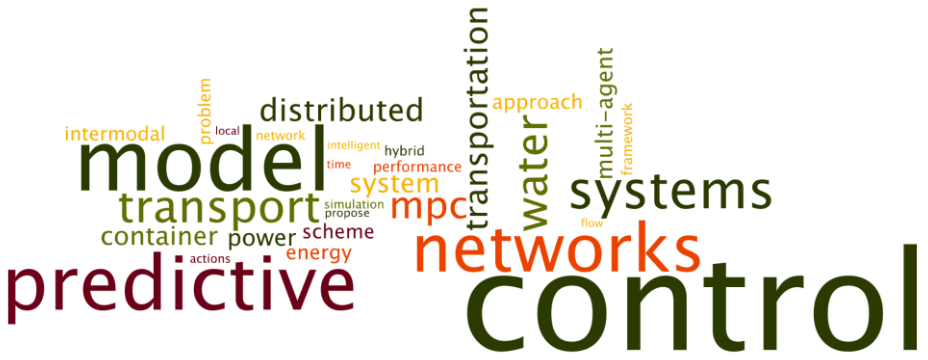
Nelen & Schuurmans



Onderzoek optimalisatie regelingen ControlNEXT middels MPC

MPC versus sturingsregels

Hoogheemraadschap Hollands Noorderkwartier

2-1-2019

Onderzoek optimalisatie regelingen ControlNEXT middels MPC

MPC versus stuurregels

Voor
Hoogheemraadschap Hollands Noorderkwartier
Postbus 250, 1700 AG
Heerhugowaard

Nelen & Schuurmans
Postbus 1219
3500 BE Utrecht

www.nelen-schuurmans.nl

Projectgegevens

Dossier : R0097
Datum : 2-1-2019

Inhoudsopgave

1 Inleiding

1.1 Aanleiding

Hoogheemraadschap Hollands Noorderkwartier voert al ruim 10 jaar het waterbeheer van haar boezems uit met het real-time controlsysteem ControlNEXT. Dit systeem bevat sturingsregels die op basis van verwachtingen en de huidige situatie een advies geeft hoe het waterbeheer te reguleren. Deze sturingsregels behoren tot de categorie feed-forward sturing. Op basis van vooraf gedefinieerde regels en verwachtingen wordt geanticipeerd op hetgeen wat komen gaat. Afgelopen jaren is een andere type sturing in opkomst. Met de komst van snellere computers en betere technieken zijn optimalisatietechnieken geïntroduceerd in het waterbeheer. HHNK wil onderzoeken of deze wijze van sturen voordelen biedt t.o.v. de huidige sturingsregels.

1.2 Doel

Het project is een onderzoeksproject waarbij gekeken wordt of het gebruik van MPC in ControlNEXT regelingen mogelijk is en tot verbeteringen leidt. Dit in vergelijking tot de huidige regelingen die zijn opgebouwd uit rekenregels. Het eindproduct van dit onderzoek is geen operationele sturing, maar een sturing die gedraaid heeft over een periode uit het verleden en gevoed is met gemeten neerslag i.p.v. verwachtingen.

In het eindresultaat is het mogelijk om de effecten te analyseren. Het project bevat een kwantitatieve beoordelingen op o.a. energiekosten en een kwalitatieve boordeling op o.a. het waterstandsgedrag. In het project wordt daarnaast het gebruiksgemak en het functionele beheer van de RTC-Tools software onderzocht.

1.3 Werkwijze

In het overleg op 15 maart 2018 hebben we de volgende mogelijkheden voor onderzoek besproken:

A. Vergelijking RTC als één bak met huidige regelingen
B. Vergelijking boezem als één RTC-bak versus een hydraulisch RTC model
C. Onderzoek mogelijkheden (geen echte vergelijking) van alles omvattend RTC model versus normale regelingen

We hebben gezamenlijk gekozen om optie A als onderzoeksrichting te kiezen. Hiermee wordt namelijk puur gekeken naar het effect van optimalisatie versus een beslisboomregeling, terwijl de andere onderzoeksrichtingen breder zijn.

HHNK is geïnteresseerd in het effect van neerslagonzekerheid op sturing. Dit wordt echter niet meegenomen in het project. Het advies is om dit te onderzoeken in een aparte studie met behulp van neerslag-afvoermodellen en te kijken wat het effect is op de variatie in de waterstand.

1.4 Opleverproducten

Aan het einde van het project worden de volgende producten opgeleverd:

- Stand-alone ControlNEXT systeem
- Gekoppeld RTC-Tools 2 model
- Presentatie aan peilbeheer
- Rapport

1.5 Leeswijzer

Dit document bevat de rapportage van het onderzoek optimalisatie regelingen ControlNEXT middels MPC. In hoofdstuk 2 wordt beschreven welke projectkeuzes zijn gemaakt, wat de uitgangspunten van het onderzoek waren, de doelkeuzes voor het model en op basis van welke punten de resultaten worden bekeken. Hoofdstuk 3 beschrijft de resultaten van het onderzoek en de vergelijking tussen de MPC en de huidige regeling. Hoofdstuk 4 concludeert en doet aanbevelingen op basis van de resultaten van het onderzoek.

2 Plan van Aanpak

Voor het onderzoek naar de optimalisatie van de regelingen middels MPC zijn er vooraf in overleg met HHNK keuzes gemaakt die de richting bepalen van het project.

