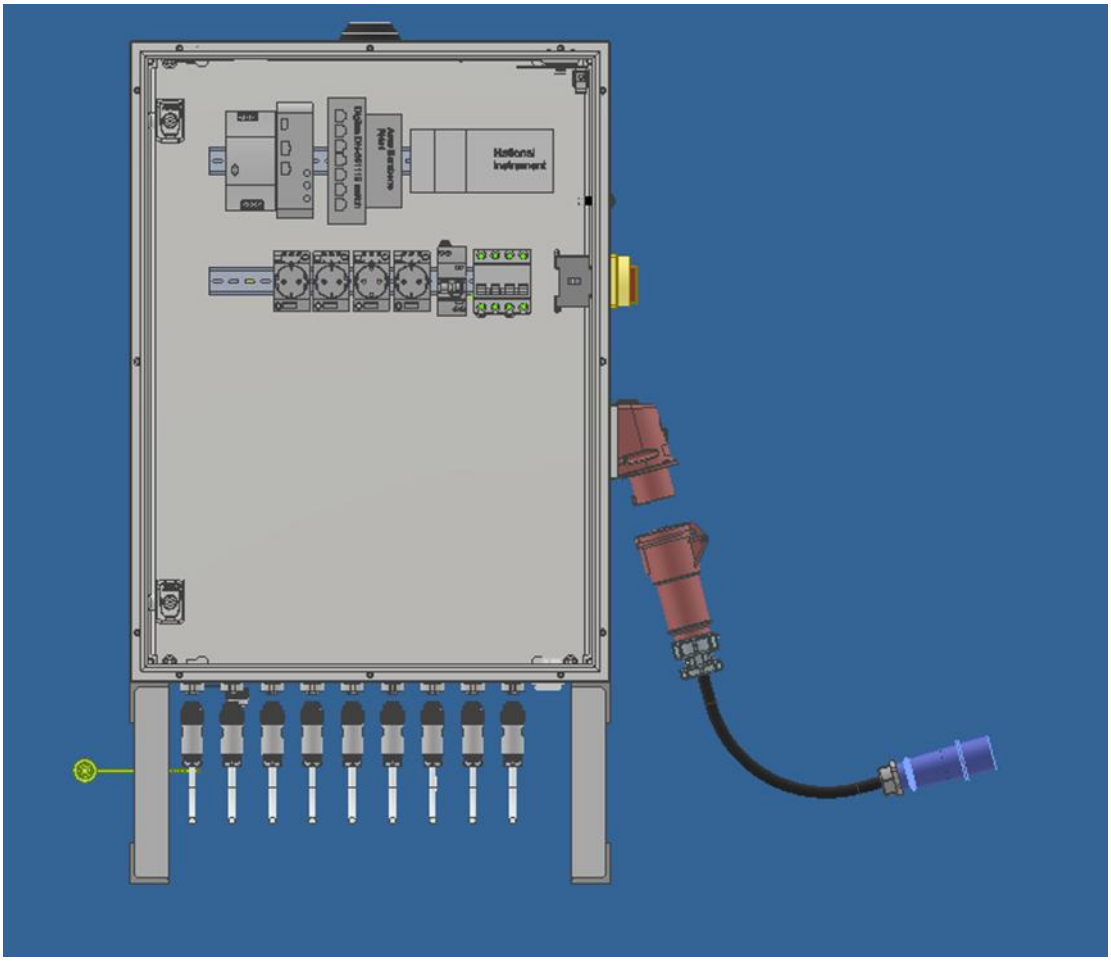


Figuur 9 Binnenkant van de opwarmkast

In onderstaand schema en figuur (Figuur 10; Figuur 11) is de aansturing en opslag van de DTS data schematisch weergegeven.



Figuur 10 Grafische representatie van de aanstuurkast om de warmtepulsen naar de AH-DTS kabels aan te sturen. De elementen binnen de blauwe stippellijn zijn de componenten in de opwarmkast.



Figuur 11 Uiteindelijke opwarmkast, waarbij de voedingskabel aan de rechterbuitenkant binnenkomt en de glasvezelkabels met opwarmelementen (AH-DTS) aan de onderkant.

6 Pilots bij Vitens ’t Klooster

In de gemeente Bronckhorst ten oosten van Hengelo (GLD) en ten noorden van Zelhem heeft Vitens een drinkwaterwinning ’t Klooster. Dit is een freatische drinkwaterwinning met een onttrekkingsvergunning van 5,44 miljoen m^3/jaar. Deze hoeveelheid wordt onttrokken met behulp van twaalf winputten op een diepte variërend tussen de 0 en -20 m NAP. Het maaiveld bevindt zich op een hoogte van zo’n 16 m+NAP. Daarnaast vindt er in de wintermaanden wateraanvoer vanuit de Veengoot plaats. Dit water wordt rondom de winning geïnfiltreerd.
Bij ’t Klooster zijn twee typen metingen uitgevoerd. Eén bestaat uit het meten van grondwaterstroming in en rondom een drinkwaterput (Paragraaf 6.1 en 6.2), en één uit het meten van infiltratie van oppervlaktewater vanuit een sloot naar de ondergrond (Paragraaf 6.2).

6.1 Stroming in de put

In de pilot ‘putstroming’ is onderzocht of en in welke mate AH-DTS kabels kunnen worden gebruikt om de stroomsnelheid van het water in een drinkwaterput te meten. Daarom is er naast de plaatsing van AH-DTS kabels voor het meten van grondwaterstroming rondom een drinkwaterput ook een glasvezelkabel geplaatst in één van de drinkwaterputten op ’t Klooster. Deze installatie heeft reeds plaatsgevonden tijdens een eerder onderzoek in 2017.

Anders dan bij de metingen van het grondwater in de ondergrond gaat het hierbij om een stroming die langs de kabel loopt. Daarmee is de vertaling van het meetsignaal (temperatuur) niet op eenzelfde wijze te vertalen naar stroomsnelheden (zoals in hoofdstuk 2 is beschreven).

Met de pilot is onderzocht of de gemeten temperatuursverandering (ΔT) met een glasvezelkabel gebruikt kunnen worden om de stromingsverdeling over de diepte in een pompput af te leiden. Een belangrijke aanname bij dit principe is dat de variatie van de koeling van de glasvezelkabel alleen wordt beïnvloed door de stroming van water in de put. Om een goede vertaling van de gemeten temperatuursverandering naar stroomsnelheden van water mogelijk te maken is een laboratorium experiment nodig, waarin deze relatie in een gecontroleerde omgeving vast gesteld kan worden. Een dergelijk laboratorium experiment is in het project niet meer uitgevoerd. Wel zijn de ΔT metingen vergeleken met een stroomsnelheidsmeting in de betreffende put die is uitgevoerd in 2017.

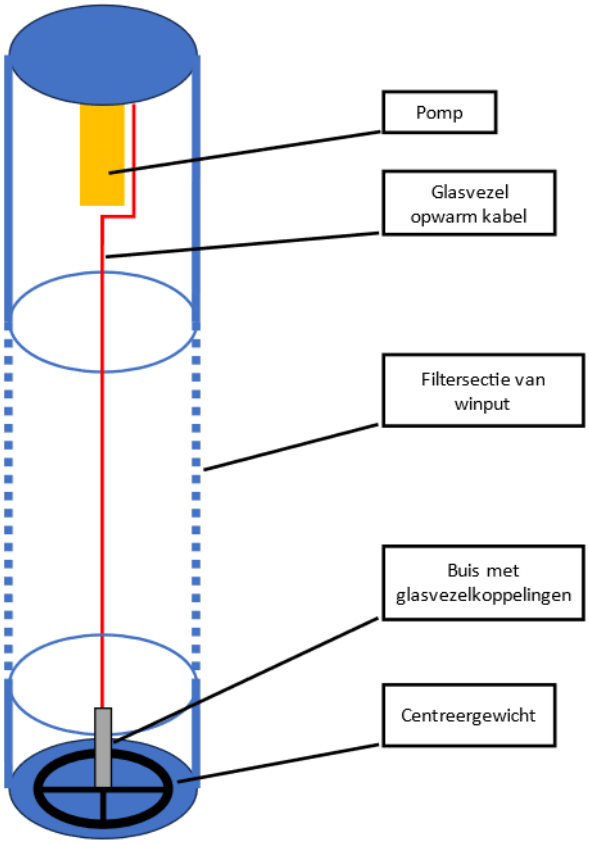
In voorliggend hoofdstuk wordt de opzet en uitkomsten van dit onderzoek beschreven. In paragraaf 6.1.1 wordt de methode en de meetopstelling toegelicht. In paragraaf 6.1.2 beschrijven wij de analyse en de resultaten op basis van de verzamelde gegevens. In paragraaf 6.1.3 staan we in de discussie stil bij wat de resultaten ons vertellen.

6.1.1 Meetopstelling

In Figuur 12 is een schematische weergave gegeven van de meetopstelling in de winput. De AH-DTS kabel is in het midden van de winput gemonteerd , aan de onderkant is een gewicht bevestigd die de kabel centreert in de buis. De kabel tot net onder de pomp. Via een door Vitens ontworpen passeerbuis wordt de AH-DTS kabel voorbij de winpomp geleidt. De kabel is bovengronds aangesloten op de Silixa Ultima DTS sensor. De geïnstalleerde AH-DTS kabel is van het originele ontwerp zoals deze is beschreven in paragraaf 4.3.1.

Het opwarmdeel van de kabel is aangesloten op een transformator die de opwarmkabel van (elektrische)stroom voorziet. Tijdens het opwarmen van de AH-DTS kabel is er 30,3 W energie

per strekkende meter nodig voor de opwarming van de kabel. Niet de gehele lengte van de AH-DTS kabel wordt hierbij opgewarmd.



Figuur 12 Schematische weergave van de meetopstelling van de AH-DTS kabel

Er zijn twee AH-DTS metingen beschikbaar op een moment dat de put actief water onttrekt. Daarnaast zijn er drie AH-DTS metingen beschikbaar op een moment dat de put niet actief water onttrekt. Bij de metingen is de AH-DTS kabel voor een periode van 90 minuten opgewarmd.

Qflow stroomsnelheid meting

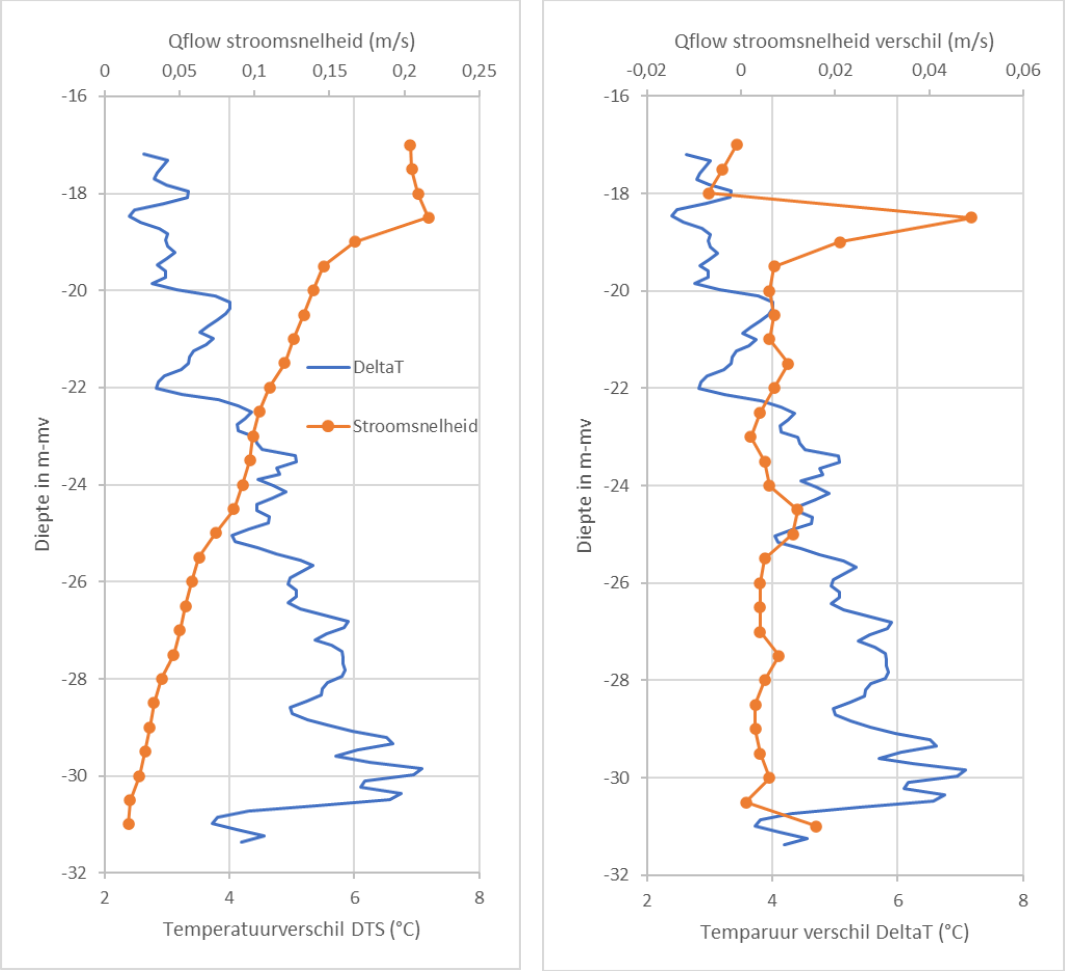
Met een afpompproef is de stroomsnelheid en stromingsverdeling over de diepte in de betreffende onttrekkingsput gemeten op 20 december 2017 door het bedrijf Qflow. Deze meting is uitgevoerd na regenaratie van de pompput. Na het uitvoeren van de Qflow meting is de glasvezelkabel geplaatst in de pompput.

Voor de Qflow meting is de stroomsnelheid over de diepte bepaald en is de snelheidsverandering per filtersectie bepaald. . Deze meetgegevens worden in de volgende paragraaf vergeleken met de meetresultaten uit de AH-DTS meetopstelling.

6.1.2 Resultaten

In Figuur 13 is de AH-DTS temperatuurverschil (verschil tussen temperatuur met en zonder opwarmen) afgezet tegen de stroomsnelheid over de diepte (linker figuur) en de variatie in stroomsnelheid over de diepte (afgeleide van de snelheid, rechter figuur) die in de Qflow meting zijn bepaald.

De Qflow meting laat een duidelijke toename van stroming over de diepte zien, waarbij de hoogste stroomsnelheden gemeten worden hoger in de pompput Dit is te verklaren door de aanwezigheid van de pomp, die zorgt voor de stroming van water.. De mate waarin deze stroomsnelheid toeneemt varieert over de diepte. Op dieptes met relatief grof materiaal in de ondergrond kan water gemakkelijker richting het filter toestromen, wat resulteert in hogere stroomsnelheden.
Op hoofdlijnen laat het temperatuurverschil, gemeten met de glasvezelkabel, een afname zien naar mate je hoger in de put komt. Het temperatuursverschil is kleiner naar mate de kabel meer gekoeld wordt. Deze koeling vindt plaats bij hogere stroomsnelheden
In de temperatuursmeting zijn echter meer fluctuaties zichtbaar over de diepte in vergelijking met de stroomsnelheidsmeting. Dit kan enerzijds verklaard worden door een kleinere meetresolutie, van 0,129 m, bij de temperatuursmeting tegenover een resolutie van 0,5 m bij de stroomsnelheidsmeting. Opvallend zijn de grotere temperatuurverschillen naar mate we dichter aan maaiveld komen. Een voorbeeld is te zien tussen 22 en 20 m-mv. Hier neemt de koeling van de kabel af ondanks dat de stroomsnelheid wel toeneemt.



Figuur 13 Vergelijking DTS temperatuurverschil en Qflow stroomsnelheid metingen

6.1.3 Discussie

Globaal genomen laten de gemeten temperatuurmetingen over de diepte een beeld zien dat aansluit bij de verwachting. Echter zien we op delen van het diepte traject dat de temperatuur toeneemt met de diepte. De kabel wordt hier dus minder goed gekoeld ondanks dat de stroomsnelheid wel is toegenomen. Waarom er op deze trajecten minder koeling plaatsvindt is

onbekend. In de beoordeling van deze resultaten zijn meerdere mogelijke processen benoemd die dit effect mogelijk veroorzaken. Dit betreffen ongetoetste hypothese uit een brainstorm sessie die nader onderzocht moeten worden:

- Variaties in de opbouw van de kabel: De gebuikte AH-DTS kabel is met de hand vervaardigd. Al is het niet de verwachting, kunnen variaties in de opbouw van de kabel doorwerken in de mate waarin de kabel kan worden beïnvloed door de omgeving. Om dit effect uit te sluiten moet gebruikt worden van een kabel met een gegarandeerde opbouw. De nieuw ontwikkelde kabel die in deze rapportage wordt gepresenteerd is daarvoor goed geschikt.
- Turbulentie in de stroming: mogelijk treedt er turbulentie in de stroming binnen de put op waardoor de koeling van de kabel minder efficiënt is. Of een dergelijk proces optreedt is onbekend en zal nader onderzocht moeten worden.
- Variaties in stroming binnen de put: mogelijk is de stroming binnen de put minder uniform dan gedacht en varieert deze over de dwarsdoorsnede. Dit kan bijvoorbeeld door een sterke instroom vanuit 1 richting. Of een dergelijk proces optreedt is onbekend en zal nader onderzocht moeten worden.
- Variaties in temperatuur van het instromend water: mogelijk varieert de temperatuur van het instromende water met de diepte en zorgt dit ook voor de variatie in de afkoeling van de kabel. Dit kan onderzocht worden door te kijken naar de passieve metingen van de kabel (dus zonder opwarming). Het is de inschatting dat deze effecten beperkt zijn
- Lekstroom via de omstorting: mogelijk wordt er water verloren uit de buis dat via de omstorting naar boven stroomt en daar weer in de buis terug komt. De Qflow metingen laten hier echter niets van zien. Of een dergelijk proces optreedt is onbekend en zal nader onderzocht moeten worden.

De gevonden resultaten leveren nieuwe vragen op en brengen mogelijk relevante processen wat betreft stromingen in een drinkwaterput in beeld, waar wij ons tot nu toe niet van bewust zijn. Om deze metingen volledig te kunnen duiden en grip te krijgen op deze fysische processen in een drinkwaterput, is nader onderzoek noodzakelijk. Het advies is dit te doen aan de hand van metingen in een gecontroleerde (laboratorium) omgeving.

6.2 Grondwaterstroming rondom de put

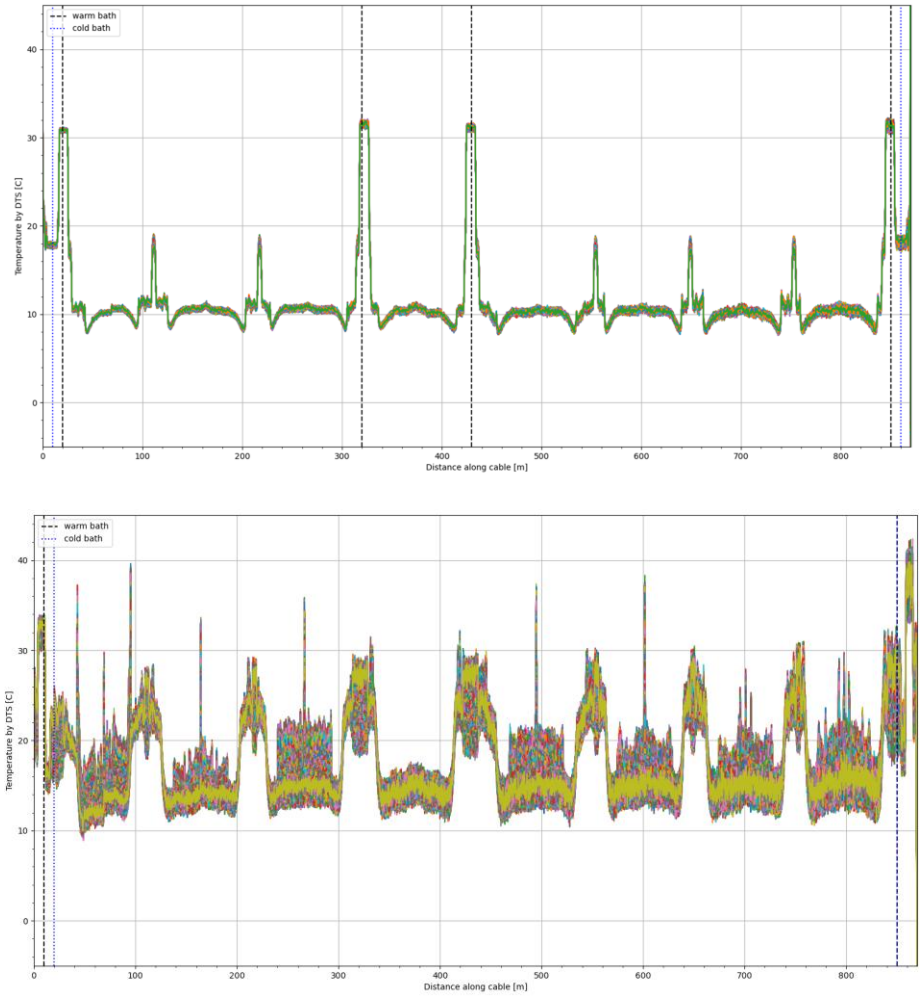
In mei 2021 is er een uitgebreide meetcampagne uitgevoerd waarbij alle aanwezigen glasvezelkabels in de pilotopstelling van ‘t Klooster zijn gemeten met als doel om stroming rondom de pompput inzichtelijk te maken. Tijdens het analyseren van de data is echter gebleken dat deze op een te grove resolutie zijn gemeten, waardoor de vertaling van temperatuur naar stroomsnelheid niet mogelijk is. In juni 2023 is een nieuwe meetronde uitgevoerd. Voorafgaand aan de meting is gebleken dat een deel van de glasvezelkabels niet meer blijkt te werken. De glasvezelkabels die nog wel werken, zijn vervolgens gemeten voor een situatie met pompput aan en voor een situatie met pompput uit.