2.1 Rolverdeling

Binnen dit project is aan de betrokken partijen de volgende rolverdeling toegedeeld:

- HHNK: praktijkervaring, toetsend
- Nelen & Schuurmans: uitvoerend, leidende rol
- Deltares: adviserend, inbreng ervaring uit eerder projecten

2.2 Projectkeuzes

Voor het onderzoeksproject zijn de volgende keuzes gemaakt:

- Er is gebruik gemaakt van RTC-Tools 2.2. Dit is de laatste stabiele release ontwikkeld door Deltares.
- De regeling voor de VRNK-boezem gaat als onderzoeksregeling gebruikt worden.
- Voor de neerslag is de daadwerkelijk gemeten neerslag gebruikt.
- Het advies mee te nemen vanuit Deltares om te kiezen voor trapsgestuurd regelen en gemengde doelfunctie te gebruiken. Het eerste doel bepaalt daarmee de randvoorwaarden voor het volgende doel.
- Ervaring leert dat de RTC-Tools software nog niet volledig gereed is voor de optimalisatietechnieken mixed-integer en tree-based. Goal programming werd bij het startoverleg in 2017 gezien als de geschikte optimalisatietechniek. Dit hangt samen met de doelfunctie die samen met HHNK opgesteld is en de best-practices van Deltares.
- In 2018 hebben de ontwikkelingen binnen RTC-Tools het mogelijk gemaakt om met commerciële solvers de mixed-integer techniek toe te passen. Deze techniek is binnen dit onderzoek gekozen omdat het de potentie van MPC ten volle benut.
- De bestaande regelingen zijn beschreven in de programmeertaal R. In 2018 zijn deze regelingen omgeschreven naar de programmeertaal Python. De regelingen worden in de resultaten aangeduid als Python regeling v1.
- In september 2018 zijn de Python regelingen geüpdatet. De afzonderlijke stijg- en peilregelingen zijn samengevoegd tot één regeling. De regelingen worden in de resultaten aangeduid als Python regeling v2.

2.3 Uitgangspunten

- Onderzoeksrichting: vergelijking met huidige regelingen
- VRNK boezem wordt als één bak geschematiseerd in RTC-Tools (overeenkomend de huidige regeling)
- Mixed-integer optimalisatie toepassen indien mogelijk anders terugvallen op goal programming
- Optimaliseren voor het gehele jaar 2017.

2.4 Doelfuncties

De beschrijving van het optimalisatieprobleem zal uit drie doelen bestaan. De doelen staan hieronder opgesomd waarbij doel 1 het belangrijkst is. Dit doel zal als eerst opgelost worden waarna met de overgebleven oplossingsruimte de oplossing voor het volgende doel berekend wordt.

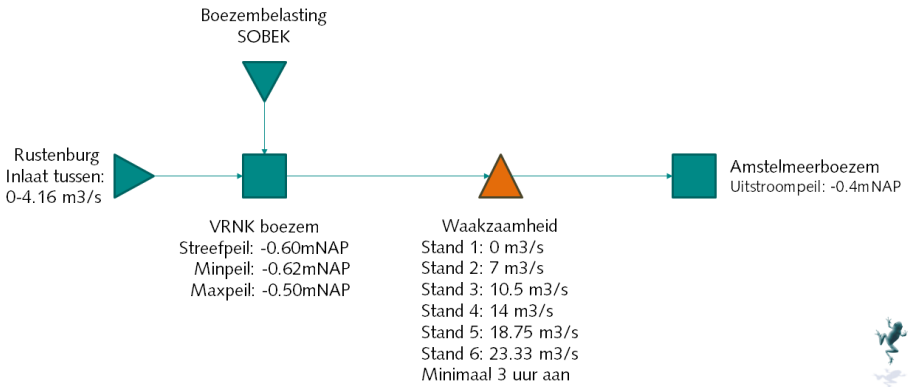
1. Peil binnen de waterstandsgrenzen (min: -0.62 mNAP / max: -0.50 mNAP)
2. Einde van de dag op streefpeil (-0.60 mNAP)
3. Optimaliseren van de energieprijs

Voor doel 1 zal het volledige bereik van de huidige regeling ingesteld worden om de ruimte in het oplossingsbereik te maximaliseren.

2.5 Modelkeuzes

Naast de doelfuncties zijn er een aantal randvoorwaardes opgenomen in het model:

- Mixed-integer trapsgestuurde pomp Waakzaamheid met 6 pompstanden.
- Het gemaal Waakzaamheid moet minimal 3 uur aanstaan.
- Rustenburg laat pas water in als het peilverschil groter is dan 0.02 m.
- Een vast streefpeil en constant Amstelmeerboezem peil voor het hele jaar.
- Er wordt per 24 uur gerekend met een tijdshorizon van 36 uur.



Figuur 1 – Schematisering van het MPC model

2.6 Vergelijking

Voor het vergelijken van de optimalisatiemethodiek en de huidige beslisregels wordt er gekeken naar een aantal toetsingscriteria onderverdeeld in drie thema's:

Inhoudelijk

- Overschrijding van grenzen
- Mate van peilfluctuatie
- Verbruikte energiekosten

Beheer

- Uitlegbaarheid
- Realiseerbaarheid
- Mogelijkheid om het in eigen beheer te nemen

Technisch

- Robuustheid
- Rekentijd

3 Resultaten

De resultaten worden beschreven volgens de drie thema's binnen de onderzoeksdoelstellingen. Er wordt gekeken hoe MPC zich inhoudelijk verhoudt tot de huidige stuurregels in Python op basis van enkele toetsingscriteria. Zoals aangegeven is gedurende het project een update uitgevoerd binnen de huidige stuurregels. Daaraan wordt op de volgende wijze gerefereerde in de resultaten:

- MPC: optimalisatiemodel in RTC-Tools
- Python regeling v1: huidige stuurregels zoals actief van 2010 – september 2018.
- Python regeling v2: huidige stuurregels actief sinds oktober 2018.

Naast deze inhoudelijke analyse is onderzocht of deze wijze van sturing gebruikt kan worden binnen het (boezem)beheer van HHNK. Als laatste maar zeker niet onbelangrijk is de technische toepasbaarheid van MPC in een operationeel systeem als ControlNEXT.

3.1 Inhoudelijk

Op inhoudelijk vlak worden de MPC regeling en de Python stuurregels vergeleken op peilfluctuatie, peiloverschrijding en energiekosten. Een groot verschil zit in de integrale wijze waarop de MPC regeling het peilbeheer beschouwd. De Python regelingen kijken eerst of er een watertekort of een overschot is om vervolgens te gaan malen of in te laten. Omdat de MPC regeling zowel per tijdstap als over de gehele horizon zoekt om de doelfuncties te realiseren, kan het optimaal zijn om gelijktijdig in te laten en te malen. Met name de doelfunctie "*einde van de dag op streefpeil*" kan tot dit gedrag leiden. Met discrete pompstappen wordt het peil tot onder streefpeil gemaald en weer aangevuld met een continue inlaatdebiet tot het streefpeildoel bereikt is.

Het model heeft over 2017 gedraaid met tijdstappen van een uur met een cyclus van een dag. Daarbij wordt steeds 36 uur vooruitgekeken voor het optimalisatieprobleem. De twaalf uur extra doorrekenen voorkomt het 'the-end-of-the-world' probleem na 24 uur. Dit probleem komt voor bij een optimalisatie waarbij niet voorbij het horizon wordt gekeken en rekening wordt gehouden hetgeen vlak erna gebeurt.

Voordat inhoudelijk wordt ingegaan op de toetsingscriteria zijn de verpompte volumina vergeleken. Deze liggen allen in dezelfde orde van grootte.

Tabel 1 - Vergelijking watervolumes.

	Volume (Mm^3)
MPC	57.35
Python v1	55.83
Python v2	57.46

De Python v1 regeling is 'afwachtend' en handelt minder adequate op verwachtingen dan de Python v2 regeling. Daardoor komt deze regeling soms buiten de gewenste bandbreedte (= onwenselijk), maar hierdoor wordt wel minder water ingelaten (= interessant). Waarschijnlijk kan bij de andere regelingen het verpompte volume gereduceerd worden als bijv. meer uitzakking wordt toestaan of anders wordt omgaan met de doelfunctie voor het streefpeil.

Peiloverschrijding

In de MPC regeling zijn de minimale en maximale peilen met prioriteit 1 als constraints opgegeven in tegenstelling tot de huidige sturing. Dit heeft tot gevolg dat bij de MPC regeling het peil als eerste ten allen tijden binnen de grenzen moet blijven terwijl bij de Python regelingen de mogelijkheid bestaat dat het peil buiten de grenzen treedt. Hierbij moet opgemerkt worden dat het voor de MPC regeling wel mogelijk is om over deze grenzen te komen als de pomp- of inlaatcapaciteit ontoereikend is. Echter is het voor het jaar 2017 binnen de MPC regeling niet nodig geweest. De door de peilbeheerder opgelegde instellingen leiden in de Python regelingen v1 tot een aantal momenten dat het peil onder of boven de grenzen komt. In de vernieuwde Python v2 sturingsregeling blijft het RBP binnen de waterstandsgrenzen. In de v1 regeling werd bij geleidelijke peilstijging meer marge toegestane bij de verschillende pompstappen. Als bij een peilverhoging alsnog een sterke stijging plaatsvond door hevige neerslag dan werd daarop te laat geanticipeerd. In de v2 regeling kan deze situatie zich niet meer voordoen.

Figuur 2 – Vergelijking tussen de MPC regeling en de Python regeling v1 en v2.

Peilfluctuatie

De MPC regeling maakt optimaal gebruik van de bandbreedte tussen het minimale en maximale peil. De doelfunctie *'einde van de dag op streefpeil'* zorgt ervoor dat de MPC regeling het peil niet continu tegen het maximum peil aanhoudt. Dit in tegenstelling tot de Python regelingen, die kennen deze doelfunctie niet. Dit verschil komt duidelijk naar voren bij het cumulatieve peilverschil van beide regelingen in 2017.