Naast dat de DTS-unit zelf een temperatuur meet, is het ook mogelijk om zelf de temperatuur te bereken. Hiervoor zijn onder andere twee kalibratiebaden nodig en de gemeten Stokes en Anti-Stokes. Het zelf berekenen van de temperaturen heeft als voordeel dat het een hogere nauwkeurigheid heeft en de gebruiker meer controle geeft tijdens het kalibreren van de temperatuurprofielen.

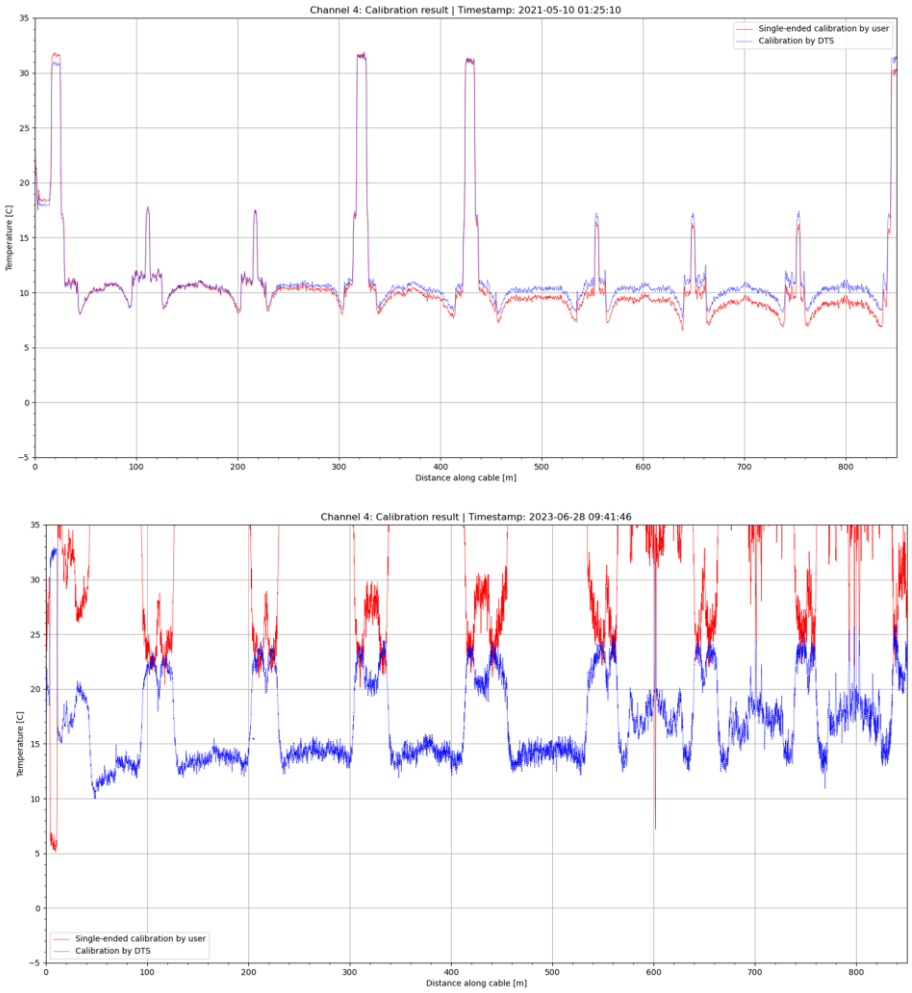
6.2.1 Resultaten

In Figuur 14 zijn de gemeten temperaturen van de DTS-unit voor een deel van de dataset gemeten in 2021 en de dataset gemeten in 2023 weergegeven. Ook zijn de locaties van de kalibratiebaden aangegeven. De temperatuurprofielen in 2021 laten een mooi constant temperatuurprofiel zien, waarbij de kalibratiebaden duidelijk herkenbaar zijn. De temperatuurprofielen in 2023 laten een tegenovergesteld beeld zien. Deze metingen bevatten

heel veel ruis, waardoor de kalibratiebaden niet herkenbaar meer zijn en daardoor ook de gemeten temperaturen heel veel ruis bevatten. Deze grote hoeveelheid ruis maken het berekenen en kalibreren van de temperatuurprofielen onmogelijk (Figuur 15).



Figuur 14 Gemeten temperatuurprofielen door de DTS-unit voor 2021 (boven) en 2023 (beneden)



Figuur 15 Berekende temperatuurprofielen en gemeten temperatuurprofielen voor 2021 (boven) en 2023 (beneden)

6.2.2 Conclusie en discussie

Doordat er in 2021 op een te grove resolutie is gemeten en door de slechte staat waarin de glasvezels verkeerden in 2023 is het niet mogelijk geweest om de onderzoeksvragen wat betreft grondwaterstroming rondom een winput hier te beantwoorden.

In vervolgtrajecten zijn zowel de instellingen van de meetfrequentie en nauwkeurigheid van de DTS-unit als de robuustheid van de glasvezelkabels belangrijke verbeterpunten.

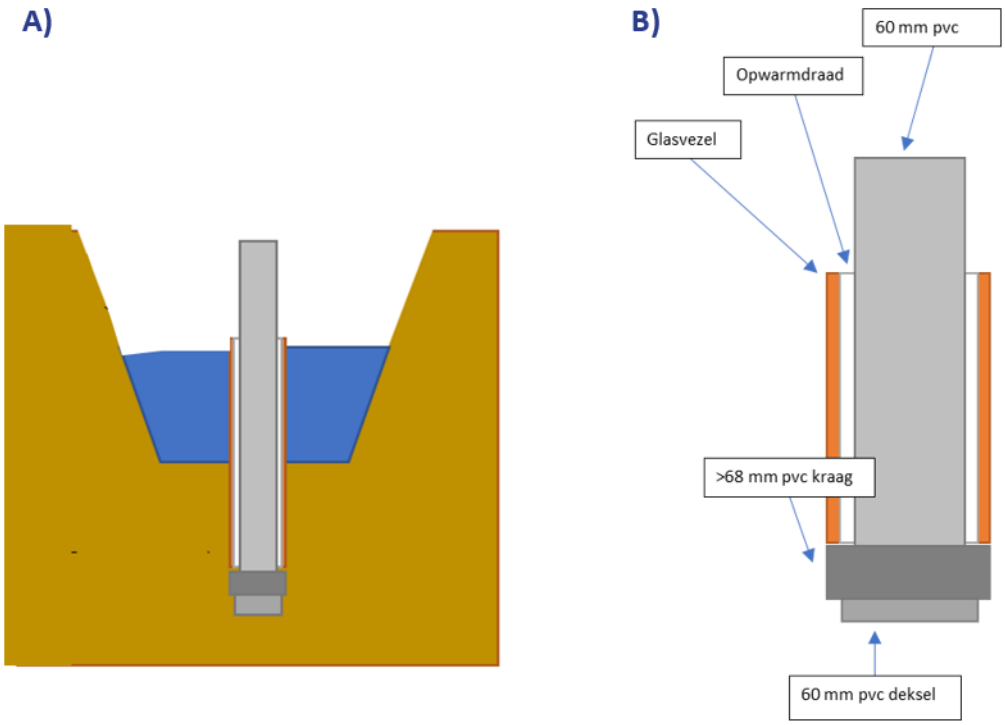
6.3 Grondwater van infiltratie sloot naar aquifer

6.3.1 Opstelling

Op verschillende winlocaties infiltreert Vitens oppervlaktewater om de effecten van de winning gedeeltelijk te compenseren. 't Klooster is één van deze locaties. Vanuit de Veengoot wordt tussen oktober en april water het gebied ingelaten. Dit water infiltreert vervolgens via een stelsel van sloten en rabatten. Vanaf april wordt er geen afvoer gemeten in de Veengoot, waardoor er vanaf april geen infiltratie meer plaatsvindt.
Hoeveel water er gedurende één infiltratieseizoen infiltreert, wordt momenteel geschat als verschil tussen de aan- en afvoer van water in het gebied. Het aantonen van het aandeel oppervlaktewater in de pompputten kan bijdrage aan een beter van de mogelijke risico's en reistijden van het geïnfiltreerde water.

Het doel van dit onderzoek is om te kijken of met behulp van de DTS metingen, dus zonder opwarmevents, de aanwezigheid van oppervlaktewater in de pompputten kan worden aangetoond. Hiervoor is de al aanwezige pilotopstelling uitgebreid door twee verticale palen, te instaleren in de sloot op zo'n 20 meter afstand van de pilotput (Figuur 16 en Figuur 17). De verticale palen zijn omwikkeld met glasvezelkabels om de verticale resolute te verhogen en om de temperatuur van de ondergrond, het slootwater en de luchttemperatuur te monitoren.
De metingen zijn uitgevoerd op twee verschillende kanalen, waarbij voor kanaal 1 de glasvezelkabel eerst naar paal A loopt en vervolgens via de slootbodem naar paal B gaat. Kanaal 2 volgt de omgekeerde route en loopt dus via paal B naar paal A. De metingen zijn uitgevoerd tussen 20 april 2021 en 23 mei 2021. De langste aaneengesloten periode loopt van 29 april 2021 tot 23 mei 2021. Deze periode is daarom gebruikt voor de data-analyse.

Figuur 16 Opstelling van de glasvezelkabels in de sloot.



Figuur 17 Schematische weergave van de verticale palen omwikkeld met glasvezelkabels die geplaatst zijn in de sloot. De opwarmdraad is niet gebruikt.

6.3.2 Resultaten en discussie

Definiëren overgangen lucht-water & water-bodem

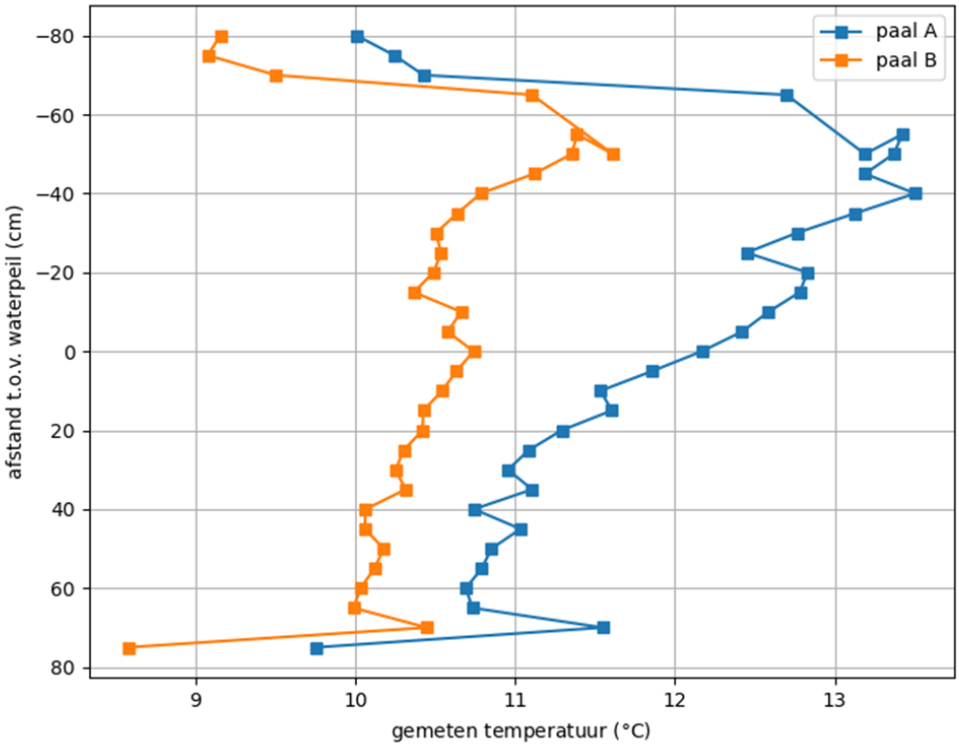
De overgangen tussen lucht-water en water-bodem zijn gedefinieerd met behulp van één opwarmevent wat plaatsvond tijdens de meetperiode (Figuur 18). Lucht geleidt warmte het slechts, waardoor hier de grootste opwarming is te zien. Stromend water geleidt warmte het best, waardoor hier de opwarming van de kabel veel kleiner is vergeleken met lucht. De bodem zit hier tussen in. Een verzadigde bodem, zal de warmte beter geleiden dan lucht, maar minder dan water omdat de stroming hier beperkt is.

Figuur 18 Overgangen lucht - water - bodem

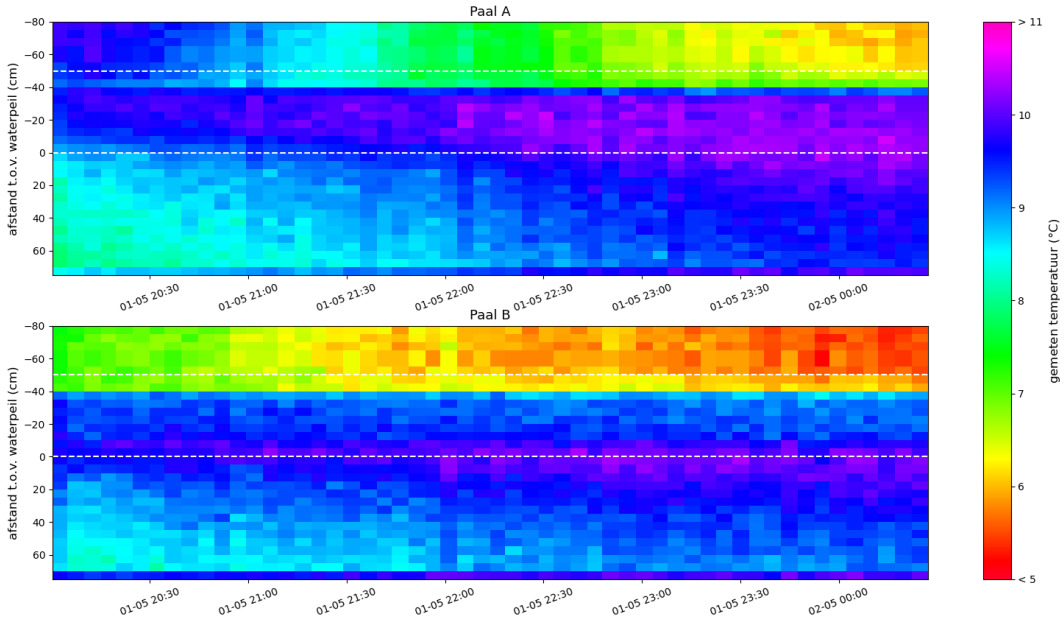
Resultaten kanaal 1

Voor kanaal 1 zijn over de gehele lengte van de paal om de 5 cm temperatuursmeetpunten gedefinieerd. Voor beide palen is een temperatuurprofiel over de diepte gemaakt (Figuur 19) en temperatuurprofielen over zowel diepte als tijd (Figuur 20).

Opvallend zijn de grote temperatuurverschillen tussen paal A en paal B voor kanaal 1. Deze kunnen lokaal oplopen tot een verschil van meer dan 2 °C. Op een afstand van zo'n 200 meter is dit niet erg waarschijnlijk. Daarom is besloten om de data van kanaal 1 niet verder te analyseren en in de rest van het onderzoek de focus te leggen op de temperaturen gemeten op kanaal 2.



Figuur 19 Temperatuurprofiel over de diepte gemeten voor kanaal 1, waarbij de overgang tussen water en bodem op 0 cm zit en de overgang tussen water en lucht op zo'n 45 cm



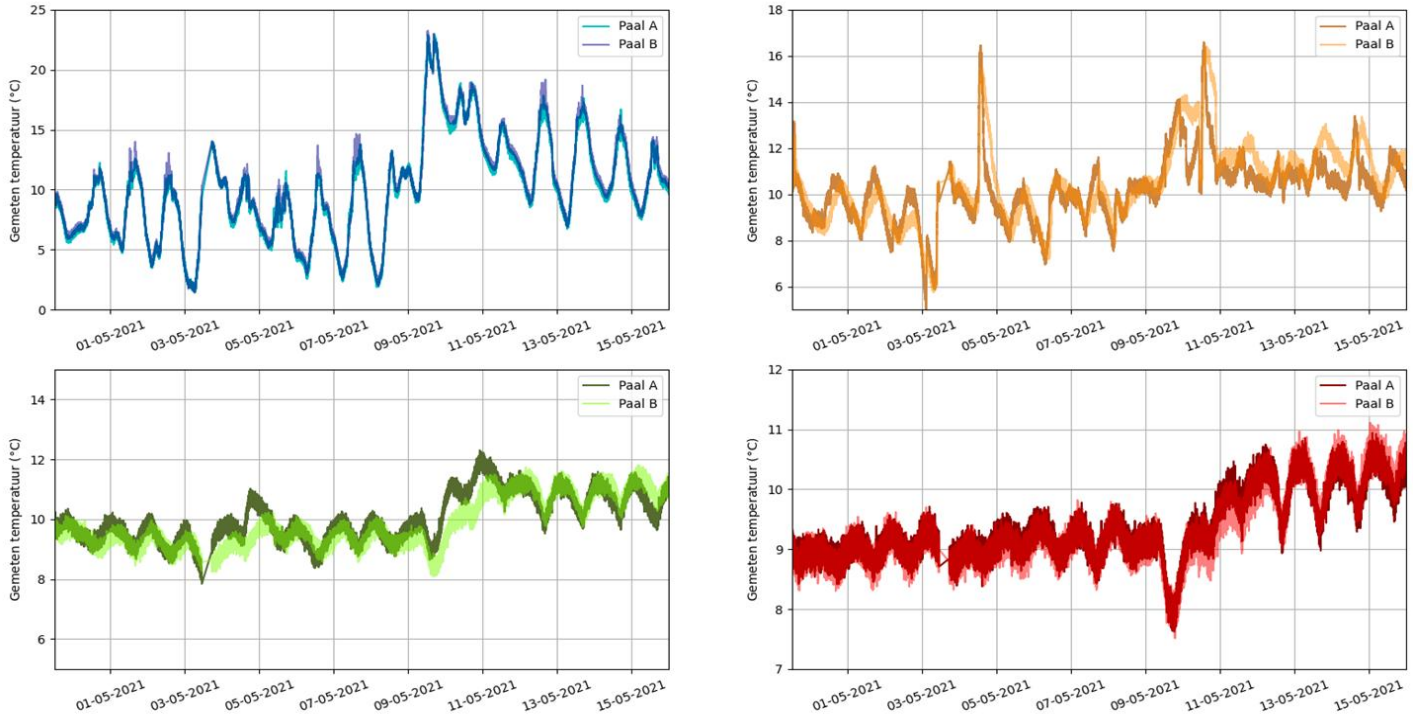
Figuur 20 Temperatuurprofielen van paal A en paal B over de diepte en de tijd

Resultaten kanaal 2

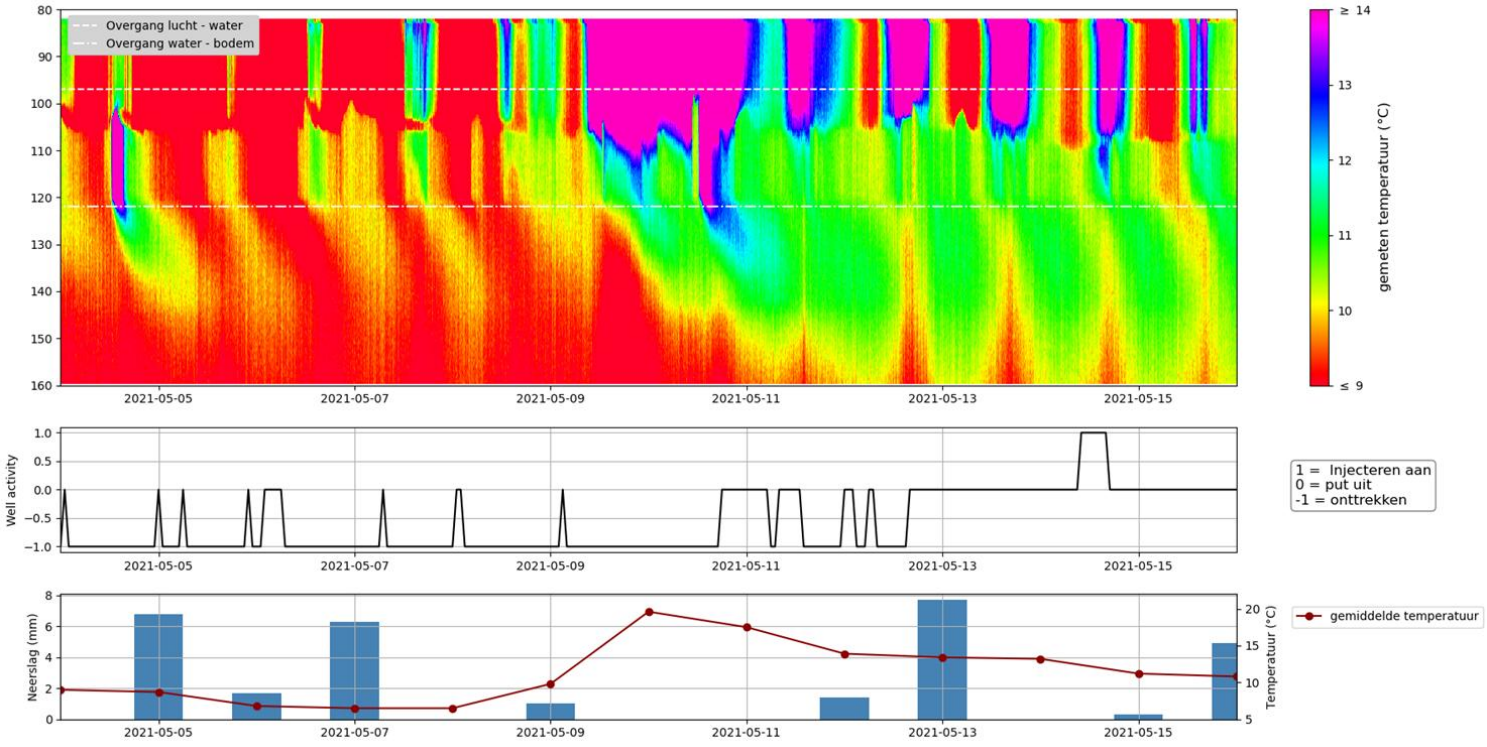
Ook voor kanaal 2 zijn de temperaturen tussen paal A en B vergeleken. Hiervoor zijn voor beide palen op 4 verschillende locaties tijdseries vergeleken. Van deze meetpunten mat 1 locatie de luchttemperatuur, 1 de watertemperatuur en 2 de bodemtemperatuur op twee verschillende dieptes. De resultaten van deze vergelijking zijn weergeven in Figuur 21. Uit dit figuur valt op te maken dat de hoeveelheid ruis in het signaal toeneemt met een toenemende diepte.