De MPC regeling fluctueert over het jaar ongeveer 50% meer dan de Python regelingen. Het dagelijks terugkeren naar streefpeil is een inefficiënte doelfunctie in combinatie met de randvoorwaarde om minimaal drie uur achtereen te pompen en een ondergrenspeil van -0.62 mNAP. De combinatie resulteert in pompen en inlaten binnen eenzelfde periode. De standaarddeviatie van de peilfluctuatie is dan ook hoger voor de MPC regeling. Dit geldt ook voor Python regeling v1 in vergelijking tot versie 2. Zoals eerder aangedragen heeft dat te maken met het adequate handelen van de versie 2 regeling bij een geleidelijke stijging van het waterpeil.

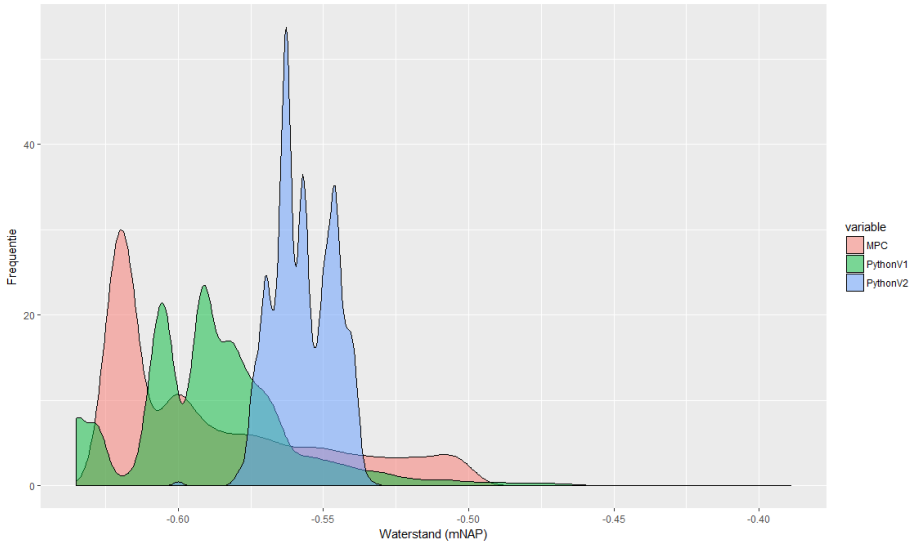
Opvallend is dat de MPC regeling frequenter kiest om de dag te beginnen door lang uit te blijven waarbij het peil oploopt tot de bovengrens. De dag wordt afgesloten door te pompen tot aan streefpeil. Het eindresultaat (op streefpeil) zou ook bereikt kunnen worden door eerder op de dag meer water te verpompen. Met de goedkoopste energieprijzen gedurende de nacht is de vraag of de MPC regeling dus wel optimaal zijn pompuren over de dag heeft verdeeld. Er zijn hiervoor twee verklaringen:

- Het optimalisatieprobleem staat niet toe dat het peil verder dan 2 cm onder streefpeil wordt getrokken. Het is daarom niet altijd een optie om 's nachts te pompen tot onder streefpeil en het peil gedurende de dag te laten oplopen tot aan streefpeil. Dit zorgt ervoor dat het optimaler is om pas later op de dag pompen.
- Met de restrictie van minimaal drie uur pompen is het ideaal om eind van de avond aan te gaan en de goedkope uren van de avond en begin van de nacht mee te pakken (23:00, 0:00, 01:00). Deze eerste uren van het nieuwe etmaal worden uiteindelijk weer genegeerd doordat enkel de eerste 24 uur van de 36 uur voorspelhorizon bewaard blijven.

Tabel 2 - Peilfluctuatie bij de verschillende regelingen.

	MPC regeling	Python regeling v1	Python regeling v2
Cumulatief peilverschil (meter)	33,4	22,5	22,1
Standaard deviatie (meter)	0,036	0,029	0,010
Mediaan (meter)	-0,601	-0,591	-0,557
1e Kwartiel (meter)	-0,620	-0,604	-0,564
3e Kwartiel (meter)	-0,564	-0,575	-0,548

Van de berekende waterstanden is een kansdichtheidsdiagram gemaakt. Voor de drie regelingen is deze weergegeven in *Figuur 3*. Daarin komen de doelstellingen van de MPC regeling duidelijk terug: het peil bevindt zicht tussen -0.62 en -0.5 mNAP met het doel om aan het einde van de dag op het streefpeil van -0.6 mNAP uit te komen.



Figuur 3 – Frequentiediagram van de berekende waterstanden.

De Python v1 regeling toont de grootste variatie in waterstanden. Met de afzonderlijke stijg- en peilregeling kan het peil verder oplopen dan met de Python v2 regeling. Daarbij komt de waterstand in de Python v1 regeling ook incidenteel buiten zijn bandbreedte wat niet wenselijk is. Deze regeling is daarom ook geüpdatet naar de v2 regeling.

In de Python v2 is gesimuleerd met dag/nacht instellingen. Bij de regeling ligt overdag het aanslagpeil op 3.8 cm boven streefpeil voor een verwachte waterstand voor de komende 8 uur. Bij bereiken van dit peil wordt gemalen tot 2.5 cm boven streefpeil. 's Nachts ligt het aanslagpeil op 2.5 cm boven streefpeil tot het uitslagpeil op streefpeil. Dit zie je ook terug in *Figuur 3*.