Uit dit figuur kan worden opgemaakt dat voor de lucht (De gemeten temperaturen voor de lucht (Figuur 21a) zijn nagenoeg identiek voor beide palen. Grotere temperatuurverschillen tussen paal A en B zijn te zien in Figuur 21 b en c. Vooral in de periode rond 5 mei 2021 en 10 mei 2021 worden duidelijk verschillende temperaturen van het oppervlaktewater gemeten tussen paal A en B. Deze temperatuurverschillen zijn niet terug te zien in de luchttemperatuur. De verklaring moet dus worden gezocht in een ander verschijnsel.
Mogelijk kan het verschil in temperatuur verklaard worden door een lozing van spoelwater in de sloot. Tijdens de productie van drinkwater ontstaat spoelwater, wat eens in de zoveel tijd geloosd wordt op de sloot waarin de metingen zijn uitgevoerd. De momenten van lozing worden echter niet geregistreerd, waardoor het niet met zekerheid te zeggen is, of dit inderdaad de verklaring voor dit verschijnsel is.

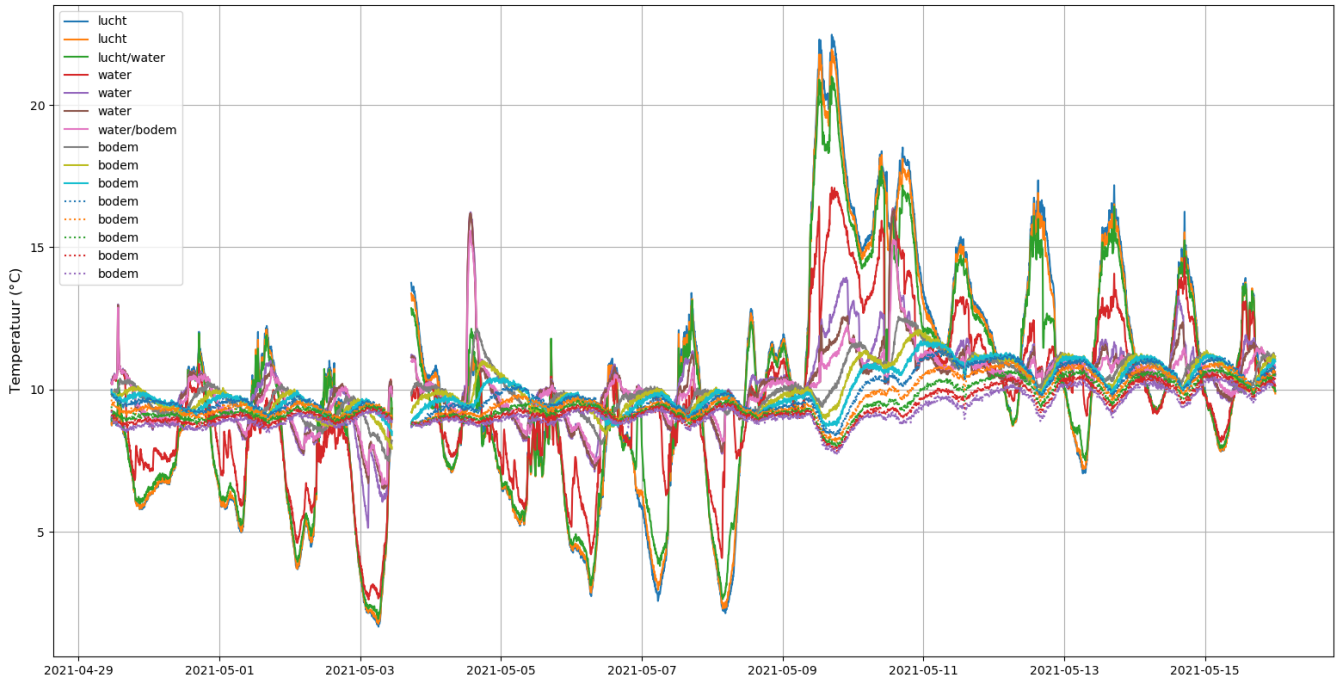
Om te kijken of er een verband kan worden aangetoond tussen de pompactiviteit en het aantrekken van oppervlaktewater door de pomp zijn de temperatuurmetingen in diepte en tijd vergeleken met de pompactiviteit (Figuur 22). Dit figuur laat zien dat luchttemperatuur gedurende de dag toeneemt. Deze toename in warmte wordt, met enige vertraging, doorgegeven aan het water en vervolgens aan de bodem. Waarbij ook de magnitude van de temperatuur steeds verder afneemt (Figuur 23,Figuur 24) met de diepte. Deze patronen worden hoofdzakelijk beïnvloedt door de luchttemperatuur, de pompactiviteit lijkt geen invloed te hebben op deze patronen. Aangezien er gedurende de getoonde periode nauwelijks tot geen veranderingen in temperatuur ontstaan die te relateren zijn aan verandering in pompactiviteit, zoals injecteren of onttrekken.



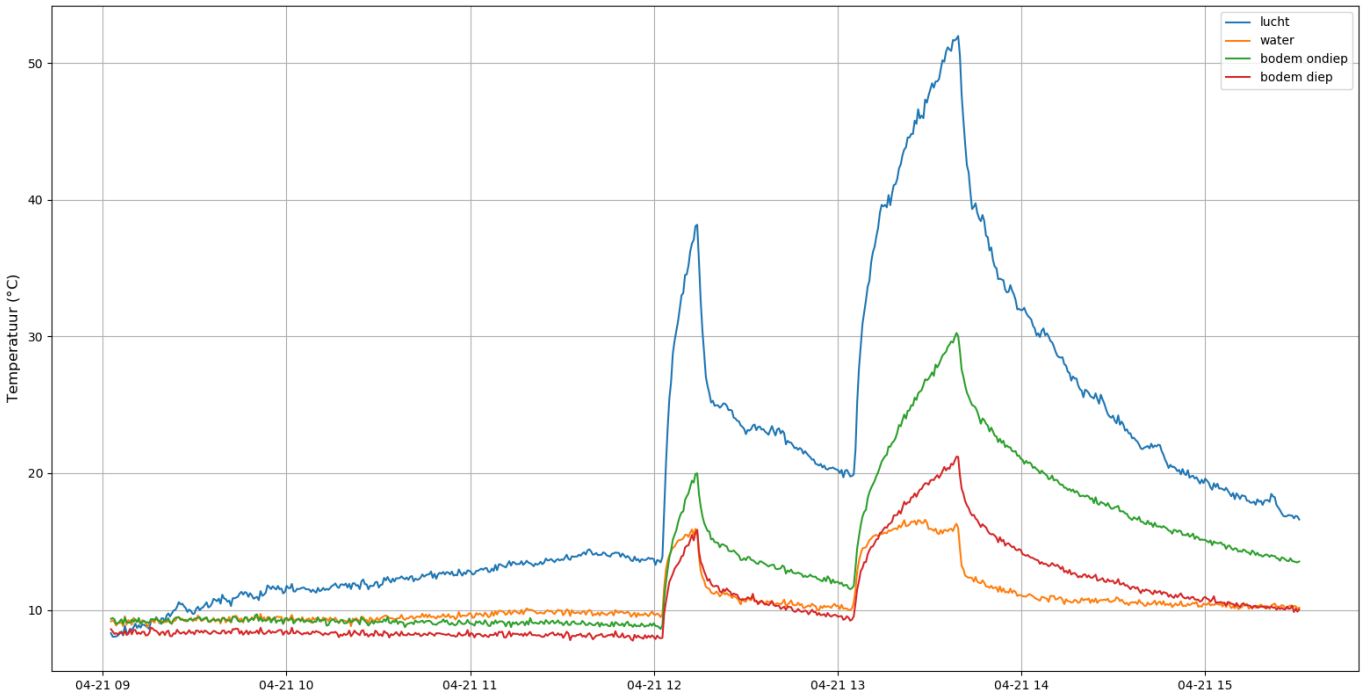
Figuur 21 Vergelijking tussen gemeten temperaturen voor paal A en paal B gemeten in kanaal 2. A) de luchttemperatuur weergeeft, B) de watertemperatuur en C) en D de bodemtemperatuur, waarbij D) dieper zit dan C)



Figuur 22 Gemeten temperaturen tussen 29-4-2021 en 16-5-2021 (boven) de activiteit van pompput 13-18 (midden) en de gemiddelde temperatuur en hoeveelheid neerslag (onder)



Figuur 23 Temperatuurprofielen over tijd voor verschillende dieptes



Figuur 24 Temperatuurprofiel voor verschillende dieptes tijdens een opwarmperiode

6.4 Conclusie en aanbevelingen

Het doel van dit deelonderzoek was om te kijken of met behulp van passieve temperatuurmetingen de aanwezigheid van oppervlaktewater in pompputten kan worden aangetoond.
Tijdens dit onderzoek is gebleken dat er gedetailleerde temperatuurprofielen in zowel de tijd als de diepte te meten zijn met de gebruikte glasvezelkabels. Het bleek daarbij niet eenvoudig om deze temperatuurmetingen door te vertalen naar de aanwezigheid van oppervlaktewater. Om deze stap te maken is meer systeemkennis nodig. De winning wordt namelijk gekenmerkt door grote freatische verlagingen, terwijl de sloot maar van beperkte omvang is, waardoor het niet automatisch een gegeven is dat de verzadigde zone onder de sloot doorloopt tot aan de pompput. Een beter inzicht in de diepte van de verzadigde zone kan bijdragen aan een betere interpretatie van de temperatuurprofielen.

Daarnaast zou het goed zijn om voorafgaand aan de meting een inschatting te maken van de verwachtte infiltratiecapaciteit. Dit in combinatie met aanvullende informatie over bijvoorbeeld waterstanden in de sloot en momenten waarop spoelwater wordt geloosd zijn noodzakelijk om de vertaling van temperatuur naar infiltratie te maken.

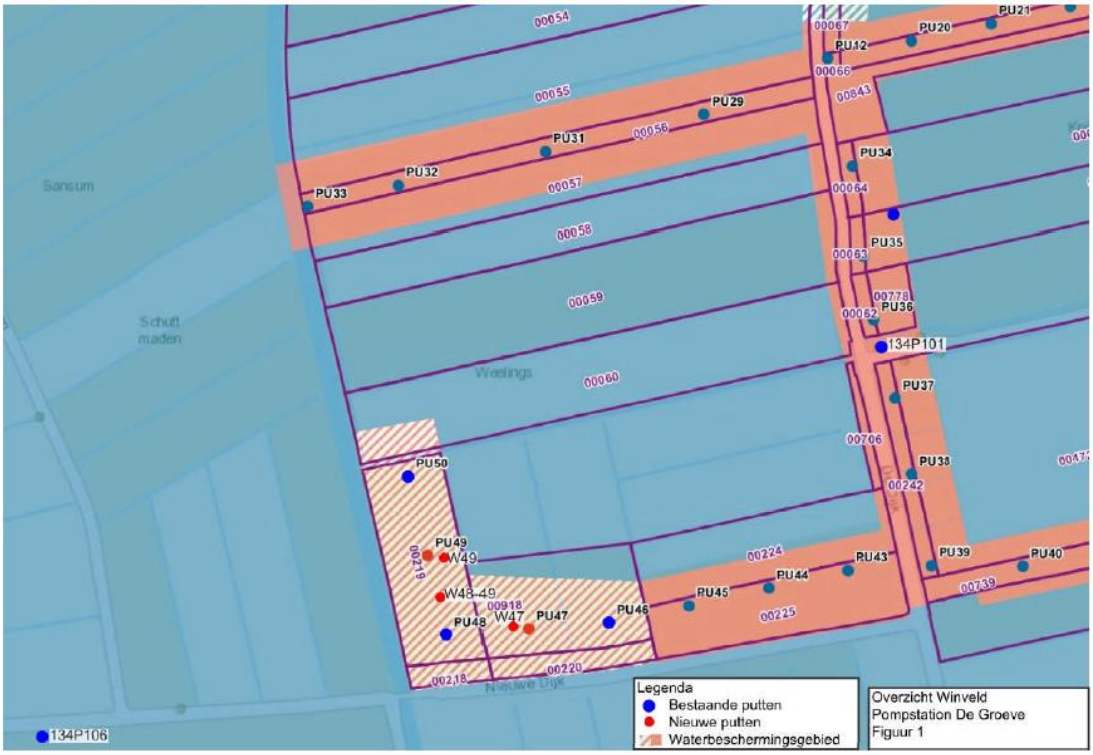
7 Pilot bij Waterbedrijf Groningen – de Groeve

7.1 Introductie

Op de winlocatie de Groeve van Waterbedrijf Groningen zijn meerdere grondwaterputten die water onttrekken tussen de 40 en 80 m beneden maaiveld. In het winveld vindt putverstopping plaats doordat kleine deeltjes de toestroom rondom de filters blokkeren. Om inzicht te krijgen in dit proces zijn AH-DTS glasvezelkabels aan beide zijden van de grondwaterput geïnstalleerd om de toestroming te monitoren. Ook is er op een afstand van 5 m een derde glasvezelkabel geïnstalleerd om meer inzicht te krijgen in de toestroom van het grondwater op afstand van de put en op welke locatie de verstopping precies plaatsvindt.

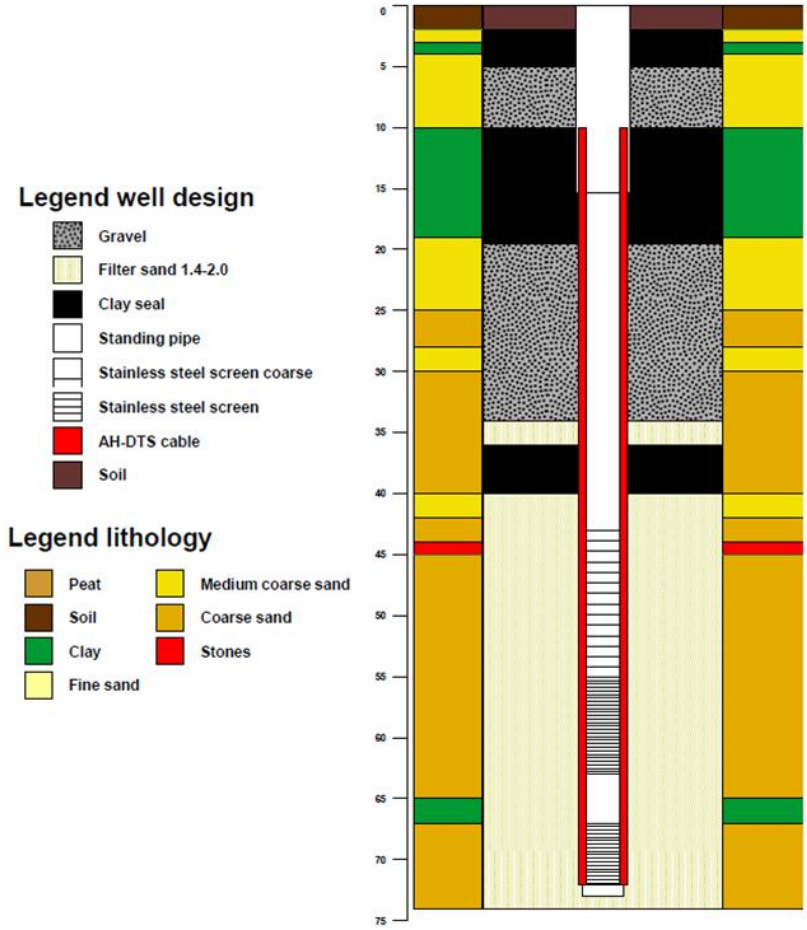
7.2 Opstelling

In 2016 zijn er bij de winning twee nieuwe winputten en drie waarnemingsputten aangelegd (Figuur 25).

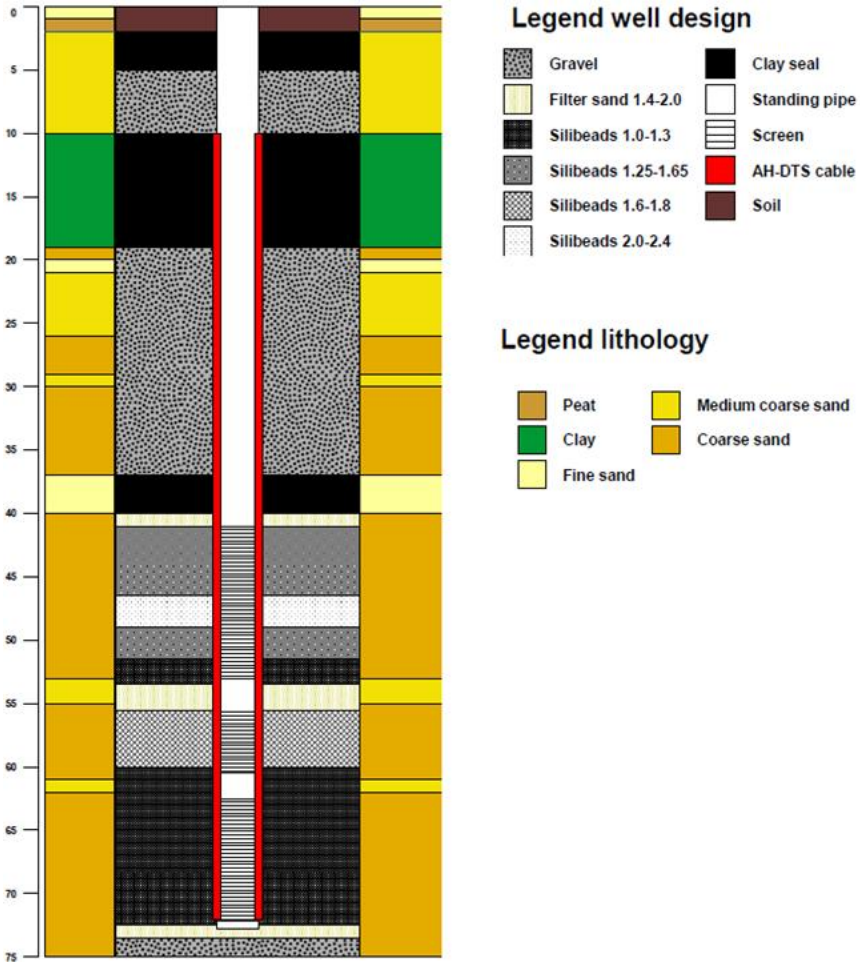


Figuur 25 Overzicht van de nieuwe putten en waarnemingsputten

Bij de winputten is er gekozen voor twee onconventionele ontwerpen om te onderzoeken of deze ontwerpen minder verstoppingsgevoelig zijn. De ontwerpen zijn een hele dunne omstorting waarbij glasparels gebruikt zijn als opvulling in plaats van filtergrind en een winput met rvs filter zonder omstorting. Naast deze ontwerpen zijn de winputten en peilbuizen uitgerust met AH-DTS om te onderzoeken of de putverstopping hiermee gemonitord kan worden. De ontwerpen van beide winputten zijn hieronder weergegeven (Figuur 26; Figuur 27).



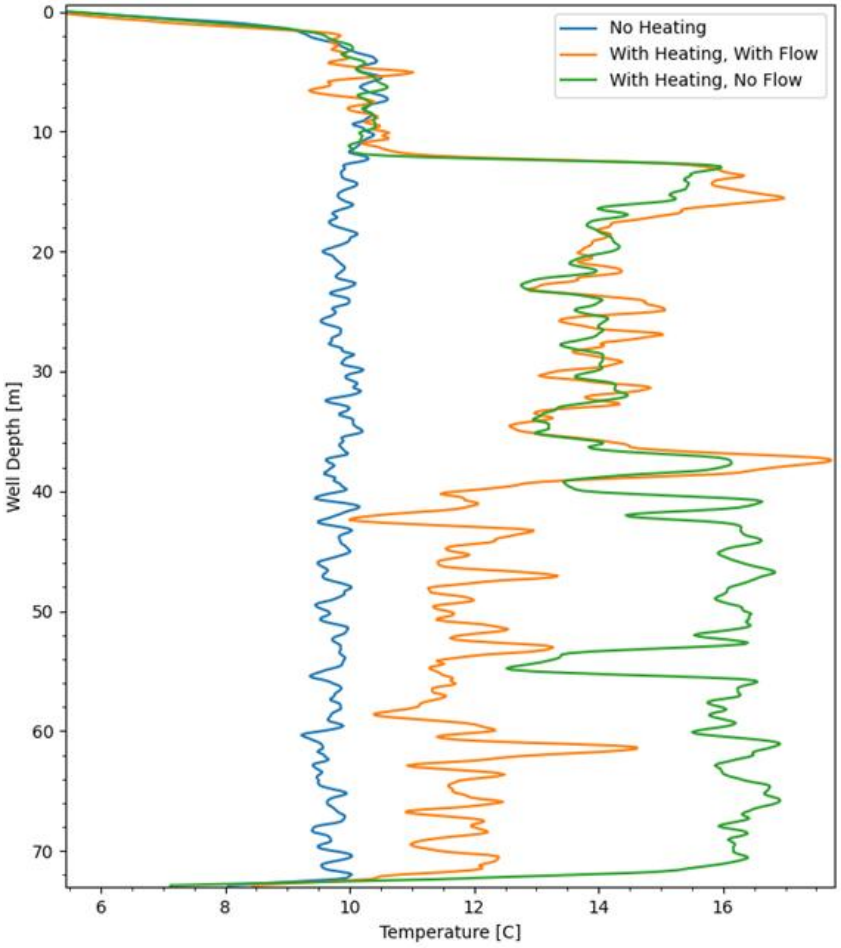
Figuur 26 Ontwerp nieuwe Winput 48



Figuur 27 Ontwerp nieuwe Winput 47

7.3 Resultaten en discussie

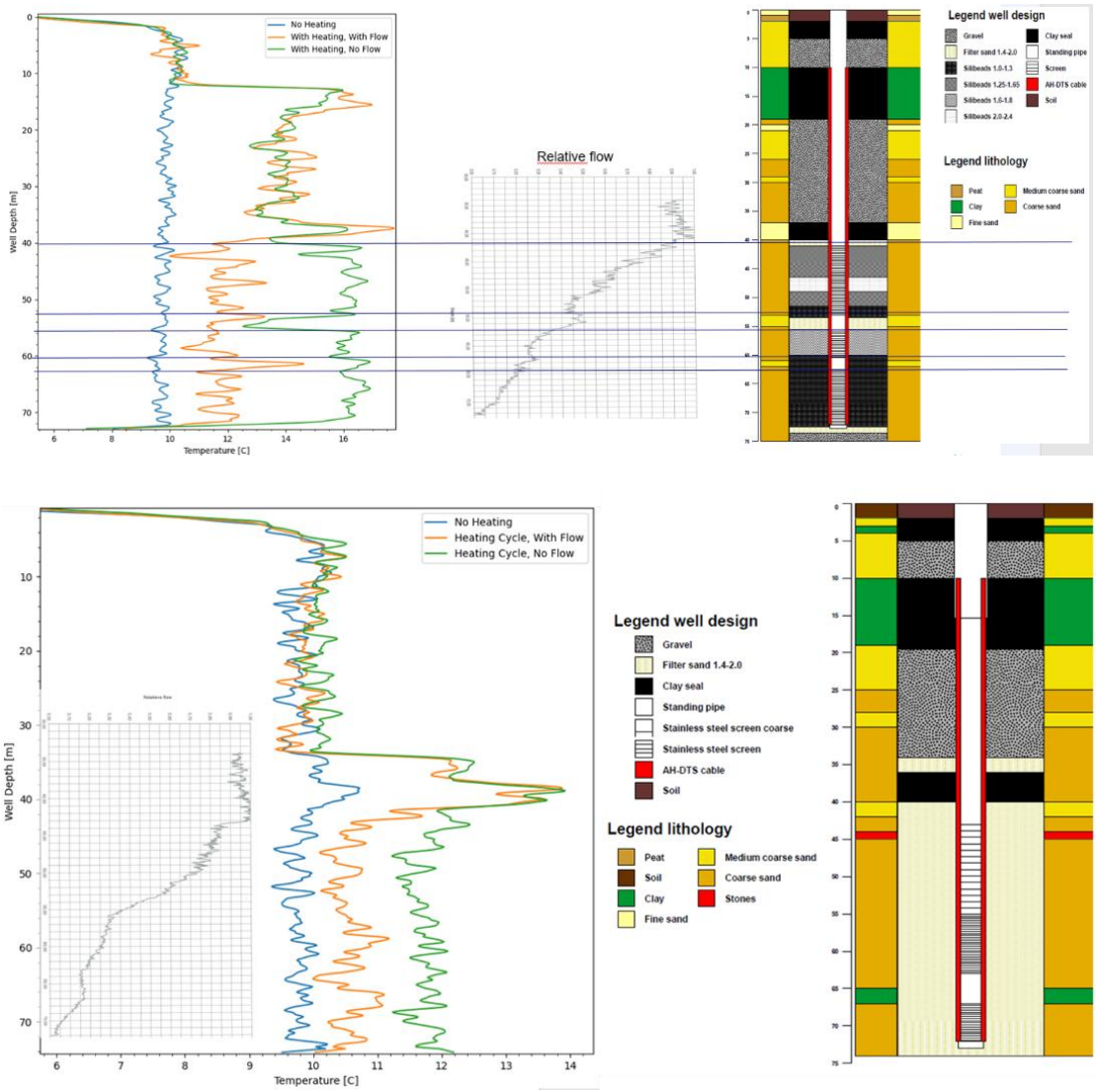
Tijdens het de onderzoeksperiode is er 3 keer gemeten aan de AH-DTS kabels. Hieronder een voorbeeld van een meting uit van winput 47 (Figuur 28). Er is gemeten in drie situaties, een keer zonder verhitting, een keer met verhitting toen de winput uitstond en een keer met verhitting toen de winput in bedrijf was. Door te kijken naar het verschil tussen de situaties met verhitting kan een uitspraak gedaan worden over de toestroming naar de winput. Hoe groter het verschil hoe groter de toestroom.



Figuur 28 Resultaten AH-DTS in Winput 47

Tijdens het aanleggen van de winputten is een flowmeting uitgevoerd langs het filter. Daarmee is een beeld gekregen van het aandeel grondwater per diepte interval op de totale onttrekking. Indirect is dat een maat voor de snelheid over die diepte naar de put tijdens onttrekking. Door deze te vergelijken met de AH-DTS metingen kan een uitspraak gedaan worden over de mate waarin toestroom naar de put te meten is.

In het onderstaande figuren (Figuur 29) is dat gedaan voor beide winputten. Het is te zien dat ter hoogte van de stukken waar geen toestroming is, de blindstukken, het verschil tussen wel en geen verhitting kleiner is. Ook lijkt het erop dat er onderscheid gemaakt kan worden tussen de verschillende omstortingen die zijn toegepast. Dit geeft vertrouwen dat eventuele putverstopping ook waarneembaar zal zijn. Hiervoor kan gekeken worden naar opéénvolgende metingen en vergelijken hoe de toestroming verandert. Vooralsnog zijn er geen putontstoppingen bekend in deze putten. Dit kan een gevolg zijn van de goede prestaties van het nieuwe ontwerp.



Figuur 29 Resultaten van de flow-metingen, metingen met AH-DTS kabels en putontwerp naast elkaar gezet voor Winput 48 (boven) en Winput 47 (beneden)

Helaas bleek er tijdens het project onvoldoende tijd beschikbaar bij Waterbedrijf Groningen om de andere metingen ook uit te werken en onderling te vergelijken. Uit reguliere metingen van putverstopping blijkt dat deze winputten nauwelijks verstoppen. Het is dan ook maar de vraag of daadwerkelijk verschillen zijn waar te nemen.

8 Discussie

Technische werkpakketten
In de technische werkpakketten van dit TKI-project zijn de volgende technieken ontwikkeld om grondwaterstroming te kunnen meten met opwarmbare glasvezelkabels:

- Een **speciale glasvezelkabel** met opwarmoptie ontwikkeld om grondwaterstroming nauwkeurig te meten
- Een **aanstuurkast** waarmee de opwarming van de kabels en de temperatuurmetingen kunnen worden aangestuurd in wisselende configuraties
- **Scripts** waarmee verkregen temperatuurdata naar grondwaterstromingsinformatie kan worden vertaald

De keuze voor een **speciale glasvezelkabel** is altijd een afweging tussen de temperatuurgevoeligheid voor grondwaterstroming, invloed van rotatie, treksterkte en buitenmantelsterkte. De ontworpen kabel is, naar ons idee, de beste afweging tussen deze factoren en de praktische haalbaarheid, en heeft uiteindelijk volgens modellering een gevoeligheid van ca. 0.046 ℃ als gevolg van draaiing, en levert een verschil in opwarming op van 4.49 ℃ tussen 0.01 m/dag grondwaterstroming en 20 m/dag.

De nauwkeurigheid van de metingen met de AH-DTS is in de orde van 0.1 ℃. Dit betekent dat de gevoeligheid voor draaiing niet significant is. De gevoeligheid leidt er wel toe dat de methode waarschijnlijk vooral toepasbaar is bij hogere grondwaterstroming, zoals rond grondwaterputten, omdat daar de relatieve temperatuurverschillen groter zullen zijn.

Met de **aanstuurkast** kunnen van afstand meetschema's worden doorlopen, waarbij de individuele glasvezelkabels kunnen worden bemeten met een bepaalde meetfrequentie en nauwkeurigheid, en tegelijkertijd een schema van opwarming. De aanstuurkast is ingebouwd in een aanhanger, die van wingebied naar wingebied kan worden gereden. Hiervoor is gekozen omdat een continue real-time monitoring van de winputten niet zinvol wordt geacht omdat de verschillen optreden over periodes van maanden tot jaren. Het verplaatsen van de aanhanger en installatie bij verschillende wingebieden is kosteneffectiever.

Met de opgeleverde **scripts** kunnen de temperatuurdata worden omgezet naar grondwaterstromingsinformatie. Hiervoor moet wel de exacte kabelconfiguratie en opwarmschema's in de excel meta-file in worden gevoerd. Er zou in de toekomst een verbeterslag over de scripts kunnen worden gedaan door professionele programmeurs om de foutgevoeligheid te verkleinen en de doorloopsnelheid te verhogen en mogelijk enkele routines te automatiseren. De documentatie in de scripts staat toe dat dit in de toekomst mogelijk is.

Pilot-studies
In de pilot-studie werkpakketten zijn de AH-DTS kabels getest op verschillende casestudies bij wingebied 't Klooster van Vitens en De Groeve van Waterbedrijf Groningen. Deze waren:

1. Meten stroming in de drinkwaterput – 't Klooster (Vitens)
2. Meten stroming rondom drinkwaterput – 't Klooster (Vitens)
3. Meten van infiltratie uit een infiltratiesloot – 't Klooster (Vitens)
4. Meten stroming naar de drinkwaterput – De Groeve (Waterbedrijf Groningen)

1. Meten stroming in de drinkwaterput – 't Klooster (Vitens)
Momenteel is het testen van de prestatie van een drinkwaterput een omslachtige handeling omdat eerst de pomp moet worden uitgebouwd alvorens een flow-meting kan worden gedaan

waaruit blijkt welke dieptetrajecten minder presteren. De voorgestelde methode met een opwarmglasvezelkabel zou kunnen worden uitgevoerd met een druk op de knop.
De in deze pilotstudie uitgevoerde metingen met AH-DTS aan stroming in de drinkwaterput toont overeenkomsten met een flow-meting die de werkelijke stroming in de winput meet. Echter, de temperatuurmetingen lijken ook te worden beïnvloed door andere factoren. Dit komt naar voren uit het feit dat het temperatuurverschil in de diepte varieert, terwijl de verwachting is dat de stroming altijd toeneemt met toenemende hoogte in de put omdat er steeds meer water uit het filter toestroomt. Mogelijke verklaringen zijn o.a. turbulentie, imperfecties in de toegepaste (experimentele) kabel of variaties in de temperatuur van het toestromende water. Deze invloeden moeten eerst beter worden begrepen voordat de methode kwantitatief kan worden ingezet. Geadviseerd wordt om de methode onder gecontroleerde omstandigheden in een schaalexperiment in een laboratorium te testen en verder te ontwikkelen.

2. Meten stroming rondom drinkwaterput – 't Klooster (Vitens)
De metingen rondom de drinkwaterput zijn verricht met reeds geïnstalleerde kabels. De nieuw ontwikkelde kabels konden ivm met vertraging in de planning door regelgeving niet op tijd worden geïnstalleerd. Bij de metingen met de reeds geïnstalleerde kabels bleek enerzijds dat goede communicatie over te gebruiken meetfrequenties essentieel is, zodat uit de metingen een voldoende nauwkeurige opwarmcurve kan worden gedestilleerd. Dit is in een eerste meetronde misgegaan, waardoor het niet mogelijk was om de stroomsnelheid te berekenen.
Bij de tweede meetronde twee jaar later bleek de integriteit van de kabels sinds de installatie achteruit te zijn gegaan. Slechts een deel van de kabels bleek nog goed door te meten, waarschijnlijk doordat plekken waar kabels aan elkaar zijn verbonden met lasjes (zg splices), of de connectoren met de interrogator, zijn in de tussenliggende periode achteruit zijn gegaan. Bij het analyseren van de kabels die nog werkten bleek dat de metingen een te hoge mate van ruis lieten zien om een nauwkeurig grondwaterstromingsprofiel mee te construeren. Blijkbaar laat de robuustheid van de gebruikte kabels te wensen over.
Met de nieuw ontwikkelde kabels hopen we (een deel van) deze robuustheid geborgd te hebben. De integriteit van de lasjes verdient wel aandacht. Deze resultaten zijn waarschijnlijk het gevolg van de experimentele kabel. Immers, glasvezelkabels onder de grond worden in de hele wereld gebruikt voor dataverkeer en zijn zeer robuust.

3. Meten van infiltratie uit een infiltratiesloot – 't Klooster (Vitens)

Ook bij het meten van de infiltratie bleken er opvallende onverklaarbare verschillen in de metingen tussen twee kabels aangesloten op twee verschillende kanalen te zitten. Hierdoor zijn de metingen van slechts één kabel geanalyseerd. De kabels in de sloot waren net voor de metingen geïnstalleerd. De grote verschillen zijn daardoor hoogstwaarschijnlijk niet te verklaren door de achteruitgang van de gebruikte glasvezelkabels, maar moet eerder gezocht worden in de robuustheid van de gebruikte interrogator (meetkast) of de koppeling door de connectoren. Tijdens de verschillende meetrondes is gebleken dat deze met de tijd de metingen steeds minder nauwkeurig werden, wat resulteerde in slechte resultaten.
Dit geeft aan dat er naast verbeterpunten omtrent de robuustheid van de metingen met de kabels ook aandacht moet worden besteed aan onderhoud van de Interrogator.

De individuele metingen laten zeer verklaarbare patronen zien in de tijd tussen de verschillende mediums waarin de kabels zijn geïnstalleerd, namelijk lucht, water en (verzadigde) grond. Helaas kunnen pompactiviteiten, en mogelijk daarmee samenhangende toename in infiltratie, niet worden teruggezien in de temperatuurdata. Het is niet duidelijk of dat komt doordat de metingen niet voldoende nauwkeurig zijn of omdat er geen (verzadigde) connectiviteit tussen de infiltratiesloot en het grondwater is.
De metingen zijn verklaarbaar en veelbelovend. Toepassing op een locatie met beter begrip van de connectiviteit tussen het oppervlaktewater en de onttrekkingskegel is belangrijk om de

techniek te testen en kwantitatief toe te passen. Een mogelijk geschikte locatie zou een winveld met infiltratiepanden zijn.

4. Meten stroming naar de drinkwaterput – De Groeve (Waterbedrijf Groningen)

De metingen bij Waterbedrijf Groningen van glasvezelkabels in de putomstorting laten duidelijk patronen zien in temperatuurverschil, die te relateren zijn aan de grofheid van de sedimenten en het putontwerp. Voor de nieuwe putten zijn er tot op heden geen putverstoppingen waargenomen wat mogelijk ook een gevolg is van de goede prestaties van het nieuwe ontwerp. De resultaten zijn wel dermate robuust dat de verwachting is dat eventuele putverstoppingen over de tijd wel kunnen worden gemeten met deze methode.

Algemeen

De methode van het meten en monitoren van grondwaterstroming rondom drinkwaterputten met behulp van AH-DTS kabels is veelbelovend., In dit project zijn belangrijke stappen gezet naar een brede toepassing hiervan. Toch moeten er nog enkele stappen worden gezet om de techniek robuust te maken en verder te automatiseren. Een belangrijk aandachtspunt is de integriteit van de splices en connectoren naar de interrogator over de tijd. Na enkele jaren bleek het onmogelijk enkele kabels door te meten, en andere kabels bleken een toename van ruis te vertonen of vertoonden onverklaarbare verschillen tussen kanalen. Er was gekozen voor één lange kabel die meerdere putten in één keer door kan meten. Met kortere afzonderlijke kabels voor iedere put, betere splices en connectoren in een robuuste droge en schone behuizing is de verwachting dat deze problemen niet optreden. Daarnaast is een aandachtspunt dat de interrogator regelmatig wordt onderhouden, zodat deze met de benodigde nauwkeurigheid blijft meten.

De verwachting is dat als deze uitdagingen zijn opgelost de kabels een robuust instrument zijn om eenvoudig en frequent de putintegriteit te bemeten. Een eerste stap hiervoor die in dit project is gezet is de nieuw ontwikkelde kabel die naar verwachting meer robuust is. Met de ontwikkelde opwarmkast in een aanhanger kunnen verschillende winvelden in vaste intervallen worden doorgemeten, en met de scripts kunnen de temperatuurmetingen naar grondwaterstroming worden vertaald.

De pilot-studies zijn deels door CoVID-19 en deels door veranderde regelgeving rondom vergunningen (stikstof) op een alternatieve manier ingevuld. Hierdoor zijn de nieuw ontwikkelde kabels niet geïnstalleerd en bemeten tijdens dit project, maar inmiddels zijn de nieuw ontwikkelde kabels wel geïnstalleerd in twee nieuwe drinkwaterputten van Vitens Als alternatief zijn laboratoriumproeven uitgevoerd om de kabel te testen. De resultaten hiervan worden in een vervolgproject over het meten van grondwaterstroming- en richting gedeeld.

Toepassing in de praktijk bleek een uitdaging, en voor bijvoorbeeld de stroming in de put en de infiltratieproeven zijn gecontroleerde schaalproeven in een laboratorium een belangrijke volgende stap om de resultaten uit deze technieken te kwantificeren.

Toelichting scripts

A.1 Toelichting Script 1 – Extractie data uit Silixa files

De temperatuurgegevens die worden geanalyseerd om grondwaterstromingsberekeningen te maken, worden verkregen met behulp van een DTS-glasvezelondervrager. Voor dit project is gebruik gemaakt van de Ultima-S interrogator, vervaardigd door Silixa, om de temperatuurgegevens vast te leggen. De oorspronkelijke gegevensuitvoer (onbewerkte gegevens) is in het XML-formaat.

Script 1 is ontwikkeld om de verzamelde xml-bestanden in te lezen en om te zetten in een formaat dat verdere verwerking in de programmeertaal Python mogelijk maakt. De gekozen datastructuur is een dictionary en maakt het mogelijk verschillende objecten (zoals temperatuurreeksen en tijdreeksen) daarin op te slaan en op naam toegankelijk te maken (de gegevens voor Sonde 1 kunnen bijvoorbeeld worden benaderd met behulp van het commando: `dictionary[“Probe 1”]).`

A.1.1 Invoer voor het script:

Wanneer het verzamelen van de te analyseren data is voltooid, worden de verkregen XML-bestanden opgeslagen in een door de gebruiker gedefinieerde map. Om Script 1 uit te voeren is de enige noodzakelijke invoer een lijst met de bestandspaden naar alle XML-bestanden die de gebruiker wil lezen en verwerken.

A.1.2 Werkstappenscript:

Het script begint met het definiëren van de datum en tijd uit de naam van elk xml-bestand (één xml-bestand wordt gemaakt voor elke tijdstap tijdens een meetronde. De exacte datum en tijd waarop het bestand werd opgeslagen, staat in de naam hiervan). De verzamelde datetimes worden vervolgens opgeslagen in een Pandas datetimeindex-formaat, wat eenvoudiger visualisatie en mogelijkheden mogelijk maakt, zoals het selecteren van specifieke tijdsegmenten in latere stadia.
Hierna opent het script het eerste xml-bestand om de totale meetafstand te bepalen die voor de meetronde is geselecteerd. Met behulp van de totale aantal tijdstappen en de totale meetafstand bepaalt het script de grootte van lege 2D Numpy-arrays die vooraf zijn toegewezen om de temperatuur-, Stokes- en Anti-Stokes-gegevens op te slaan. Lege 1D Numpy-arrays voor gegevens van Probe 1 en 2 en de referentietemperatuur worden in dit stadium ook vooraf toegewezen met behulp van het totale aantal tijdstappen.
Het script gaat vervolgens verder met het invullen van de lege 2D- en 1D-arrays door elk XML-bestand te herhalen en de informatie te extraheren die bij elke vooraf toegewezen array hoort.
Tijdens het draaien van dit script, wordt er ook gecontroleerd op beschadigde bestanden. De bestandsnamen hiervan worden opgeslagen in een aparte lijst.

A.1.3 Resultaten script:

Wanneer het script is uitgevoerd, worden de volgende objecten gemaakt:
- Temperature: 2D numpy array, waarbij de rijen overeenkomen met tijdstappen en de kolommen met afstandsmarkeringen op de glasvezelkabel.
- Stokes: 2D numpy-array met dezelfde structuur als temperatuur.
- Anti-Stokes: 2D numpy-array met dezelfde structuur als temperatuur.
- Probe 1: 1D numpy-array met één meting in het warme/koude kalibratiebad per tijdstap.
- Probe 2: 1D numpy-array met één meting in het warme/koude kalibratiebad per tijdstap.
- Reference temperature: 1D numpy-array met één temperatuurmeting in de interrogator behuizing per tijdstap.

- Distance: 1D numpy-array met de afstandsmarkeringen langs de kabel waarvoor een meting beschikbaar is.
- Datetimes: 1D panda's datetimeindex met de datum en tijd van elke meting.
- User Acquisition Time: de meetduur (in seconden), vereist voor de kalibratiestap.
- Corrupted files: een lijst met de bestandspaden naar beschadigde bestanden.

De bovenstaande objecten zijn allemaal opgeslagen in een library en toegankelijk via de syntaxis beschreven in paragraaf 3.1. De dictionary wordt vervolgens opgeslagen in het pickle-format, waardoor de grootte wordt gecomprimeerd en het delen vervolgens weer gebruiken van deze data eenvoudiger wordt.

A.2 Toelichting script 2 – Kalibratie van de DTS data

In script 2 wordt de kalibratie van de DTS data uitgevoerd. Silixa geeft zelf ook een temperatuurwaarde die hoort bij de metingen op basis van een interne referentie in de machine zelf. Deze kalibratie is echter vrij onnauwkeurig. Wanneer de meetopstelling is voorzien van kalibratiebaden, kan de data aan de hand van de temperatuur in deze baden nauwkeuriger worden gekalibreerd.
Het script gebruikt hiervoor de zogenaamde ‘single ended’ kalibratiemethode en is gebaseerd op het MATLAB script die ontwikkeld is door de Oregon State University (DTS CTEMPs, 2015). Ondanks dat de kabels in onze pilotlocaties altijd aan beide zijdes zijn aangesloten op de DTS (dus op twee kanalen), blijkt uit onze ervaring dat bij de kabellengtes van deze pilots een ‘double ended’ kalibratie niet tot een beter resultaat leidt.