Energiekosten

De Python regelingen draaien via een interface met Eneco (BiedOptimaal) voor het optimaliseren op APX prijzen. Deze interface is niet toegankelijk voor historische analyses. Daarom is voor dit onderzoek het advies van de Python regeling verrekend met de APX prijzen en naast de dag/nacht tarieven gelegd. Dit om alle randvoorwaarden zo gelijk mogelijk te houden. De MPC regeling heeft kostenminimalisatie als 3^{e} en laatste doelfunctie. De spelingsruimte wordt bepaald door de restricties in de doelen 1 en 2.

Het effect van deze doelfunctie is gekwantificeerd door de totale energieprijs te vergelijken met een nulscenario. Het nulscenario is berekend door de pompmomenten uit de MPC regeling te vermenigvuldigen met de daggemiddelde APX prijs. In andere woorden, een constante APX prijs per dag. Daaruit volgt dat de MPC regeling zo'n 9% heeft bespaard door te sturen op energieprijzen. Deze besparing wordt beperkt door het gedrag dat de optimale oplossing neigt naar pompen aan het einde van de dag om het tot streefpeil te komen zoals eerder is beschreven. Kortom, de formulering van het optimalisatieprobleem is niet optimaal voor de reductie van energiekosten.

De Python regelingen hebben gesimuleerd zonder APX optimalisatie, maar zijn wel verrekend met de APX prijzen. Voor de Python regelingen leidt het in dit geval tot lagere kosten, maar dit had ook hoger kunnen uitvallen. Daarbij geeft het een beeld van de variatie in APX kosten.

De Python v1 regeling heeft geen voorkeur voor dag/nacht of een andere vorm van sturing op energie. Daarentegen heeft de Python v2 regeling gedraaid op dag/nacht instellingen wat inhoudt dat tussen 23:00-7:00 uur een lager aanslagpeil geldt dan overdag. Daarmee behaalt de Python regeling v2 als enige ook lagere kosten op dag/nacht tarief. De sturing op dag/nacht tarief valt voor de MPC en Python v1 hoger uit. Dit komt voornamelijk omdat de APX prijs gemiddeld onder het nachttarief ligt.

Tabel 3 – APX en Dag/Nacht prijs voor de verschillende regelingen

	MPC regeling	Python regeling v1	Python regeling v2
APX tarief (€/jaar)	2799,64	2804,55	3030,33
Daggemiddelde APX tarief (€/jaar)	3064,14	2968,50	3061,64
Besparing sturing (%)	8,6	5,5	1,0
Dag/Nacht tarief (€/jaar)	3291,56	3200,46	3016,49

3.2 Beheer

De MPC regeling is complexer dan de Python sturingsregels, zowel in opzet als inzichtelijkheid. Het aanpassen of debuggen van het model vergt kennis van RTC-tools en ondersteuning van Deltares. Door de complexe aard van een optimalisatiemodel is het niet mogelijk om te herleiden waarom een bepaalde oplossing de meest optimale is. Het optimalisatieprobleem neemt namelijk alle mogelijkheden mee in de afweging. Voor de Waakzaamheid enkel betekent dit met 6 pompstanden over 36 tijdstappen al ruim $1x\ 10^{28}$ mogelijkheden.

Bij de MPC regeling kan een bepaalde draai aan de knoppen een volledig ander optimaal antwoord genereren zonder dat wordt aangegeven waarom. Deltares speelt met de gedachten om een functionaliteit toe te voegen waarbij gedurende de simulatie informatie wordt gepresenteerd met een indicatie over de genomen keuzes. Dit zal leiden tot meer begrip omtrent de uitkomsten.

Met de Python regelingen is het mogelijk om te herleiden welke keuzes gemaakt zijn, omdat de regelingen werken op basis van een beslisboom. Indien een waterstand voor de komende 8 uur een aanslagpeil overschrijdt dan zal de betreffende pompstand geadviseerd worden totdat het afslagpeil bereikt wordt. Hierbij moet worden opgemerkt dat de expertise om dit binnen ControlNEXT te achterhalen op dit moment niet aanwezig is bij HHNK. Om de regelingen in eigen beheer te nemen dient een duidelijk verschil aangemerkt te worden. Dit is in onderstaande tabel geduid.

Tabel 4 Verschillen tussen MPC en Python regeling.

	MPC regeling	Python regeling
Type software	Applicatie	Bibliotheek
Software	Java, Python, Modellica, CPLEX	Python
Onderliggende theorie	Complex	Eenvoudig
Community	Groeiend	Geen (maatwerk)

3.3 Technisch

Voor het optimaliseren binnen een MPC regeling is de *solver* keuze van belang. RTC-tools maakt standaard gebruik van open-source solvers die niet voor elk optimalisatie probleem geschikt zijn. De MPC regeling is zo opgezet dat gemaal Waakzaamheid als een getrapte *mixed-integer* pomp in het model zit. Dit houdt in dat de solver alleen mag kiezen uit de opgegeven pompstanden. Doordat de solver niet vrij mag kiezen tussen een minimaal en maximaal pompdebiet, maar vast zit aan de gedefinieerde pompstanden, kost het de solver veel tijd om tot de optimale oplossing te komen.

Met de open-source solvers die RTC-tools standaard aanroept wordt de rekentijd te lang. Bij een horizon van 10 uur komt het model in 15 minuten tot een oplossing, maar bij een horizon van 11 uur schiet dit exponentieel omhoog naar een rekentijd van uren. Op dat moment is het model niet meer robuust voor een operationeel systeem. Vanwege de lange rekentijd is besloten binnen dit onderzoek het model te draaien met de commerciële solver CPLEX, ontwikkeld door het softwarebedrijf IBM. Deltares is in het bezit van een licentie voor deze solver. Hiermee kan het model een horizon van 36 uur binnen een halve minuut doorrekenen.

Naast het type solver is de robuustheid afhankelijk van verdere instellingen. In dit model is aandacht besteed aan fine-tuning van de optimalisatieparameters. Deze parameters geven aan wanneer precies aan een doelstelling wordt voldaan. Bijv. bij de doelfunctie *"einde van de dag op streefpeil"* is -0.6 mNAP opgegeven als streefpeil. Maar voldoet het peil pas aan de doelstelling bij -0.6001 mNAP of al bij -0.61 mNAP. Dit zijn belangrijke instellingen om te analyseren voordat een MPC regeling operationeel wordt ingezet.

3.4 Samenwerking Deltares

Deltares heeft vanuit het TKI project ondersteuning geleverd bij het opzetten van het RTC-tools model. De ondersteuning bestond uit:

- Twee werkdagen bij Deltares
- Het RTC-Tools models koppelen aan de CPLEX solver
- Draaien van het model met de commerciele solver CPLEX
- Advies en klankbord

Tijdens de werkdagen bij Deltares is intensief samengewerkt met het RTC-tools team om de uitdagingen van de VRNK boezem goed in het model op te nemen. Ook is er gekeken naar de mogelijke solvers om het probleem op te lossen. Uiteindelijk is er gekozen voor de commerciele solver (CPLEX).

Naast deze dagen bij Deltares is het model een aantal keer aangepast door N&S om vervolgens te laten doorrekenen door Deltares. Voor het doorrekenen heeft Deltares een aantal verwerkingsslagen gedaan om het model geschikt te krijgen voor CPLEX.

Verder is er uitgebreid mailcontact geweest waarin adviezen vanuit Deltares werden aangedragen en ondersteuning bij het debuggen van het model.

4 Conclusies en aanbevelingen

Het doel van dit onderzoek is inzicht krijgen in de mogelijkheden van een MPC regeling ten opzichte van de Python sturingsregeling voor het ControlNEXT systeem bij HHNK. Gedurende het project is een update van de Python regelingen doorgevoerd. De geüpdatete versie is ook meegenomen in de analyse. Uiteindelijk is een vergelijking gemaakt tussen:

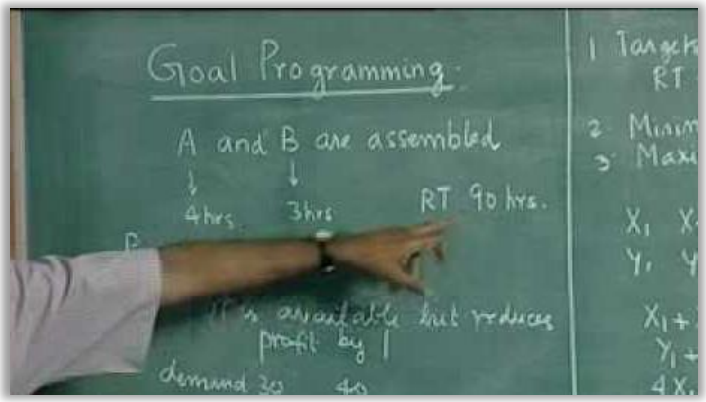
- MPC: optimalisatiemodel in RTC-Tools
- Python regeling v1: huidige stuurregels zoals actief van 2010 – september 2018.
- Python regeling v2: huidige stuurregels actief sinds oktober 2018.

4.1 Conclusies

In deze studie is een MPC regeling ontwikkeld en vergeleken met de huidige beslisregels. De vergelijking is uitgevoerd op drie thema's: inhoudelijk, beheer en technisch. Dit heeft geleid tot de volgende conclusies:

› De Python regeling v1 stuurt onvoldoende op peilhandhaving waardoor het peil geregeld buiten de wenselijke bandbreedtes valt. Dit is los van dit project reeds geconstateerd en heeft geleid tot implementatie van de Python regeling v2. De Python v1 regeling is hierdoor geen realistische vorm van sturing om in gebiedsregelingen toe te passen. Deze sturing leert ons echter wel dat minder strak sturen op peil een forse afname van het inlaatdebiet kan opleveren.

› In principe leidt sturing in de vorm van een MPC regeling (geprioriteerde goal functies) tot een optimalere oplossing dan sturing via vooraf gedefinieerde beslisregels, omdat dergelijke regels onbedoeld een optimalisatiedomein kunnen uitsluiten. De mate waarin MPC (of sturing op beslisregels) op energie kan sturen, hangt sterk af van de speelruimte die de regeling krijgt. Als er teveel en/of onnodig beperkende functies van hogere prioriteit zijn gedefinieerd resteert er onvoldoende ruimte om te sturen op energie. Met de doelfuncties toegepast in deze studie is een beperkte winst in energiekosten geboekt. Tevens is sturing op energieprijs niet toegepast op de Python regelingen waardoor een vergelijking met de MPC regeling niet goed mogelijk is. Tot slot is het vermoeden dat het aanhouden van ruime marges voor het peil verreweg de belangrijkste factor is voor het besparen op energiekosten. Het creëren van die marge kan zowel onder een MPC als een sturing met beslisregels.