In het script wordt elke kabel dus vanuit weerzijde gekalibreerd. Dit betekent dat er voor elke locatie langs de kabel dus twee temperatuurwaardes beschikbaar zijn. In een later stap kan worden gekozen van welk kanaal de kalibratietemperatuur wordt gebruikt voor het specifieke segment van de kabel. In de regel geldt dat dit de kalibratie is vanuit het kanaal waarbij het segment het dichts is gelegen op de kabel.

Kader Kalibratiebaden
De inzet van kalibratiebaden in de meetopstelling geeft de mogelijkheid tot een nauwkeuriger resultaat. Voor het meten van grondwaterstroming is dit noodzakelijk. Een meetopstelling is daarom idealiter voorzien van 2 kalibratiebaden. Deze kalibratiebaden hebben een constante temperatuur waarbij één bad een lagere temperatuur dan het gemiddelde meetbereik heeft, en één bad een hogere temperatuur dan het gemiddelde meetbereik. De temperatuur van de kalibratiebaden worden middels een 2^{e} sensor (PT100) gemonitord. De meetkabel wordt voor een aanzienlijk deel in elk kalibratiebad geplaatst. Het beste resultaat wordt verkregen wanneer dit gebeurd voor zowel een segment aan het begin als aan het einde van de glasvezelkabel. Het beschreven kalibratiescript in deze rapportage gaat uit van deze opzet van de kalibratiebaden.

Omdat het kalibratiebad een constante en bekende temperatuur heeft kan het segment van de kabel dat door dit bad loopt worden ingezet als kalibratiesegment. In de vertaling van het meetsignaal naar temperatuur kan het kalibratiescript de constanten zo aanpassen dat zo goed mogelijk wordt aangesloten bij deze ijksegmenten.

A.2.1 Input voor het script:

- Locatie van de bestanden die met script 1 zijn gegenereerd.
- Locatie en naam van de MetaFile Excelbestand. Uit dit bestand worden de volgende tabbladen gebruikt:
  - ‘DTS_cali_loc’: locatie van de kalibratiebaden

- Splice_loc: Locatie van bekende splices in de kabel

A.2.2 Werkstappen script:

In het script worden de volgende stappen doorlopen:

- De benodigde functies worden aangeroepen en gemaakt. Er wordt een doelmap aangemaakt waar het resultaat wordt weggeschreven. Deze map wordt aangemaakt op 1 niveau hoger dan de map waarin de input staat. (regel 43 - 56)
- Voor elk kanaal worden de kalibratiesegmenten opgehaald uit de metafile en als parameter ten behoeve van de kalibratie opgeslagen. Het kalibratiesegment is dat deel van de kabel dat in één van de kalibratiebaden is gelegen. (regel 57 - 138)
- Loop die langs alle files in de inputdirectory gaat. Aan het begin van de loop wordt de data die nodig is uit de files gehaald. Het gaat om distance, datetimes, antistokes, stokes, de referentietemperaturen van de kalibratiebaden (trefs) en de temperatuur van de kabel volgens de DTS unit (tempC). (regel 140 - 169)
- Wanneer nodig (dus wanneer er in de metafile een splicelocatie is opgegeven) wordt in het volgende segment een splicecorrectie toegepast. Bij een splice ontstaat een soms een sprong in het signaal. Deze sprong wordt rechtgetrokken. Deze stap dient voor het daadwerkelijke kalibreren plaats te vinden. Er wordt een figuur gemaakt om de correctie te controleren. (regel 171 - 207)
- Voor elke file wordt hier de data voor de kalibratiesegmenten opgehaald (regel 209 - 336).
- De referentietemperatuur van de kalibratiebaden wordt met een moving average gemiddeld. Dit heeft als doel dat er in de kalibratie niet wordt gepoogd de kleinschalige verschillen in de referentietemperatuur te benaderen. Aansluitend worden de juiste baden aan de juiste parameters gekoppeld.(regel 339 - 354)
- Definiëren van ‘lege’ parameters waar in de vervolgstappen de resultaten kunnen worden weggeschreven (355 - 364).
- De eigenlijke kalibratieslag waarbij wordt gefit op 3 parameters. Het resultaat is de parameter calTemp. De kalibratie is uitgevoerd conform de methode zoals is beschreven door Hausner et al. 2011 [4] (regel367 - 389).
- Er worden parameters aangemaakt voor een controle grafiek en deze grafiek wordt geprint (regel 391 - 408).
- Het resultaat wordt weggeschreven (regel 412 - 420).

A.2.3 Resultaten script:

In het weggeschreven resultaat worden alle inputparameters meegenomen en bewaard. Er wordt een gekalibreerde temperatuur als parameter toegevoegd met de naam calTemp.

A.3 Toelichting script 3

In script 3 wordt de data teruggebracht tot de segmenten van de DTS kabel die ook daadwerkelijk de meetlocaties bevatten. Alle meetgegevens die zijn verzamelt over stukken van de kabel die alleen de verschillende meetlocaties verbinden worden hiermee verwijderd. De data wordt per meetlocatie apart opgeslagen in een parameter.

A.3.1 Input voor het script:

- Locatie van de bestanden die met script 2 zijn gegenereerd.
- Locatie en naam van de MetaFile Excelbestand. Uit dit bestand worden de volgende tabbladen gebruikt:
  - ‘DTS_channel_loc’: segmenten op de kabel van de werkelijke meetlocaties

[4] Hausner, M.B.; Suárez, F.; Glander, K.E.; van de Giesen, N.; Selker, J.S.; Tyler, S.W. Calibrating single-ended fiber-optic raman spectra distributed temperature sensing data. *Sensors* **2011**, *11*, 10859–10879.

A.3.2 Werkstappen script:

In het script worden de volgende stappen doorlopen:

- De benodigde functies worden aangeroepen en gemaakt. Er wordt een doelmap aangemaakt waar het resultaat wordt weggeschreven. Deze map wordt aangemaakt op 1 niveau hoger dan de map waarin de input staat. (regel 44 - 61)
- De lijst met bestanden wordt opgehaald en er wordt een loop gestart die door alle bestanden loopt (regel 67 - 70).
- Uit de naam van het bestand wordt bepaald welk kanaal het betreft. Voor dat kanaal wordt een parameter aangemaakt die alle metadata voor dat specifieke kanaal bevat (dataframe). Het vindt hierbij dus ook welke locaties er op dat kanaal zitten. (regel 71 - 74)
- De naam van de file die moet worden opgehaald wordt gemaakt en de data wordt ingelezen. Er wordt een kopieparameter aangemaakt van de metafile data die op dat moment door de loop gaat om straks de resultaten voor die specifiek rij uit de metafile te halen. De afstand en datum/tijd wordt opgehaald uit de data als parameter (regel 78-84).
- Er wordt een loop gestart die door de kolommen van de selectie metafile loopt en hier de informatie uit ophaalt die nodig is (afstanden op de kabel van die specifiek locatie). In de afstand data van de DTS wordt gezocht naar de min en max die in de metafile is opgeven. (regel 85 - 92)
- De naam van de locatie die op dat moment in de loop zit wordt opgehaald en de tijddatum wordt opgeslagen in de dictonary waarin de resultaten gaan komen. Er is nog een optie om ook de XY locaties mee te nemen naar deze dictonaryfile.(regel 93-98)
- Er wordt een loop doorlopen waarin voor elke tijdstap de kalibratietemperatuur (calTemp), de afstand en de temperature volgens de DTS (cTemp) worden beperkt tot de de range tussen de min en max uit de bovengaande loop. Dit wordt als nieuwe parameters met de naam van de locatie in de totaal dictonary opgeslagen (regel 99 - 102)
- Er worden een 3tal grafieken gegenereerd om het resultaat, indien gewenst, te controleren. Dit kan naar eigen hand worden gezet of niet actief gemaakt. (regel 104 - 121)
- Aan het eind wordt het resultaat in 1 dictonary (Total_data_DTS) opgeslagen (regel 124 - 129)

A.3.3 Resultaten script:

Eén dictonary file met alle resultaten van het script. Per meetlocatie is een parameter calTemp, cTemp, datetimes en distance opgenomen.

A.4 Toelichting script 4

A.4.1 Input voor het script:

- Locatie van de bestanden die met script 3 zijn gegenereerd.
- Locatie en naam van de MetaFile Excelbestand. Uit dit bestand worden de volgende tabbladen gebruikt:
  - ‘Heatexper_wellact’: Overzicht van de opwarmexperimenten met daarbij de activiteiten van de winput zelf en de winputten in de omgeving
  - Channeltolocation: De koppeling tissen de kanalen en de meetlocaties. Ook wordt hier een shift opgegeven voor de juiste verschuiving tussen de kabel die naar beneden en omhoog gaat bij elke meetlocatie

A.4.2 Werkstappen script:

- De benodigde modules worden ingeladen en er is een optie om de heatcurve zoekactie aan of uit te zetten. Ook kan het maken van validatiefiguren worden aan en uit gezet. (regel 26 - 38)
- De benodigde metadata wordt uit de metadatafile gehaald en een directory voor het resultaat wordt aangemaakt (regel 50 - 64).
- Er zijn twee alternatieve functies voor het vinden van een dichtstbijzijnde waarde gemaakt. Er is een functie om de verschuiving van de data te maken om opgaand en neergaande delen van de kabel gelijk te zetten voor elke locatie, met behulp van de shift die is benoemd in de metafile. (regel 68 - 90)
- Elke file wordt apart geopend en worden de bijbehorende kanalen opgezocht. Daarnaast worden de begin datum en tijd en de einddatum en tijd van het bestand opgeslagen als variabele
- Per opwarmexpermint (heatrun) worden de begin en eindtijd opgezocht in de DTS data en de bijebhorende heatcurve wordt als variabele van de opwarmcurve opgeslagen.
- na de opwarmcurve vindt deafkoelcurve plaats. Deze start 30 minuten nadat het opwarmexperiment is gestopt. Deze waarde is eventueel aan het begin van het script aan te passen (130-133)
- Er wordt een loop gemaakt door alle tijdstappen om de juiste shift te vinden tussen het opgaande en neergaande deel van de kabel voor elke meetlocatie. Deze shift zorgt ervoor dat het opgaande en neergaande deel van de kabel uiteindelijk over dezelfde lengte lopen. Er wordt voor een duidelijk resultaat een smoothing van het signaal gemaakt. Vervolgens wordt er een waarde verwijderd als het om een oneven aantal waarden gaat. Na deze stappen wordt het midden van het segment gevonden en het deel omhoog en het deel omlaag gedefinieerd (regel 139 - 148)
- De opgegeven shift wordt toegepast en de MSE wordt bepaald. De MSE kan worden gebruikt om te beoordelen of er een verbetering is door de shift. Wanneer aan het begin van het script is gekozen voor validatieplots dan worden deze hier nu gemaakt (regel 150 - 171).
- Het resultaat wordt samengebracht in één dictonary (regel 174-176)
- Dezelfde stappen als voor de heatcurve worden doorlopen voor de coolcurve (regel 179 - 199)
- Er worden figuren gemaakt van het resultaat. Zowel de heatcurve als de coolcurve worden geplot en opgeslagen als 1 figuur (regel 203 - 215).
- In het twee deel van het script wordt de passieve data klaargemaakt, dit is alle gekalibreerde temperatuursdata, en wordt de shift tussen opgaande en neergaande kabels toegepast..
- Van de passieve data wordt op eenzelfde wijze een grafiek gemaakt (regel 255 - 263).
- De resultaten worden weggegeschreven in 2 dictonary bestanden. 1 bestand bevat de opwarmdata. De andere bevat de passieve meetdata (regel 265 - 275).

A.4.3 Resultaten script:

2 dictonaryfiles:

- File die alle opwarm en afkoel curves bevat. Deze data kan worden gebruikt voor het bepalen van grondwaterstroomsnelheden
- File met alle gekalibreerde temperatuursdta data. Deze data geeft zowel informatie over de gemeten grondwatertemperatuur wanneer de kabel is opgewarmd, als wanneer deze niet is opgewarmd

A.5 Toelichting script 5

In dit script wordt vanuit de AH-DTS metingen de ΔT en daaruit de stroomsnelheid berekend. Script 4 is hiervoor de input. In script 4 zijn reeds de opwarm en afkoelcurves bepaald.

A.5.1 Input voor het script:

- Locatie van de bestanden die met script 4 zijn gegenereerd.
- Locatie waar de output mag worden weggeschreven.
- Locatie en naam van de MetaFile Excelbestand. Uit dit bestand worden de volgende tabbladen gebruikt:
  - 'Vel_parameters': Alle constanten die nodig zijn voor de berekening van de stroomsnelheid.

A.5.2 Werkstappen script:

- De benodigde modules worden ingeladen en er is een optie om de heatcurve zoekactie aan of uit te zetten. Ook kan het maken van validatiefiguren worden aan en uit gezet. (regel 26 - 34)
- De constanten worden opgehaald uit de metafile en als parameters gedefinieerd (regel 48 - 93)
- De benodigde metadata wordt uit de metadatafile gehaald en een directory voor het resultaat wordt aangemaakt (regel 100 - 114).
- Formules voor het fitten van de curves worden gedefinieerd (regel 116 - 122).
- Enkele constantes worden berekend (regel 125 - 140).
- In een loop worden de afkoelcurves bepaald en wordt met drie methodes de minimale waarde bepaald (gemiddelde laagste 5 waarden, 2% percentiel laagste waarden, curvefitting evenwichtstemperatuur) (regel 146 - 171).
- In een loop worden de opwarmcurves bepaald en wordt met drie methodes de maximale waarde bepaald (gemiddelde hoogste 5 waarden, 2% percentiel hoogste waarden, curvefitting evenwichtstemperatuur) (regel 175 - 200).
- Het temperatuurverschil (DeltaT) wordt berekend (regel 217 - 243).
- De stroomsnelheid wordt berekend (regel 246 - 280).
- De resultaten worden weggeschreven in 1 dictonary file (regel 282 - 294).

N.B. De curvefitting van python blijkt in de praktijk vaak geen oplossing of onwerkelijke oplossingen te genereren. Daarom is gekozen ook de 2 alternatieve methode in het script op te nemen. Aanbevolen wordt de curvefitting tool van Matlab als alternatief voor de python curve fitting te gebruiken. Deze blijkt in de praktijk wel tot juiste resultaten te komen.

A.5.3 Resultaten script:

1 dictonaryfile:

- File die alle opwarm en afkoel curves bevat en de DeltaT en stroomsnelheden.

A.6 Toelichting MetaFile

In de MetaFile is alle informatie die naast de DTS data als input dient voor de scripts samengebracht. Denk hierbij aan de locaties van de kalibratiebaden, segmenten van de werkelijke metingen en momenten wanneer de kabel werd opgewarmd. De MetaFile is een Excelbestand met op de verschillende tabbladen de specifieke informatie. De MetaFile wordt in de scripts ingelezen en de benodigde metadata wordt uit de betreffende tabbladen uitgelezen. De MetaFile is als volgt opgebouwd:

- Tabblad DTS_channel_loc: Hier wordt de werkelijk meetlocaties beschreven als segmenten op de kabel per kanaal. Ook zijn hier de werkelijke coördinaten van de meetpunten opgenomen. Dit tabblad wordt aangeroepen door script 2 en 3.

- Tabblad DTS_cali_loc: Per kanaal worden de segmenten van de kabel genoemd die door de kalibratiebaden lopen. Dit tabblad wordt aangeroepen door script 2.
- Heatexper_wellact: Overzicht van de opwarmexperimenten met daarbij de activiteiten van de winput zelf en de winputten in de omgeving. Dit betreft een selectie met alleen de zinvolle opwarmexperimenten.
- CHanneltolocation: De koppeling tissen de kanalen en de meetlocaties. Ook wordt hier een shift opgegeven voor de juiste verschuiving tussen de kabel die naar beneden en omhoog gaat bij elke meetlocatie. Ook de coördinaten van de punten zijn hier genoemd.
- Well_activity (optioneel): Datasheet met de activiteit van de winputten in het winveld per uur.
- Useless_heating (optioneel): Overzicht van de opwarmexperimenten met daarbij de activiteiten van de winput zelf en de winputten in de omgeving die niet konden worden gebruikt omdat er ongewenste putschakelingen waren tijdens de experimenten.
- Useless_injection (optioneel): Overzicht van de opwarmexperimenten gedurende de injectie met daarbij de activiteiten van de winput zelf en de winputten in de omgeving die niet konden worden gebruikt omdat er ongewenste putschakelingen waren tijdens de experimenten.
- Vel_parameters: De constanten horende bij de meetkabels die nodig zijn voor de snelheidsberekening.

een uitgebreide beschrijving van de verschillende parameters en variabelen in de verschillende tabbladen, is weergegeven in Bijlage C

B Scripts

B.1 Script 1 XML files van de Silixa worden omgezet naar een bruikbaar Python format

```
"""
Created on Wed Aug 17 12:23:19 2022

@author: pefkos
"""

"""
    Function: extract_dates (used internally in the xml_to_dict function)
        Input: A list containing the filepaths of xml files.
        Output: A list containing the date and time of acquisition for each xml file passed as
input.

    Function: xml_to_dict
        Input: A list containing the filepaths of xml files to be read.
        Output: A dictionary containing:
                - A time series of the Probe 1 temperatures
                - A time series of the Probe 2 temperatures
                - A time series of the DTS unit reference temperatures
                - A 1D numpy array of the distances where the measurements are taken
                - A pandas datetimeindex containing the date and time of each measurement point
                - A 2D numpy array of Stokes data
                - A 2D numpy array of Anti-Stokes data
                - A 2D numpy array of the raw temperatures
                - A list of any corrupted files encountered while reading the data
                - A 1D numpy array containing the measurement duration (user acquisition time)
        Sample call:
                - data_dict = xml_to_dict(filepaths_list)
"""


def extract_dates(filepaths):

    counts = filepaths[0].split("\\")[-1].count(".")

    if (counts == 1) or (counts==2 and "UTC" not in filepaths[0]):
        datetimes = [filepaths[j].split("\\")[-1].split(".")[0].split("_")[-1][:-3] for j in
range(len(filepaths))]
    elif counts == 2 and "UTC" in filepaths[0]:
        datetimes = [filepaths[j].split("\\")[-1].split(".xml")[0].split(".")[0].split("UTC_")[-
1].replace("_"," ") for j in range(len(filepaths))]

    return(datetimes)


def xml_to_dict(dts_filepaths):
    from xml.parsers.expat import ExpatError
    import pandas as pd
    import numpy as np
    import xmltodict
    import time
```

```
    start = time.time()

    probe_1 = np.zeros(len(dts_filepaths))
    probe_2 = np.zeros(len(dts_filepaths))
    reference_temp = np.zeros(len(dts_filepaths))

    dates = extract_dates(dts_filepaths)

    corrupted_files = []

    for j in range(len(dts_filepaths)):
        print("Reading file:",str(j),"of",str(len(dts_filepaths)))

        ##----Read first file separately to obtain distance vector in order to pre-allocate numpy
array sizes----##
        if j == 0:
            with open(dts_filepaths[j]) as file:

                doc = xmltodict.parse(file.read())
                data_dict = doc["logs"]["log"]["logData"]["data"]
                measurement = np.array([data_dict[i].split(',') for i in
range(len(data_dict))]).astype(np.float64)
                distance = measurement[:,0]
                stokes_d = measurement[:,1]
                anti_stokes_d = measurement[:,2]
                temps = measurement[:,-1]

                probe_1[0] = float(doc["logs"]["log"]["customData"]["probe1Temperature"]["#text"])
                probe_2[0] = float(doc["logs"]["log"]["customData"]["probe2Temperature"]["#text"])
                reference_temp[0] =
float(doc["logs"]["log"]["customData"]["referenceTemperature"]["#text"])
                user_acquisition_time =
float((pd.to_datetime(doc["logs"]["log"]["endDateTimeIndex"]) -
pd.to_datetime(doc["logs"]["log"]["startDateTimeIndex"])).seconds)

            temperatures = np.zeros((len(dts_filepaths),len(data_dict)))
            stokes = np.zeros((len(dts_filepaths),len(data_dict)))
            anti_stokes = np.zeros((len(dts_filepaths),len(data_dict)))

            temperatures[0,:] = temps
            stokes[0,:] = stokes_d
            anti_stokes[0,:] = anti_stokes_d

            file.close()

        else:
            ##----Read remaining files----##
            try:
                with open(dts_filepaths[j]) as file:
                    doc = xmltodict.parse(file.read())
                    data_dict = doc["logs"]["log"]["logData"]["data"]
                    measurement = np.array([data_dict[i].split(',') for i in
range(len(data_dict))]).astype(np.float64)
```

```
                stokes[j,:] = measurement[:,1]
                anti_stokes[j,:] = measurement[:,2]
                temperatures[j,:] = measurement[:,-1]

                probe_1[j] = float(doc["logs"]["log"]["customData"]["probe1Temperature"]["#text"])
                probe_2[j] = float(doc["logs"]["log"]["customData"]["probe2Temperature"]["#text"])
                reference_temp[j] =
float(doc["logs"]["log"]["customData"]["referenceTemperature"]["#text"])

                file.close()
            except ExpatError:
                corrupted_files.append(dts_filepaths[j])
                continue

    end = time.time()
    reading_time = end-start

    print("------------------")
    print("Data read successfuly. Data reading time:",str(round(reading_time,3)),"s.")
    print("Datetime range:",str(dates[0]),"to",str(dates[-1]))
    print("Distance range:",str(distance[0]),"m to",str(distance[-1]),"m.")
    print("Temperature values range:",str(np.min(temperatures)),"C
to",str(np.max(temperatures)),"C.")
    print("------------------")

    my_dict = dict()
    my_dict["Temperature"] = temperatures
    my_dict["Probe 1"] = probe_1
    my_dict["Probe 2"] = probe_2
    my_dict["Reference Temperature"] = reference_temp
    my_dict["Stokes"] = stokes
    my_dict["Anti-Stokes"] = anti_stokes
    my_dict["Distance"] = distance
    dates = pd.to_datetime(dates)
    my_dict["Datetimes"] = dates
    my_dict["User Acquisition Time"] = user_acquisition_time
    my_dict["Corrupted files"] = corrupted_files

    return(my_dict)
```