- De huidige software voor (operationele) sturing via een MPC model is nog verre van optimaal. Het is hierdoor niet mogelijk om alle potentie van MPC te benutten. Met de huidige solver lopen de rekentijden voor het oplossen van het optimalisatieprobleem als snel hoog op voor de gewenste tijdshorizon. De software raakt in de problemen bij toename in de optimalisatiehorizon, gebruik wordt gemaakt van mixed integer instellingen (bijv. pomp stappen) en/of rekening wordt gehouden met onzekerheden (tree-based optimalisatietechniek).

- MPC wordt naarmate het aantal doelfuncties toeneemt relatief steeds effectiever ten opzichte van sturing op basis van beslisregels. Als er rekening gehouden moet worden met (te)veel handelingen en/of domeinen zijn de gevolgen van een beslissing voor mensen niet meer te overzien en is het opzetten van een goede beslisboom niet goed mogelijk.

- Bij eenvoudige sturing is de methodiek van sturing via beslisregels eenvoudiger te implementeren en beheren dan MPC. Als het aantal factoren waarmee rekening moet worden gehouden (doelfuncties) toeneemt, wordt MPC met geprioriteerde doelen echter overzichtelijker dan een regeling op basis van een beslisboom.

- Tijdens het uitvoeren van het onderzoek is gebleken dat er veel ontwikkeling plaats vindt in het softwarepakket voor MPC, RTC-Tools, waardoor deze techniek op afzienbare termijn waarschijnlijk in al haar facetten is in te zetten.

- Bij MPC is niet expliciet na te gaan waarom een bepaalde beslissing optimaal is. Dit vereist dat de doelfuncties één voor één worden toegevoegd en gevoeligheidsanalyses met instellingen nodig zijn. Door de resultaten van een extra functie of een aangepaste functie over een historisch jaar te beoordelen kan worden ingeschat hoe groot en wenselijk het effect is.

4.2 Aanbevelingen

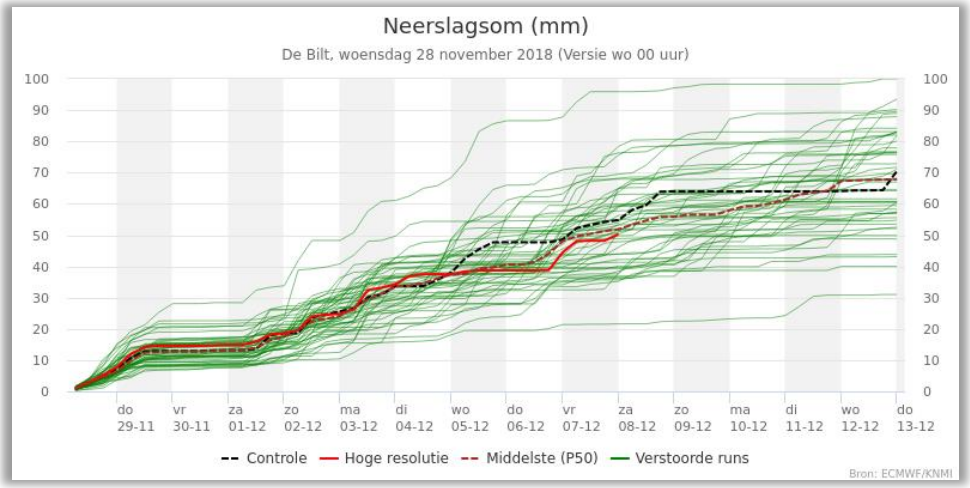
Dit onderzoek gaf veel nieuwe inzichten waardoor verdere ideeën ontstonden. Daaruit volgen onderstaande aanbevelingen:

- Het uitvoeren van een gevoeligheidsanalyse en/of onderzoeken van meer scenario's op een vergelijkbare manier als in dit onderzoek is gedaan. Via een dergelijke analyse kan bijvoorbeeld worden bepaald:
    - Hoeveel energiewinst een MPC regeling kan behalen met ander type doelfuncties.
    - Hoeveel reductie van inlaatwater kan worden behaald.
    - Wat het effect is van het incrementeel verder oprekken van marges op de energiekoste.
    - Welke marges zijn minimaal nodig en tot hoe ver loont het om de marges verder op te rekken? Voor het oprekken van de marges is doelfunctie 2 van belang. De beperkende invloed van deze functie kan worden weggenomen door een marge rond streefpeil toe te kennen of de voorspelhorizon te vergroten en ver mogelijk in de toekomst terug te keren naar streefpeil.

- Hoe de verschillende regelingen presteren bij hevige neerslag en in periode van droogte.
- Of verder optimalisatie binnen het waterbeheer mogelijk is door bijv. het aantal schakelmomenten, de duur van afvoeren op maximale capaciteit of het fluctueren van het inlaatdebiet te verkleinen.

› Voorlopig wordt niet aanbevolen om over te stappen van sturing op beslisregels naar sturing via MPC omdat de huidige sturing niet ingewikkeld is (kent maar een beperkt aantal doelen) en er niet wordt geoptimaliseerd. Als gezocht wordt naar optimalisatie zit de grootste winst in andere aspecten van de gebiedsregeling dan het type sturing (sturing op beslisregels versus MPC).