B.2 Script 2 Kalibratie van de data

```
# -*- coding: utf-8 -*-
"""
Created on Thu Aug 15 12:02:44 2019

_____Description_____

Calibrates DTS data using a single ended calibration.
The cable needs 2 sections running through a cold bath and 2 sections running through a
warm bath

----Explanation-
_____

Developed by W. Bakx
date 28/01/2022
_____


_____INPUT_____
The following input is needed for the script to run:
- Location of the dictonary input files containing the uncalibrated data.
- xlsx file containing the metadata
    - In the metadata file a sheet 'DTS_cali_loc' needs to be present containing the segments
of the cable (in meters) that are running through the calibration baths
- Location where the results need to be placed

_____


"""

import matplotlib.pyplot as plt
import numpy as np
import os.path
import pickle
import pandas as pd

#----------------------------------------INPUT
#Location of the SILIXA data - dictonary files (resulting from Manos scripts)
Fileslocation = r"C:\PY_PROJECTS\GroFloMo\klooster_2021\data\20210507\d20210507"
#Location of the metadata XLSX file and sheet containing the calibration sections
meta_cali =
pd.read_excel(r"C:\PY_PROJECTS\GroFloMo\klooster_2021\Meta_input\Klooster_2021_met
adata.xlsx",sheet_name='DTS_cali_loc')
splice_loc =
pd.read_excel(r"C:\PY_PROJECTS\GroFloMo\klooster_2021\Meta_input\Klooster_2021_met
adata.xlsx",sheet_name='splice_loc')
```

```
#---------------------------------------FUNCTIONS
#Function for finding nearest value in array
def find_nearest(array, value):
    array = np.asarray(array)
    idx = (np.abs(array - value)).argmin()
    return array[idx]

# Creating directory and location to save the calirbated result
save_path = os.path.dirname(Fileslocation)
submaploc = 'S2_calibrated'
dircrloc= f'{save_path}\{submaploc}'
#Check if dir already exists otherwise create
if not os.path.exists(dircrloc):
    os.makedirs(dircrloc)

#---------------------------------------Gathering data and making it ready for the process
#Making list of all files in the directory
Fileslist= os.listdir(Fileslocation)

#Getting calibration sections in m along cable (will be transfered in loop to needed index)
nr_channels = np.max(meta_cali['Channel'])
for i in range(len(meta_cali['Channel'])):
    if meta_cali['Channel'][i]==1 and meta_cali['Temperature'][i]=='Passive' and
meta_cali['Segment'][i] == 1:
        cold1 = [meta_cali['Start'][i],meta_cali['End'][i]]
    if meta_cali['Channel'][i]==1 and meta_cali['Temperature'][i]=='Passive'and
meta_cali['Segment'][i] ==2:
        cold2 = [meta_cali['Start'][i],meta_cali['End'][i]]
    if meta_cali['Channel'][i]==1 and meta_cali['Temperature'][i]=='Active'and
meta_cali['Segment'][i] == 1:
        warm1 = [meta_cali['Start'][i],meta_cali['End'][i]]
    if meta_cali['Channel'][i]==1 and meta_cali['Temperature'][i]=='Active'and
meta_cali['Segment'][i] == 2:
        warm2 = [meta_cali['Start'][i],meta_cali['End'][i]]
z1m = [warm1,cold1,warm2]
zVAL1m = cold2
del cold1,cold2,warm1,warm2

for i in range(len(meta_cali['Channel'])):
    if meta_cali['Channel'][i]==2 and meta_cali['Temperature'][i]=='Passive' and
meta_cali['Segment'][i] == 1:
        cold1 = [meta_cali['Start'][i],meta_cali['End'][i]]
    if meta_cali['Channel'][i]==2 and meta_cali['Temperature'][i]=='Passive'and
meta_cali['Segment'][i] ==2:
        cold2 = [meta_cali['Start'][i],meta_cali['End'][i]]
    if meta_cali['Channel'][i]==2 and meta_cali['Temperature'][i]=='Active'and
meta_cali['Segment'][i] == 1:
        warm1 = [meta_cali['Start'][i],meta_cali['End'][i]]
    if meta_cali['Channel'][i]==2 and meta_cali['Temperature'][i]=='Active'and
meta_cali['Segment'][i] == 2:
        warm2 = [meta_cali['Start'][i],meta_cali['End'][i]]
z2m = [warm1,cold1,warm2]
zVAL2m = cold2
```

```
del cold1,cold2,warm1,warm2

for i in range(len(meta_cali['Channel'])):
   if meta_cali['Channel'][i]==3 and meta_cali['Temperature'][i]=='Passive' and
meta_cali['Segment'][i] == 1:
      cold1 = [meta_cali['Start'][i],meta_cali['End'][i]]
   if meta_cali['Channel'][i]==3 and meta_cali['Temperature'][i]=='Passive'and
meta_cali['Segment'][i] ==2:
      cold2 = [meta_cali['Start'][i],meta_cali['End'][i]]
   if meta_cali['Channel'][i]==3 and meta_cali['Temperature'][i]=='Active'and
meta_cali['Segment'][i] == 1:
      warm1 = [meta_cali['Start'][i],meta_cali['End'][i]]
   if meta_cali['Channel'][i]==3 and meta_cali['Temperature'][i]=='Active'and
meta_cali['Segment'][i] == 2:
      warm2 = [meta_cali['Start'][i],meta_cali['End'][i]]
z3m = [warm1,cold1,warm2]
zVAL3m = cold2
del cold1,cold2,warm1,warm2

for i in range(len(meta_cali['Channel'])):
   if meta_cali['Channel'][i]==4 and meta_cali['Temperature'][i]=='Passive' and
meta_cali['Segment'][i] == 1:
      cold1 = [meta_cali['Start'][i],meta_cali['End'][i]]
   if meta_cali['Channel'][i]==4 and meta_cali['Temperature'][i]=='Passive'and
meta_cali['Segment'][i] ==2:
      cold2 = [meta_cali['Start'][i],meta_cali['End'][i]]
   if meta_cali['Channel'][i]==4 and meta_cali['Temperature'][i]=='Active'and
meta_cali['Segment'][i] == 1:
      warm1 = [meta_cali['Start'][i],meta_cali['End'][i]]
   if meta_cali['Channel'][i]==4 and meta_cali['Temperature'][i]=='Active'and
meta_cali['Segment'][i] == 2:
      warm2 = [meta_cali['Start'][i],meta_cali['End'][i]]
z4m = [warm1,cold1,warm2]
zVAL4m = cold2
del cold1,cold2,warm1,warm2

for i in range(len(meta_cali['Channel'])):
   if meta_cali['Channel'][i]==5 and meta_cali['Temperature'][i]=='Passive' and
meta_cali['Segment'][i] == 1:
      cold1 = [meta_cali['Start'][i],meta_cali['End'][i]]
   if meta_cali['Channel'][i]==5 and meta_cali['Temperature'][i]=='Passive'and
meta_cali['Segment'][i] ==2:
      cold2 = [meta_cali['Start'][i],meta_cali['End'][i]]
   if meta_cali['Channel'][i]==5 and meta_cali['Temperature'][i]=='Active'and
meta_cali['Segment'][i] == 1:
      warm1 = [meta_cali['Start'][i],meta_cali['End'][i]]
   if meta_cali['Channel'][i]==5 and meta_cali['Temperature'][i]=='Active'and
meta_cali['Segment'][i] == 2:
      warm2 = [meta_cali['Start'][i],meta_cali['End'][i]]
z5m = [warm1,cold1,warm2]
zVAL5m = cold2
del cold1,cold2,warm1,warm2
for i in range(len(meta_cali['Channel'])):
```

```
    if meta_cali['Channel'][i]==6 and meta_cali['Temperature'][i]=='Passive' and
meta_cali['Segment'][i] == 1:
        cold1 = [meta_cali['Start'][i],meta_cali['End'][i]]
    if meta_cali['Channel'][i]==6 and meta_cali['Temperature'][i]=='Passive'and
meta_cali['Segment'][i] ==2:
        cold2 = [meta_cali['Start'][i],meta_cali['End'][i]]
    if meta_cali['Channel'][i]==6 and meta_cali['Temperature'][i]=='Active'and
meta_cali['Segment'][i] == 1:
        warm1 = [meta_cali['Start'][i],meta_cali['End'][i]]
    if meta_cali['Channel'][i]==6 and meta_cali['Temperature'][i]=='Active'and
meta_cali['Segment'][i] == 2:
        warm2 = [meta_cali['Start'][i],meta_cali['End'][i]]
z6m = [warm1,cold1,warm2]
zVAL6m = cold2
del cold1,cold2,warm1,warm2

#============LOOP for running to all files
for q in range(len(Fileslist)):
    #-------------------------INPUT---------------------------
    # Getting the input
    Filename=Fileslist[q]   #Getting the name of the file processed in the list
    Filenameloc= os.path.join(Fileslocation,Filename) #Creating file loc and name combination
    pickle_data=open(Filenameloc,"rb") # open the file wit pickle
    data = pickle.load(pickle_data) #defining the file as data
    #getting channel
    splitpath=Filename.split('_')
    splitpath=splitpath[1].split('.')
    nbt=len(splitpath)
    pickchannel = int(splitpath[nbt-2])
    #Getting the data from the datafile and create parameters
    distance=data['Distance']
    datetimes=data['Datetimes']
    AntiStokes=data['Anti-Stokes']
    Stokes=data['Stokes']
    tref_1_ruw=data['Probe 1']
    tref_2_ruw=data['Probe 2']
    tref_3=data['Reference Temperature']
    tempC=data['Temperature']
    #check if the matrix is dis,datetime as should be
    if len(tempC[:,1]) == len(distance):
        print('Inputdata was right and not corrected')
    else:
        print('Inputdata not right, correction applied')
        AntiStokes= np.transpose(AntiStokes)
        Stokes= np.transpose(Stokes)
        tempC= np.transpose(tempC)

    #%%doing the splice correction if needed
    Stokes_splitcor=np.copy(Stokes)
    AntiStokes_splitcor=np.copy(AntiStokes)
    for l in range(len(splice_loc.loc[:,"Channel"])):
        if  splice_loc.loc[l,"Channel"] == pickchannel: # limit action to splices on the same
channel of the file that is loaded
            Chdist = splice_loc.loc[l,"Chdist"]# get the distancelocation of the known splice
```

```
        for k in range(len(datetimes)):
            AS_pre_avg=np.average(AntiStokes_splitcor[Chdist-10:Chdist-1,k])
            AS_aft_avg=np.average(AntiStokes_splitcor[Chdist+1:Chdist+10,k])
            AS_corr=AS_pre_avg-AS_aft_avg
            S_pre_avg=np.average(Stokes_splitcor[Chdist-10:Chdist-1,k])
            S_aft_avg=np.average(Stokes_splitcor[Chdist+1:Chdist+10,k])
            S_corr=S_pre_avg-S_aft_avg
            for m in range(len(distance)):
                if m <Chdist:
                    AntiStokes_splitcor[m,k]=AntiStokes_splitcor[m,k]
                    Stokes_splitcor[m,k]=Stokes_splitcor[m,k]
                    #print('geen aanpassing')
                else:
                    AntiStokes_splitcor[m,k]=(AntiStokes_splitcor[m,k]+AS_corr)
                    Stokes_splitcor[m,k]=(Stokes_splitcor[m,k]+S_corr)
                    #print('aanpassing')

        #Check figure
        plt.figure()
        #plt.title('Channel_'+str(pickchannel))
        plt.plot(AntiStokes[:,150],'r',linewidth=0.5,label='AntiStokes')
        plt.plot(AntiStokes_splitcor[:,150],'r--',linewidth=0.5,label='AntiStokes')
        #plt.plot(Stokes[:,150],'g',linewidth=0.5,label='Stokes')
        #plt.plot(Stokes_splitcor[:,150],'g--',linewidth=0.5,label='Stokes')
        #plt.plot(tempC[:,150],'b',linewidth=0.5,label='Calibration by DTS')
        #plt.axvline(x = Chdist, color = 'b')
        #plt.plot(Xcordstref1,Ycordstref1,'g--')
        #plt.legend(('calTemp', 'tempC', 'tref_2','tref_1'),
        #           loc='lower right')
        plt.show()

    #%%getting the calibrationsections from the xlsx and find the needed indexes
    z=[]
    if pickchannel ==1:
        warm1s=find_nearest(distance,z1m[0][0])
        warm1s=np.where(distance ==warm1s)
        warm1e=find_nearest(distance,z1m[0][1])
        warm1e=np.where(distance ==warm1e)
        z.append([int(warm1s[0]),int(warm1e[0])])
        cold1s=find_nearest(distance,z1m[1][0])
        cold1s=np.where(distance ==cold1s)
        cold1e=find_nearest(distance,z1m[1][1])
        cold1e=np.where(distance ==cold1e)
        z.append([int(cold1s[0]),int(cold1e[0])])
        warm2s=find_nearest(distance,z1m[2][0])
        warm2s=np.where(distance ==warm2s)
        warm2e=find_nearest(distance,z1m[2][1])
        warm2e=np.where(distance ==warm2e)
        z.append([int(warm2s[0]),int(warm2e[0])])
        cold2s=find_nearest(distance,zVAL1m[0])
        cold2s=np.where(distance ==cold2s)
        cold2e=find_nearest(distance,zVAL1m[1])
        cold2e=np.where(distance ==cold2e)
```

```
        zVAL=[int(cold2s[0]),int(cold2e[0])]
    elif pickchannel ==2:
        warm1s=find_nearest(distance,z2m[0][0])
        warm1s=np.where(distance ==warm1s)
        warm1e=find_nearest(distance,z2m[0][1])
        warm1e=np.where(distance ==warm1e)
        z.append([int(warm1s[0]),int(warm1e[0])])
        cold1s=find_nearest(distance,z2m[1][0])
        cold1s=np.where(distance ==cold1s)
        cold1e=find_nearest(distance,z2m[1][1])
        cold1e=np.where(distance ==cold1e)
        z.append([int(cold1s[0]),int(cold1e[0])])
        warm2s=find_nearest(distance,z2m[2][0])
        warm2s=np.where(distance ==warm2s)
        warm2e=find_nearest(distance,z2m[2][1])
        warm2e=np.where(distance ==warm2e)
        z.append([int(warm2s[0]),int(warm2e[0])])
        cold2s=find_nearest(distance,zVAL2m[0])
        cold2s=np.where(distance ==cold2s)
        cold2e=find_nearest(distance,zVAL2m[1])
        cold2e=np.where(distance ==cold2e)
        zVAL=[int(cold2s[0]),int(cold2e[0])]
    elif pickchannel ==3:
        warm1s=find_nearest(distance,z3m[0][0])
        warm1s=np.where(distance ==warm1s)
        warm1e=find_nearest(distance,z3m[0][1])
        warm1e=np.where(distance ==warm1e)
        z.append([int(warm1s[0]),int(warm1e[0])])
        cold1s=find_nearest(distance,z3m[1][0])
        cold1s=np.where(distance ==cold1s)
        cold1e=find_nearest(distance,z3m[1][1])
        cold1e=np.where(distance ==cold1e)
        z.append([int(cold1s[0]),int(cold1e[0])])
        warm2s=find_nearest(distance,z3m[2][0])
        warm2s=np.where(distance ==warm2s)
        warm2e=find_nearest(distance,z3m[2][1])
        warm2e=np.where(distance ==warm2e)
        z.append([int(warm2s[0]),int(warm2e[0])])
        cold2s=find_nearest(distance,zVAL3m[0])
        cold2s=np.where(distance ==cold2s)
        cold2e=find_nearest(distance,zVAL3m[1])
        cold2e=np.where(distance ==cold2e)
        zVAL=[int(cold2s[0]),int(cold2e[0])]
    elif pickchannel ==4:
        warm1s=find_nearest(distance,z4m[0][0])
        warm1s=np.where(distance ==warm1s)
        warm1e=find_nearest(distance,z4m[0][1])
        warm1e=np.where(distance ==warm1e)
        z.append([int(warm1s[0]),int(warm1e[0])])
        cold1s=find_nearest(distance,z4m[1][0])
        cold1s=np.where(distance ==cold1s)
        cold1e=find_nearest(distance,z4m[1][1])
        cold1e=np.where(distance ==cold1e)
        z.append([int(cold1s[0]),int(cold1e[0])])
```

```
        warm2s=find_nearest(distance,z4m[2][0])
        warm2s=np.where(distance ==warm2s)
        warm2e=find_nearest(distance,z4m[2][1])
        warm2e=np.where(distance ==warm2e)
        z.append([int(warm2s[0]),int(warm2e[0])])
        cold2s=find_nearest(distance,zVAL4m[0])
        cold2s=np.where(distance ==cold2s)
        cold2e=find_nearest(distance,zVAL4m[1])
        cold2e=np.where(distance ==cold2e)
        zVAL=[int(cold2s[0]),int(cold2e[0])]
    elif pickchannel ==5:
        warm1s=find_nearest(distance,z5m[0][0])
        warm1s=np.where(distance ==warm1s)
        warm1e=find_nearest(distance,z5m[0][1])
        warm1e=np.where(distance ==warm1e)
        z.append([int(warm1s[0]),int(warm1e[0])])
        cold1s=find_nearest(distance,z5m[1][0])
        cold1s=np.where(distance ==cold1s)
        cold1e=find_nearest(distance,z5m[1][1])
        cold1e=np.where(distance ==cold1e)
        z.append([int(cold1s[0]),int(cold1e[0])])
        warm2s=find_nearest(distance,z5m[2][0])
        warm2s=np.where(distance ==warm2s)
        warm2e=find_nearest(distance,z5m[2][1])
        warm2e=np.where(distance ==warm2e)
        z.append([int(warm2s[0]),int(warm2e[0])])
        cold2s=find_nearest(distance,zVAL5m[0])
        cold2s=np.where(distance ==cold2s)
        cold2e=find_nearest(distance,zVAL5m[1])
        cold2e=np.where(distance ==cold2e)
        zVAL=[int(cold2s[0]),int(cold2e[0])]
    elif pickchannel ==6:
        warm1s=find_nearest(distance,z6m[0][0])
        warm1s=np.where(distance ==warm1s)
        warm1e=find_nearest(distance,z6m[0][1])
        warm1e=np.where(distance ==warm1e)
        z.append([int(warm1s[0]),int(warm1e[0])])
        cold1s=find_nearest(distance,z6m[1][0])
        cold1s=np.where(distance ==cold1s)
        cold1e=find_nearest(distance,z6m[1][1])
        cold1e=np.where(distance ==cold1e)
        z.append([int(cold1s[0]),int(cold1e[0])])
        warm2s=find_nearest(distance,z6m[2][0])
        warm2s=np.where(distance ==warm2s)
        warm2e=find_nearest(distance,z6m[2][1])
        warm2e=np.where(distance ==warm2e)
        z.append([int(warm2s[0]),int(warm2e[0])])
        cold2s=find_nearest(distance,zVAL6m[0])
        cold2s=np.where(distance ==cold2s)
        cold2e=find_nearest(distance,zVAL6m[1])
        cold2e=np.where(distance ==cold2e)
        zVAL=[int(cold2s[0]),int(cold2e[0])]
    else:
        print('en nu')
```

```
    #smoothing the trefs
    tref_1=np.convolve(tref_1_ruw,np.ones((10,))/10,mode='same')
    tref_2=np.convolve(tref_2_ruw,np.ones((10,))/10,mode='same')
    #tref_1_plot= pd.concat([tref_1[1]]*distance, ignore_index=True)
    # Setting the correct PT100 measurements to the correct segments
    bath1 = tref_2 # insert the reference temperature belonging to the first calibration range
    bath2 = tref_1 # insert the reference temperature belonging to the second calibration range
    bath3 = tref_2 # insert the reference temperature belonging to the third calibration range
    Tv=np.zeros(len(tref_1))
    T=np.zeros([3,len(tref_1)])
    for i in range(len(tref_1)):
        Tv[i] = tref_1[i] + 273.15 # insert the reference temperature for the validation bath
        T[0,i]= bath1[i] + 273.15
        T[1,i]= bath2[i] + 273.15
        T[2,i]= bath3[i] + 273.15

    #Creating empty parameters for the results with the correct lengths
    ntrace=len(datetimes)
    ltrace=len(distance)
    calTemp=np.zeros([ltrace,ntrace])
    R=np.log(Stokes/AntiStokes)
    g = np.zeros([1, ntrace])
    C = np.zeros([1, ntrace])
    Da = np.zeros([1, ntrace])
    RMSEavg = np.zeros([1, ntrace])
    valRMSE = np.zeros([1, ntrace])
    #-------- Actual calibration script

    for i in range(len(datetimes)):
        A=np.zeros([3,3])
        # Create matrix A (Equation 3)
        A[0,:]=[1,-T[0,i],T[0,i]*np.mean(distance[z[0][0]:z[0][1]])]
        A[1,:]=[1,-T[1,i],T[1,i]*np.mean(distance[z[1][0]:z[1][1]])]
        A[2,:]=[1,-T[2,i],T[2,i]*np.mean(distance[z[2][0]:z[2][1]])]
        if np.linalg.det(A)==0:
            g[0,i]=497.6
            C[0,i]=1.7243
            Da[0,i]=6.38E-5
            del A
        else:
            # Calculate x (Equation 3)
            inva=np.linalg.inv(A)

Q=inva.dot(np.array([T[0,i]*np.mean(R[z[0][0]:z[0][1],i]),T[1,i]*np.mean(R[z[1][0]:z[1][1],i]),T[2,i
]*np.mean(R[z[2][0]:z[2][1],i])]))
            # Solve calibration parameters
            g[0,i]=Q[0]
            C[0,i]=Q[1]
            Da[0,i]=Q[2]
            del Q
            del A
        # Calculate calibrated temperature (Equation 1)
        calTemp[:,i]=np.real(g[0,i]/(R[:,i]+C[0,i]-Da[0,i]*distance)) - 273.15
```

```
    #creating calibrationlines for the checkplot
    Xcordstref1=[0,3900]
    avgtref1=np.average(tref_1)
    Ycordstref1=[avgtref1,avgtref1]
    Xcordstref2=[0,3900]
    avgtref2=np.average(tref_2)
    Ycordstref2=[avgtref2,avgtref2]

    #Checking calibration result graph
    plt.figure()
    plt.title('Channel_'+str(pickchannel))
    plt.plot(calTemp[:,150],'r',linewidth=0.5,label='Single-ended calibration by user')
    plt.plot(tempC[:,150],'b--',linewidth=0.5,label='Calibration by DTS')
    plt.plot(Xcordstref2,Ycordstref2,'g')
    plt.plot(Xcordstref1,Ycordstref1,'g--')
    plt.legend(('calTemp', 'tempC', 'tref_2','tref_1'),
           loc='lower right')
    plt.show()


#
=========================================================================
=======
#    #------------------Saving results
#    data['calTemp']=calTemp
#    # Saving Total2_loc_run, Heat_curves and Cool_curves
#    Exportname = Filename.split('.')
#    Exportname = f'{Exportname[0]}_cali'
#    ExportNameLoc = os.path.join(dircrloc, Exportname)
#    pickle_out=open(ExportNameLoc,"wb")
#    pickle.dump(data,pickle_out)
#    pickle_out.close()
#
#
=========================================================================
=======
```

B.3 Script 3 Selectie van de werkelijke data horende bij de meetlocaties

```
"""
Created on Fri Jan 28 15:31:19 2022

_____Description_____
_____

Script deleting all inrelevant data and creating data per location

_____
_____

Developed by W. Bakx
date 28/01/2022
_____
_____

_____INPUT_____
_____
The following input is needed for the script to run:
- Location of the calibrated data, result from script 2. Alle data files present in the dir will be
processed
- Excel containing the metadata
- Pickle files of all locations containing the X,Y,Z information (needs to be generated for a
different location)
_____
_____

_____OUTPUT_____
_____
- a pickle file for all locations together

_____
_____

@author: bakxw
"""

#Loading modules and packages
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import os.path
import os
import pickle

#----------------------------------------INPUT
#Location of the SILIXA data - dictonary files (resulting from Manos scripts)
Fileslocation = r"C:\PY_PROJECTS\GroFloMo\klooster_2021\data\20210507\S2_calibrated"
#Location of the metadata XLSX file and sheet containing heating data
metaloc =
'C:\PY_PROJECTS\GroFloMo\klooster_2021\Meta_input/Klooster_2021_metadata.xlsx'
```

```
#---------------------------------------FUNCTIONS
#Function for finding nearest value in array
def find_nearest(array, value):
    array = np.asarray(array)
    idx = (np.abs(array - value)).argmin()
    return array[idx]

#---------------------------------------SCRIPT
#getting metadata from file
Meta_locations = pd.read_excel(f"{metaloc}",sheet_name='DTS_channel_loc') # getting the
locations on the channel

# Creating directory and location to save the calirbated result
save_path = os.path.dirname(Fileslocation)
submaploc = 'S3_selected'
dircrloc= f'{save_path}\{submaploc}'
#Check if dir already exists otherwise create
if not os.path.exists(dircrloc):
    os.makedirs(dircrloc)

#--------------CREATING a dictionary for the usefull data----------------
#Making list of all files in the directory
Filelist= os.listdir(Fileslocation)

#Reading metadata on locations and location name and creating variables
Total_data_DTS={} #Defining Dictonary for storage of data per location of interest
for i in range(len(Filelist)):
    #getting channel
    Filename=Filelist[i]
    splitpath=Filename.split('_')
    pickchannel = int(splitpath[1])  # determine the channel
    vars()[f'Meta_loc_ch{pickchannel}'] = Meta_locations.loc[Meta_locations['CH'] ==
pickchannel] #get metadlocdata
    if pickchannel == 0:
        print(f'channel {pickchannel} is not processed')
    else:
        Filenameloc= os.path.join(Fileslocation,Filename) #Creating file loc and name
combination
        pickle_data=open(Filenameloc,"rb") # open the file wit pickle
        data = pickle.load(pickle_data) #defining the file as data
        locdatach=vars()[f'Meta_loc_ch{pickchannel}']
        lenght=len(locdatach)
        distance = data['Distance']
        datetimes = data['Datetimes']
        for k in range(lenght):#Run through all locations on the channel
            locdata =locdatach.iloc[k]
            Maxdis_ruw = locdata['MaxDis']
            Mindis_ruw = locdata['MinDis']
            Maxdis_exact = find_nearest(distance,Maxdis_ruw)
            Mindis_exact = find_nearest(distance,Mindis_ruw)
            MaxLoc = int(np.where(distance == Maxdis_exact)[0])
            MinLoc = int(np.where(distance == Mindis_exact)[0])
            Name = locdata['Name']
```

```
        #Total_data_DTS[f'LocX_{Name}']=vars()[f'LocXYZ_{Name}'][0][0:int(MaxLoc-
MinLoc)]
        #Total_data_DTS[f'LocY_{Name}']=vars()[f'LocXYZ_{Name}'][1][0:int(MaxLoc-
MinLoc)]
        #Total_data_DTS[f'CoordX_{Name}']=vars()[f'LocXYZ_{Name}'][3][0:int(MaxLoc-
MinLoc)]
        #Total_data_DTS[f'CoordY_{Name}']=vars()[f'LocXYZ_{Name}'][4][0:int(MaxLoc-
MinLoc)]
        Total_data_DTS[f'datetimes_{Name}'] = datetimes
        for l in range(len(datetimes)): # run through all timesteps
            Total_data_DTS[f'calTemp_{Name}'] = data['calTemp'][MinLoc:MaxLoc,:]
            Total_data_DTS[f'Distance_{Name}'] = distance[MinLoc:MaxLoc]
            Total_data_DTS[f'cTemp_{Name}'] = data['Temperature'][MinLoc:MaxLoc,:]

       #Checking calibration result graph
        plt.figure()
        plt.title(f'Channel_{pickchannel}_{Name}')
        plt.plot(Total_data_DTS[f'calTemp_{Name}'][:,150],'r',linewidth=0.5,label='Single-
ended calibration by user')
        #plt.plot(data['calTemp'][:,150],'b--',linewidth=0.5,label='Calibration by DTS')
        plt.show()
        #checking correct location
        plt.figure()
        plt.title(f'Channel_{pickchannel}_{Name}')
        plt.plot(data['calTemp'][:,150],'r',linewidth=0.5,label='Single-ended calibration by user')
        #plt.plot(data['calTemp'][:,150],'b--',linewidth=0.5,label='Calibration by DTS')
        plt.show()
        #Checking calibration result graph for splices
        plt.figure()
        plt.title(f'Channel_{pickchannel}_{Name}')

plt.plot(Total_data_DTS[f'calTemp_{Name}'][:,150],Total_data_DTS[f'Distance_{Name}'][:],'r',l
inewidth=0.5,label='Single-ended calibration by user')
        #plt.plot(data['calTemp'][:,150],'b--',linewidth=0.5,label='Calibration by DTS')
        plt.show()

#
=============================================================================
=======
# #------------------Saving results
# Exportname = 'Total_data_DTS'
# ExportNameLoc = os.path.join(dircrloc, Exportname)
# pickle_out=open(ExportNameLoc,"wb")
# pickle.dump(Total_data_DTS,pickle_out)
# pickle_out.close()
#
=============================================================================
=======
```

B.4 Script 4 Opdelen van de meetgegevens in data met en zonder opwarming

"""
Created on Wed Feb 02 08:25:19 2022

_____Description_____
_____

Script adding the heatdata from the metafile and creating selection of the temperaturedata off the heatexperiments.
All other data is kept in the save as in the next step the updown will be corrected also for the raw selection data

_____
_____

_____
_____

_____INPUT_____
_____
The following input is needed for the script to run:
- Location of the dataselection data, result from script 3. Alle data files present in the dir will be processed
- Excel containing the metadata

_____
_____

_____OUTPUT_____
_____
- a pickle file containing all data from the previous script and this script

_____
_____
"""

```
#Loading modules and packages needed for the script
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import datetime as dt
import os.path
import os
import pickle
from copy import copy
#script_path=os.path.dirname(__file__)

Find_heatcurve=True
Print_ON= True # Set to false if you dont want to plot validation plots.

#%% INPUT data LOCATIONS
#Location of the SILIXA data - dictonary files (resulting from SCRIPT 3)
```

```
Fileslocation = r"C:\PY_PROJECTS\GroFloMo\klooster_2021\data\20210507\S3_selected"
#os.path.dirname(__file__)+'\\data'
#Location of the metadata XLSX file and sheet containing heating data
metaloc =
'C:\PY_PROJECTS\GroFloMo\klooster_2021\Meta_input/Klooster_2021_metadata.xlsx'
#os.path.dirname(__file__)+'\Klooster_2021_metadata.xlsx'

# Location of the location data (X,Y,Z coordinates and lenght along cable)
#Locdirectory =  'C:\Py_projects\GroFloMo\klooster_2021\Meta_input\Location_data'

#%% METADATA
#getting metadata from file
Meta_heatwell = pd.read_excel(f"{metaloc}",sheet_name='Heatexper_wellact')
# get predefined shifts and the link between channel and location
chtoloc = pd.read_excel(f"{metaloc}",sheet_name='CHanneltolocation')
# duration cooling curve
dt_cooling = dt.timedelta(minutes = 30)

# Creating directory and location to save the calibrated result
save_path = os.path.dirname(Fileslocation) # creating the path based on the inputlocation
submaploc = 'S4_totalheat'# Name of the export map
dircrloc= f'{save_path}\{submaploc}' #location of the export map

#Check if directory for results already exists otherwise create
if not os.path.exists(dircrloc):
    os.makedirs(dircrloc)

#Making list of all files in the directory
Filelist= os.listdir(Fileslocation)
#%% FUNCTIONS
#Function for finding nearest value in array
def find_nearest(array, value):
    array = np.asarray(array)
    idx = (np.abs(array - value)).argmin()
    return array[idx]

#Alternative function for finding the nearest value
def nearest(items, pivot):
    return min(items, key=lambda x: abs(x - pivot))

#functions for shifting
def shift_heatcurve(heatcurve,step, halfloc):
    if step <0:
        Heatcurvesm_up_shift = heatcurve[0+3:halfloc+step] #defining the up part of the cable
        Heatcurvesm_down_shift = np.flip(heatcurve[halfloc+step:len(heatcurve)-3+step*2]) #
Defining the down part of the cable
        MSE1 = ((Heatcurvesm_up_shift - Heatcurvesm_down_shift)**2).mean(axis=0) #
calculating the MSE between the up and down
    else:
        Heatcurvesm_up_shift = heatcurve[0+3+step*2:halfloc+step] #defining the up part of the
cable
        Heatcurvesm_down_shift = np.flip(heatcurve[halfloc+step:len(heatcurve)-3]) # Defining
the down part of the cable
```

```
    MSE1 = np.round(((Heatcurvesm_up_shift - Heatcurvesm_down_shift)**2).mean(axis=0),2) # calculating

Heatcurve_average=np.mean(np.array([Heatcurvesm_up_shift,Heatcurvesm_down_shift]),axis=0)
  return MSE1, Heatcurvesm_up_shift, Heatcurvesm_down_shift, Heatcurve_average


#%% Active heating
active_heating={}
for i, file in enumerate(Filelist):
  Filenameloc= os.path.join(Fileslocation,file) #Creating file loc and name combination for opening
  pickle_data=open(Filenameloc,"rb") # open the file with pickle

  dataset = pickle.load(pickle_data) #defining the file as data

  #select heatruns per channel
  for ii, location in enumerate(chtoloc.loc[:, "LOCATION"]):
    chloc = chtoloc.loc[chtoloc['LOCATION'] == location].squeeze() # finding the channel name belonging to the location name (e.g. W3 = Ch3)
    channel = chtoloc.loc[chtoloc['LOCATION'] == location]["CHANNEL"].squeeze()
    if type(channel)!=str:
      channel='geen CH' #er zijn locaties zonder channel, die geven nu errors dus moet dit regeltje erbij

    DateTime= dataset[f'datetimes_{location}']# getting the datetime from the dataset
    DateTime_min = DateTime.min()# getting the earliest moment in the dataset
    DateTime_max = DateTime.max()# getting the end datetime of the dataset

    # Loop through all heating experiments/RUNs for the corresponding channel (as listed in the metafile)
    subset_heatruns = Meta_heatwell[Meta_heatwell["channel"] == channel]
    for l, runs in enumerate(subset_heatruns.loc[:,"Begindatum"]):
      heatbegin = subset_heatruns.iloc[l]["Begindatum"] # start datetime of run
      heatend = subset_heatruns.iloc[l]["Einddatum"] #end datetime of run

      # determine if the run is in the dataset (is in the same timedomain)
      if heatbegin > DateTime_min and heatend < DateTime_max:
        begin_run = nearest(DateTime,heatbegin) # getting the start of the run in the dataset datetime
        begin_run_ind=DateTime.get_loc(f'{begin_run}')-5 # getting the index belonging to the start datetime of the run
        end_run = nearest(DateTime,heatend) #getting the end of the run in the dataset datetime
        end_run_ind = DateTime.get_loc(f'{end_run}') # getting the index belonging to the end datetime of the run
        run= subset_heatruns.iloc[l]["RUN"] # getting the run id from the metafile

        Heatcurve = dataset[f'calTemp_{location}'][:,begin_run_ind:end_run_ind] #g getting the calTemp from the dataset for the timedomain of the run and the specific location
        Heatcurvesm=copy(Heatcurve) # creating a copy of the heatcurve to not smooth the heatcurve parameter
```

```
        # Coolcurve
        end_cooling = nearest(DateTime,end_run + dt_cooling)
        end_cooling_ind = DateTime.get_loc(f'{end_cooling}')
        Coolcurve = dataset[f'calTemp_{location}'][:,end_run_ind:end_cooling_ind] #g getting the calTemp from the dataset for the timedomain of the run and the specific location
        Coolcurvesm = copy(Coolcurve)

        #Loop through calTemp timesteps - Script finding the best shift of the up and down segments based on MSE. shift of 1 index up and down is considered
        Heatcurve_def=[] #empty list to store data
        Coolcurve_def = []

        for d in range(Heatcurvesm.shape[1]):
            Heatcurvesm_single = np.convolve(Heatcurvesm[:,d], np.ones(5)/5, mode='same') # smoothing the caltemp over the length of the cable for every timestep
            if len(Heatcurvesm_single) % 2 == 1: # determining if length is odd #deleting 1 datavalue at the end to make it even
                Heatcurvesm_single=Heatcurvesm_single[0:len(Heatcurvesm_single)-1] #deleting 1 datavalue at the end to make it even

            halfloc = int(len(Heatcurvesm_single)/2) # finding the mid index of the parameter

            # Define up and down part of the cable without a shift
            Heatcurvesm_up_org = Heatcurvesm_single[0+3:halfloc] #defining the up part of the cable
            Heatcurvesm_down_org = np.flip(Heatcurvesm_single[halfloc:len(Heatcurvesm_single)-3]) # Defining the down part of the cable

            # Extract shift for the corresponding channel and location
            shift = chtoloc.loc[chtoloc.index[chtoloc['CHANNEL'] == channel][0],'SHIFT'] #get shift from metadata
            #Determing MSE, up and down part of the cable with the predefined shift
            MSE1, Heatcurvesm_up_shift, Heatcurvesm_down_shift, Heatcurve_average=shift_heatcurve(Heatcurvesm_single, shift, halfloc)

            Heatcurve_def.append(Heatcurve_average)

            if Print_ON == True: # print the 3 heatcurves up and down on t=10
              if d==10:
                # Determining MSE for basic up down combination
                MSE_org = np.round(((Heatcurvesm_up_org - Heatcurvesm_down_org)**2).mean(axis=0),2) # calculating the MSE between the up and down
                fig, ax = plt.subplots(figsize=([12,6]))
                ax.plot(Heatcurvesm_up_org,'b',linewidth=1,label=f'No shift UP {MSE_org}')
                ax.plot(Heatcurvesm_down_org,'b--',linewidth=1,label='No shift DOWN')
                ax.plot(Heatcurvesm_up_shift,'g',linewidth=1,label=f'Shift UP {MSE1}')
                ax.plot(Heatcurvesm_down_shift,'g--',linewidth=1,label='Shift DOWN')
                ax.plot(Heatcurve_def[-1],'k',linewidth=1,label='Heatcurve def')
                ax.legend()
                ax.set_title(' run='+str(run)+' loc=' +str(channel))
                ax.set_xlabel('z')
                ax.set_ylabel('T')
```

```
                plt.savefig(script_path+'\png/'+str(run)+str(channel)+'.png')

            # add lists with MSE and heatcurve to empty output dictionary
        Heatcurve_def = np.vstack(Heatcurve_def)
        active_heating['Heatcurve_'+str(channel)+'_'+str(run)]=Heatcurve_def
        active_heating['DateTimes_'+str(channel)+'_'+str(run)]=
DateTime[begin_run_ind:end_run_ind]


        for d in range(Coolcurvesm.shape[1]):
            Coolcurvesm_single = np.convolve(Coolcurvesm[:,d], np.ones(5)/5,
mode='same') # smoothing the caltemp over the length of the cable for every timestep
            if len(Coolcurvesm_single) % 2 == 1: # determining if length is odd #deleting 1
datavalue at the end to make it even
                Coolcurvesm_single = Coolcurvesm_single[0:len(Coolcurvesm_single)-1]
#deleting 1 datavalue at the end to make it even

            halfloc = int(len(Coolcurvesm_single)/2) # finding the mid index of the parameter

            # Define up and down part of the cable without a shift
            Coolcurvesm_up_org = Coolcurvesm_single[0+3:halfloc] #defining the up part of
the cable
            Coolcurvesm_down_org =
np.flip(Coolcurvesm_single[halfloc:len(Coolcurvesm_single)-3]) # Defining the down part of the
cable

            # Extract shift for the corresponding channel and location
            shift = chtoloc.loc[chtoloc.index[chtoloc['CHANNEL'] == channel][0],'SHIFT'] #get
shift from metadata
            #Determing MSE, up and down part of the cable with the predefined shift
            MSE1, Heatcurvesm_up_shift, Coolcurvesm_down_shift, Coolcurve_average=
shift_heatcurve(Coolcurvesm_single, shift, halfloc)

            Coolcurve_def.append(Coolcurve_average)

        Coolcurve_def = np.vstack(Coolcurve_def)
        active_heating['Coolcurve_'+str(channel)+'_'+str(run)]=Coolcurve_def
        active_heating['DateTimes_Cool_'+str(channel)+'_'+str(run)]=
DateTime[end_run_ind:end_cooling_ind]

    # xyz_variable later inbouwen als de rest klaar is

#%% PLOT active heating results of the heating and cooling curve
act_heat_key = list(active_heating.keys())
channel_run = list(set(["Ch" + i.split('Ch', 1)[-1] for i in act_heat_key]))

for combination in channel_run:
    fig, axs = plt.subplots(1,2)
    axs[0].plot(active_heating["DateTimes_" + combination], active_heating["Heatcurve_" +
combination])
    axs[0].set_title("Heatcurve_" + combination)

    axs[1].plot(active_heating["DateTimes_Cool_" + combination],
        active_heating["Coolcurve_" + combination])
```

```
    axs[1].set_title("Coolcurve_" + combination)
    plt.savefig(dircrloc+ combination +'.png')

#%% adjust all data; passive heating
passive_heating={}
for i, file in enumerate(Filelist):
    Filenameloc= os.path.join(Fileslocation,file) #Creating file loc and name combination for opening
    pickle_data=open(Filenameloc,"rb") # open the file with pickle
    dataset = pickle.load(pickle_data) #defining the file as data

    for location in chtoloc.loc[:, "LOCATION"]:
        chloc = chtoloc.loc[chtoloc['LOCATION'] == location].squeeze() # finding the channel name belonging to the location name (e.g. W3 = Ch3)
        channel = chtoloc.loc[chtoloc['LOCATION'] == location]["CHANNEL"].squeeze()
        if type(channel)!=str:
            channel='geen CH' #er zijn locaties zonder channel, die geven nu errors dus moet dit regeltje erbij

        #select calTemp of the corresponding channel
        calTemp_df = dataset["calTemp_" + location]

        # SHIFT
        calTemp_def = []
        for time_ind in range(calTemp_df.shape[1]):
            calTemp_single = np.convolve(calTemp_df[:,time_ind], np.ones(5)/5, mode='same')
            if len(calTemp_single) % 2 == 1: # determining if length is odd #deleting 1 datavalue at the end to make it even
                calTemp_single = calTemp_single[0:len(calTemp_single)-1] #deleting 1 datavalue at the end to make it even

            halfloc = int(len(calTemp_single)/2) # finding the mid index of the parameter
            # Determining MSE for basic up down combination
            calTemp_up_org = calTemp_single[0+3:halfloc] #defining the up part of the cable
            calTemp_down_org = np.flip(calTemp_single[halfloc:len(calTemp_single)-3]) # Defining the down part of the cable
            MSE_org = np.round(((calTemp_up_org - calTemp_down_org)**2).mean(axis=0),2) # calculating the MSE between the up and down

            step= chtoloc.loc[chtoloc.index[chtoloc['CHANNEL'] == channel][0],'SHIFT'] #get shift from metadata
            #Determing MSE for shift
            MSE1, calTemp_up_shift, calTemp_down_shift, calTemp_average=shift_heatcurve(calTemp_single,step,halfloc)
            calTemp_def.append(calTemp_average)

        calTemp_def = np.vstack(calTemp_def)
        passive_heating['calTemp_'+str(channel)]=calTemp_def
        passive_heating['DateTimes_'+str(channel)] = dataset["datetimes_" + location]
#%%
pas_heat_key = list(passive_heating.keys())
channel_run = list(set([i.split('_', 1)[-1] for i in pas_heat_key]))

for combination in channel_run:
```

```
    fig, axs = plt.subplots()
    axs.plot(passive_heating["DateTimes_" + combination],
             passive_heating["calTemp_" + combination])
    axs.set_title("calTemp " + combination)
    plt.savefig(dircrloc+combination+'.png')
#%%Saving results
Exportname = 'Active_heatdata_DTS'
ExportNameLoc = os.path.join(dircrloc, Exportname)
pickle_out=open(ExportNameLoc,"wb")
pickle.dump(active_heating,pickle_out)
pickle_out.close()

Exportname = 'Passive_heatdata_DTS'
ExportNameLoc = os.path.join(dircrloc, Exportname)
pickle_out=open(ExportNameLoc,"wb")
pickle.dump(active_heating,pickle_out)
pickle_out.close()
```

B.5 Script 5 Bereken ΔT en stroomsnelheid

```
# -*- coding: utf-8 -*-
"""

_____Description_____
_____

_____
_____

Developed by W. Bakx
date 24/06/2022
_____
_____


_____INPUT_____
_____
The following input is needed for the script to run:
- Location of the result files of script 5
- Output location

_____
_____

_____OUTPUT_____
_____
The following output is generated:
- File containing all the combined data

_____
_____

@author: bakxw
"""
#Loading modules and packages
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import datetime as dt
import os.path
import os
import pickle
from scipy.optimize import curve_fit

#%% ----------Specify location of input and output

Inputlocation_S5 =
r"C:\PY_PROJECTS\GroFloMo\klooster_2021\data\20210507\S4_totalheat" # Results
location previous script 4

Outputlocation =  r"C:\PY_PROJECTS\GroFloMo\klooster_2021\data\20210507\S5_Velocity"
# location for the results of this script
```

```
#Location of the metadata XLSX file and sheet containing Velocity parameters
meta_parameters =
pd.read_excel(r"C:\PY_PROJECTS\GroFloMo\klooster_2021\Meta_input\Klooster_2021_met
adata.xlsx",sheet_name='Vel_parameters')


#%% Bring to xlsx meta file
#==================  Input to the velocity calculation
#Heating cable
Q_new  =meta_parameters.loc[meta_parameters['Parameter'] =='Q_new', 'Value'].iloc[0]
#34.13              # Heat input                                  [W/m]
rh2_new = meta_parameters.loc[meta_parameters['Parameter'] =='rh2_new', 'Value'].iloc[0]
# 0.00145           # Outer radius of heating cable
rh1_new = meta_parameters.loc[meta_parameters['Parameter'] =='rh1_new', 'Value'].iloc[0]
#0.000175          # Radius of heating part heating cable
Kh_new = meta_parameters.loc[meta_parameters['Parameter'] =='Kh_new', 'Value'].iloc[0]
#0.2               # Thermal conductivity of heating cable material (silicone rubber)

#Heatin wire
rhc2_new = meta_parameters.loc[meta_parameters['Parameter'] =='rhc2_new',
'Value'].iloc[0]  #0.000175            # Radius of outer layer cable                  [m]
Kc_new = meta_parameters.loc[meta_parameters['Parameter'] =='Kc_new', 'Value'].iloc[0]
#390               # Thermal conductivity of cable protection material [W/m/K]

#Fibre optic cable
#Jacket
rj1_new = meta_parameters.loc[meta_parameters['Parameter'] =='rj1_new', 'Value'].iloc[0]
#0.000944125           # Radius of inner steel core of cable              [m]
rj2_new = meta_parameters.loc[meta_parameters['Parameter'] =='rj2_new', 'Value'].iloc[0]
#0.00135            # Radius of outer layer cable                  [m]
Kj_new = meta_parameters.loc[meta_parameters['Parameter'] =='Kj_new', 'Value'].iloc[0]
#0.196              # Thermal conductivity of cable protection material [W/m/K]

#aramid
ra1_new = meta_parameters.loc[meta_parameters['Parameter'] =='ra1_new', 'Value'].iloc[0]
#0.0004            # Radius of inner steel core of cable              [m]
ra2_new = meta_parameters.loc[meta_parameters['Parameter'] =='ra2_new', 'Value'].iloc[0]
#0.000944125           # Radius of outer layer cable                  [m]
Ka_new = meta_parameters.loc[meta_parameters['Parameter'] =='Ka_new', 'Value'].iloc[0]
#0.04              # Thermal conductivity of cable protection material [W/m/K]

#mantle
rm1_new = meta_parameters.loc[meta_parameters['Parameter'] =='rm1_new', 'Value'].iloc[0]
#0.00005           # Radius of inner steel core of cable              [m]
rm2_new = meta_parameters.loc[meta_parameters['Parameter'] =='rm2_new', 'Value'].iloc[0]
#0.0004            # Radius of outer layer cable                  [m]
Km_new = meta_parameters.loc[meta_parameters['Parameter'] =='Km_new', 'Value'].iloc[0]
#0.196             # Thermal conductivity of cable protection material [W/m/K]

#fibre
rf2_new = meta_parameters.loc[meta_parameters['Parameter'] =='rf2_new', 'Value'].iloc[0]
#0.00005           # Radius of outer layer cable                  [m]
```

```
Kf_new = meta_parameters.loc[meta_parameters['Parameter'] =='Kf_new', 'Value'].iloc[0]  #2
# Thermal conductivity of cable protection material [W/m/K]

#Sediment
n_new  = meta_parameters.loc[meta_parameters['Parameter'] =='n_new', 'Value'].iloc[0]
#0.41              # Porosity                                  [-]
Ks_new = meta_parameters.loc[meta_parameters['Parameter'] =='Ks_new', 'Value'].iloc[0]
#5                 # Thermal conductivity of solid particles          [W/m/K]


mu_new = meta_parameters.loc[meta_parameters['Parameter'] =='mu_new', 'Value'].iloc[0]
#0.001             # Dynamic viscosity                         [N*s/m^2]

# Water
Cp_new = meta_parameters.loc[meta_parameters['Parameter'] =='Cp_new', 'Value'].iloc[0]
#4186              # Specific heat capacity of water                 [J/kg/K]
v_new  = meta_parameters.loc[meta_parameters['Parameter'] =='v_new', 'Value'].iloc[0]
#0.000001          # Kinematic viscosity of water                    [m^2/s]
Kw_new = meta_parameters.loc[meta_parameters['Parameter'] =='Kw_new', 'Value'].iloc[0]
#0.591             # Thermal conductivity of water                   [W/m/K]

#Fit parameters
C_new = meta_parameters.loc[meta_parameters['Parameter'] =='Kw_new', 'Value'].iloc[0]
#0.00259355#0.47090328              #0.4113;     % Coefficient to determine Nusselt number
(fitted)
m = meta_parameters.loc[meta_parameters['Parameter'] =='Kw_new', 'Value'].iloc[0]
#0.47936169#0.22884307              #0.1488;     % Coefficient to determine Nusselt number
(fitted)
x_new = meta_parameters.loc[meta_parameters['Parameter'] =='Kw_new', 'Value'].iloc[0]
#0.000001          # Distance through the sediment (near zero because of the direct contact)



#%%Collecting the data from script 4


#Getting the DTS date for every date locationspecific
ListOfFiles_S5=os.listdir(Inputlocation_S5)
for i in range(len(ListOfFiles_S5)):
    Picklename = ListOfFiles_S5[i]
    Picklepathname = os.path.join(Inputlocation_S5,Picklename)
    Pickleloc=open(f"{Picklepathname}","rb")
    vars()[f'{Picklename}']=pickle.load(Pickleloc)
del Pickleloc, Picklename, Picklepathname, i


#Check if output dir already exists otherwise create
if not os.path.exists(Outputlocation):
    os.makedirs(Outputlocation)
```

```
#---------------------------Defining functions for curve fitting to the heatdata and cooldata
# Function of the heat curve fit
def heatfit(x, a, b, c, d):
    return a/((b*x)+c)+d
#Function of the cool curve fit
def coolfit(x, e, f, g):
    return e*np.exp(-f*x)+g


# Calculated constant parameters for the velocity calculation
L_new  = 2*rh2_new              # 0.15*10^-6;% Characteristic length (cable diameter)
[m]
Keff_new = n_new*Kw_new + (1-n_new)*Ks_new # Effective thermal conductivity of water
and solids combined   [W/m/K]
Pr_new = mu_new*Cp_new/Kw_new;          # Prandtl number
P_constant = (L_new*v_new**m)/(rh2_new*Keff_new*C_new*Pr_new**(1/3)*L_new**m)

Tot_dis_new = x_new+rj2_new+rh2_new
Frac_hc_new = rhc2_new/Tot_dis_new
Frac_h_new = (rh2_new-rh1_new)/Tot_dis_new
Frac_j_new = (rj2_new-rj1_new)/Tot_dis_new
Frac_a_new = (ra2_new-ra1_new)/Tot_dis_new
Frac_m_new =(rm2_new-rm1_new)/Tot_dis_new
Frac_f_new = (rf2_new)/Tot_dis_new
Frac_sed_new = x_new/Tot_dis_new
K_tot_new =
(1/(Frac_hc_new/Kc_new))+(1/(Frac_h_new/Kh_new))+(1/(Frac_j_new/Kj_new))+(1/(Frac_a_
new/Ka_new))+(1/(Frac_m_new/Km_new))+(1/(Frac_f_new/Kf_new))+(1/(Frac_sed_new/Keff
_new))
b=(1/K_tot_new)*(np.log((x_new+rh2_new+(rj2_new-rf2_new))/rh1_new))




#%% ------------------------------Calculating DeltaT 3 ways
keyloc=list(Active_heatdata_DTS.keys())

#-------Determing lowest temperature value (3 methods)
Low_cool={}
for i in range(len(keyloc)):
    keysplit=keyloc[i].split('_')
    if len(keysplit) == 4 and keysplit[0] =='DateTimes' and keysplit[1] == 'Cool':
        Namepie = keyloc[i]
        Namerun = keysplit[3]
        Nameloc = keysplit[2]
        Xval_cool = Active_heatdata_DTS[f'{Namepie}']
        Xval_cool2 = (Xval_cool - np.datetime64('1970-01-01T00:00:00Z')) / np.timedelta64(1,
's')
        Xval_cool2=np.zeros(shape=(len(Xval_cool2)))
        Yval_cool = Active_heatdata_DTS[f'Coolcurve_{Nameloc}_{Namerun}']
        Low_cool[f'Lowcool_{Nameloc}_{Namerun}_asym'] = np.zeros(len(Yval_cool))
```

```
        Low_cool[f'Lowcool_{Nameloc}_{Namerun}_2proc'] = np.zeros(len(Yval_cool))
        Low_cool[f'Lowcool_{Nameloc}_{Namerun}_5val'] = np.zeros(len(Yval_cool))
        for k in range(len(Yval_cool)):
            Yval_cool2=Yval_cool[:,k]
            try:
                Hcurve, rddd=curve_fit(coolfit,Xval_cool2,Yval_cool2,bounds=((-np.inf,-
np.inf,0),(np.inf, np.inf, 50)),maxfev=1000)
            except RuntimeError:
                print("Error - curve_fit failed", k)
                Hcurve[2] = 0
            Low_cool[f'Lowcool_{Nameloc}_{Namerun}_asym'][k] = Hcurve[2]
            sort=sorted(Yval_cool2)
            Low_cool[f'Lowcool_{Nameloc}_{Namerun}_5val'][k] = np.mean(sort[:5])
            Low_cool[f'Lowcool_{Nameloc}_{Namerun}_2proc'][k] = np.percentile(Yval_cool2,2)



#------------Determing highest temperature value (3 methods)
Piek_heat={}
for i in range(len(keyloc)):
    keysplit=keyloc[i].split('_')
    if len(keysplit) ==3 and keysplit[0] == 'DateTimes':
        Namepie = keyloc[i]
        Namerun = keysplit[2]
        Nameloc = keysplit[1]
        Xval_heat = Active_heatdata_DTS[f'{Namepie}']
        Xval_heat2 = (Xval_heat - np.datetime64('1970-01-01T00:00:00Z')) / np.timedelta64(1,
's')
        Xval_heat2=np.zeros(shape=(len(Xval_heat2)))
        Yval_heat = Active_heatdata_DTS[f'Heatcurve_{Nameloc}_{Namerun}']
        Piek_heat[f'Heatpiek_{Nameloc}_{Namerun}_asym'] = np.zeros(len(Yval_heat))
        Piek_heat[f'Heatpiek_{Nameloc}_{Namerun}_5val'] = np.zeros(len(Yval_heat))
        Piek_heat[f'Heatpiek_{Nameloc}_{Namerun}_2proc'] = np.zeros(len(Yval_heat))
        for k in range(len(Yval_heat)):
            Yval_heat2=Yval_heat[:,k]
            try:
                Hcurve, rddd=curve_fit(heatfit,Xval_heat2,Yval_heat2,bounds=((-np.inf,-np.inf,-
np.inf,0),(np.inf, np.inf, np.inf, 50)),maxfev=1000)
            except RuntimeError:
                print("Error - curve_fit failed", k)
                Hcurve[3] = 0
            Piek_heat[f'Heatpiek_{Nameloc}_{Namerun}_asym'][k] = Hcurve[3]
            sort=sorted(Yval_heat2,reverse=True)
            Piek_heat[f'Heatpiek_{Nameloc}_{Namerun}_5val'][k] = np.mean(sort[:5])
            Piek_heat[f'Heatpiek_{Nameloc}_{Namerun}_2proc'][k] =
np.percentile(Yval_heat2,98)


#
==========================================================================
=======
#           #Check figure
#           plt.figure()
#           #plt.title('Channel_'+str(pickchannel))
```

```
#           plt.plot(AntiStokes[:,150],'r',linewidth=0.5,label='AntiStokes')
#           plt.plot(Stokes[:,150],'g',linewidth=0.5,label='Stokes')
#           plt.plot(tempC[:,150],'b',linewidth=0.5,label='Calibration by DTS')
#           plt.axvline(x = Chdist, color = 'b')
#           #plt.plot(Xcordstref1,Ycordstref1,'g--')
#           #plt.legend(('calTemp', 'tempC', 'tref_2','tref_1'),
#           #          loc='lower right')
#           plt.show()
#
=========================================================================
=======

   # ----------------------Calculating deltaT for well off
DeltaT={}
keylist_cool=list(Low_cool.keys())
keylist_heat=list(Piek_heat.keys())
for i in range(len(keylist_cool)):
   keysplit_cool=keylist_cool[i].split('_')
   Namerun_cool=keysplit_cool[1]
   Nameloc_cool=keysplit_cool[2]
   Namemeth_cool=keysplit_cool[3]
   Coolneeded=keylist_cool[i]
   cool_in=Low_cool[f'{Coolneeded}']
   #Loopje door de heat data om dezelfde loc en run op te halen
   for j in range(len(keylist_heat)):
      keysplit_heat=keylist_heat[j].split('_')
      Namerun_heat=keysplit_heat[1]
      Nameloc_heat=keysplit_heat[2]
      Namemeth_heat=keysplit_heat[3]
      if Namerun_cool == Namerun_heat and Nameloc_cool == Nameloc_heat and
Namemeth_cool == Namemeth_heat:
         Heatneeded=keylist_heat[j]
         heat_in=Piek_heat[f'{Heatneeded}']
   if len(heat_in) < len(cool_in):
      inkort=len(heat_in)
      cool_in=cool_in[:inkort]
   elif len(heat_in) > len(cool_in):
      inkort=len(cool_in)
      heat_in=heat_in[:inkort]
   DeltaT[f'DeltaT_{Nameloc_cool}_{Namerun_cool}_{Namemeth_cool}']=heat_in - cool_in


#%%----------Calculating velocity


Velocities={}
keylist_deltaT=list(DeltaT.keys())
for i in range(len(keylist_deltaT)):
   keysplit_deltaT=keylist_deltaT[i].split('_')
   Namerun_deltaT=keysplit_deltaT[1]
   Nameloc_deltaT=keysplit_deltaT[2]
   Namemeth_deltaT=keysplit_deltaT[3]
   if Nameloc_deltaT == 'Ch9' or Nameloc_deltaT=='Ch2'or Nameloc_deltaT=='Ch1'or
Nameloc_deltaT=='Ch3':
```

```
        Q  =30.31075
    elif Nameloc_deltaT=='Ch11':
        Q  =30.82667
    elif Nameloc_deltaT== 'Ch14':
        Q  =30.78967
    elif Nameloc_deltaT== 'Ch16':
        Q  =30.97487
    elif Nameloc_deltaT== 'Ch5':
        Q  =25.36102
    elif Nameloc_deltaT== 'Ch7' or Nameloc_deltaT=='Ch8' or Nameloc_deltaT=='Ch12':
        Q  =25.62667
    elif Nameloc_deltaT== 'Ch10':
        Q  =31.08626
    elif Nameloc_deltaT== 'Ch6':
        Q  =25.39102
    else:
        print('Q not correctly determined yet for location  '+Nameloc_deltaT)
        Q =30.3 #Estimate voor C17
    incalc1=keylist_deltaT[i]
    incalc2=DeltaT[f'{incalc1}']

Velocities[f'{Namerun_deltaT}_{Nameloc_deltaT}_vel_{Namemeth_deltaT}']=np.zeros(len(inc
alc2))
    for j in range(len(incalc2)):
        Velocities[f'{Namerun_deltaT}_{Nameloc_deltaT}_vel_{Namemeth_deltaT}'][j]=
(P_constant/(((incalc2[j]*2*np.pi)/Q)+b))**(1/m)


#%% ----------Export results
#
======================================================================
# Exporting the raw Asymps and DeltaT as one big dictonary incl xyz for well on
Total_Well={**DeltaT, **Velocities}
Total_Well={**Total_Well, **Low_cool}
Total_Well={**Total_Well, **Piek_heat}

#Saving part
Name_welloff_export1='Total_Well'
Exportnamepath_welloff_export1 = os.path.join(Outputlocation, Name_welloff_export1)
Pickle_welloff_export1 = open(Exportnamepath_welloff_export1, "wb")
pickle.dump(Total_Well, Pickle_welloff_export1)
Pickle_welloff_export1.close()
```

C Beschrijving parameters van de metadata file

DTS_channel_loc
Het tabblad DTS_channel_loc is als volgt opgebouwd:

- Name: Naam van de meetlocatie
- CH: Kanaal waarop de meetlocatie voorkomt
- MinDis: Minimale afstand van het kabelsegment dat bij de locatie hoort. Geef hier in wat je in de DTS afleest en script 3 wil de exacte locatie bepalen.
- MidDis Midden afstand van het kabelsegment dat bij de locatie hoort. Geef hier in wat je in de DTS afleest en script 3 wil de exacte locatie bepalen.
- MaxDis Maximale afstand van het kabelsegment dat bij de locatie hoort. Geef hier in wat je in de DTS afleest en script 3 wil de exacte locatie bepalen.
- MV: Maaiveldhoogte van de locatie
- Xloc: afstand van de meetlocatie ten opzichte van de winput in de X richting
- Yloc: afstand van de meetlocatie ten opzichte van de winput in de Y richting
- Xcor: Y coordinaat van de meetlocatie
- Ycor: X coordinaat van de meetlocatie

De overige kolommen in de metafile zijn niet meer relevant.

DTS_cali_loc
Beschrijving van de kabelsegmenten door de kalibratiebaden, met:

- Channel: Het kanaal van de DTS unit waarop de kabel is aangesloten
- Segment: Een nummering van de segmenten die op het kanaal aanwezig zijn.
- Temperature: Geeft aan welke temperatuur het kalibratiebad heeft. Passive betekent dat het kalibratiebad waarin water op een zo constant mogelijke temperatuur werd gehouden (maar niet actief werd verwarmd of gekoeld). Active betekent dat het kalibratiebad actief op een constante hogere temperatuur werd gehouden.
- Start: Begin afstand op de kabel
- End: Eind afstand op de kabel
-

Heatexper_wellact
De volgende kolommen zijn in het tabblad opgenomen:

- Date: datum notatie maar dan omgedraaid
- Begindatum: Begindatum en tijd van het opwarmexperiment
- Einddatum: Einddatum en tijd van het opwarmexperiment
- HEKP 01-14 t/m HEKP13-18: Kolomnamen zijn de aanwezige winputten. Met 0,1 of 2 is de status van de pomp aangegeven. Bij 0 is de winput uit. Bij 1 is de winput aan het onttrekken. Bij 2 is de winput aan het infiltreren.
- Channel: Locatie die in het experiment wordt opgewarmd
- Duration: Duur van het opwarmexperiment
- Endtime: eindtijd van het opwarmexperiment
- Starttime: starttijd van het opwarmexperiment
- RUN: Nummer van de opwarmrun

De tabbladen Useless_heating en Useless_injection zijn op dezelfde wijze opgebouwd.

Well_activity (Optioneel)
Tabblad die per uur aangeeft wat de winputten in het winveld doen. De volgende kolommen zijn opgenomen:

- Datetime: kolom met de datum en tijd. Loopt per uur op

- Well 08-11 t/m Well 12-19: Kolomnamen zijn de aanwezige winputten. Met 0,1 of 2 is de status van de pomp aangegeven. Bij 0 is de winput uit. Bij 1 is de winput aan het onttrekken. Bij 2 is de winput aan het infiltreren

Vel_parameters

Tabblad waarin de eigenschappen en constantes zijn opgenomen die van belang zijn voor de stroomsnelheid berekening. De volgende waarden zijn opgenomen:

- Q_new = 34,13 (W/m) Heating cable
- rh2_new = 0,00145 (m) Heating cable
- rh1_new = 0,000175 (m) Heating cable
- Kh_new = 0,2(W/m/K) Heating cable

- rhc2_new = 0,000175 (m) Heating wire
- Kc_new = 390 (W/m/K) Heating wire

- rj1_new = 0,000944125 (m) Jacket FOC
- rj2_new = 0,00135 (m) Jacket FOC
- Kj_new = 0,196 (W/m/K) Jacket FOC

- ra1_new = 0,0004 (m) aramid FOC
- ra2_new = 0,000944125 (m) aramid FOC
- Ka_new = 0,04 (W/m/K) aramid FOC

- rm1_new = 0,00005 (m) Mantle FOC
- rm2_new = 0,0004 (m) Mantle FOC
- Km_new = 0,196 (W/m/K) Mantle FOC

- rf2_new = 0,00005 (m) fibre FOC
- Kf_new = 2 (W/m/K) fibre FOC

- n_new = 0,41 Sediment
- Ks_new = 5 (W/m/K) Sediment
- mu_new = 0,001 Sediment
- x_new = 0,000001 (m) Sediment

- Cp_new = 4186 (J/kg/K) Water
- v_new = 0,000001 (m^2/s) Water
- Kw_new = 0,591 (W/m/K) Water
-
- C_new = 0,00259355 FIT parameter
- M = 0,47936169 FIT parameter