› Als sturing meer omvattend wordt (meer doelfuncties krijgt of rekening moet gaan houden met onzekerheden) wordt aanbevolen over te stappen van de huidige sturing op basis van beslisregels naar sturing via MPC. Dit heeft echter alleen zin als er een functioneel beheerder komt die de kennis en ervaring heeft om dergelijke regelingen te analyseren, op te zetten en te verbeteren. Tevens moet de MPC software ver genoeg zijn doorontwikkeld waardoor de beoogde doelfuncties naar behoren kunnen worden opgelost.



I. Lessons learned

Informatie over de Solvers

IPOPT: Interior Point Optimizer (https://projects.coin-or.org/Ipopt)

BONMIN: Basic Open-source Nonlinear Mixed Integer programming (Branch and bound method) (https://projects.coin-or.org/Bonmin)

CPLEX: Commerciele solver van IBM. Afhankelijk van het project beschikbaar bij Deltares

https://www.ibm.com/analytics/cplex-optimizer

Tips-and-tricks

Model horizon:

Het aantal tijdstappen en de grote van elke tijdstap die RTC gebruikt is afhankelijk van de timeseries_import-file. Dus afhankelijk van wat vanuit FEWS wordt opgelegd.

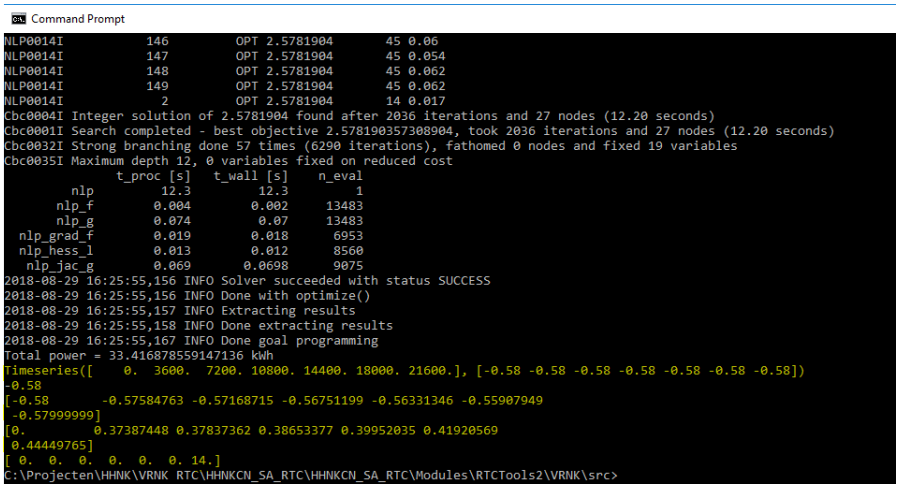
Import / Export tijdseries met FEWS:

- Maak gebruik van "*from rtctools.optimization.pi_mixin import PIMixin*"
- Configureer "*rtcDataConfig.xml" (deze staat samen met rtcParameterConfig.xml in de Input map)* waarin:
    - timeSeriesId de naam draagt van de tijdserie uit het model
    - locationId & parameterId gelijk zijn aan FEWS

  Je definieert hierbinnen de mapping voor zowel de import als de export. Een IdMapping in FEWS is **NIET** meer nodig!

- Zorg dat je een preprocessing stap maakt in FEWS waarin je in ieder geval interpoleert. RTC wil geen NaN's of -999's.

Output weergeven aan het einde van je modelrun in je terminal:



Voeg aan het einde van je script het volgende toe:

"print(results['*Variabele*'])"

Bijvoorbeeld: "print(results['waakzaamheid.pump1.Q'])" print voor elke tijdstap de Q van pomp1 uit.

Bijvoorbeeld: "print(self.timeseries_at('storage_streef', self.times()[-1]))" print voor de laatste tijdstap het streefpeil uit

LET OP! Bovenstaande variabelen zijn afhankelijk van het Modellica model.

Solver options:

Afhankelijk van de solver kan je verschillende opties meegeven. Deze opties hebben invloed op de manier waarop de solver het probleem aanpakt. Zie urls voor opties:

- Ipopt solver: https://www.coin-or.org/Bonmin/option_pages/options_list_ipopt.html
- Bonmin solver: https://www.coin-or.org/Bonmin/option_pages/options_list_bonmin.html

Specifiek voor VRNK (vanuit correspondentie met Deltares)

CPLEX was sowieso moeilijk te draaien met quadratic constraints, dus alle constraints lineair gemaakt.

Aandachtspunten

- parameters.xml ; H.berekend.rtc hoge resolutie gemaakt (float ipv round to 0.001). Dat is wel zo netjes als je hem als state doorgeeft, anders is het een beetje vals spelen.
- GA_RTC_VRNK.xml / rtcDataConfig.xml / parameters.xml: History doorgeven van pump status.
- Toestaan dat de solver timet out met 120 s. Normaal gesproken wil je dit echt niet, maar omdat het een desk-studie is (geen operationeel systeem) met een paar zeer grove/bijzondere aannames (zie onderstaand), is het OK. De solver heeft namelijk erg veel moeite met bewijzen dat zijn antwoord optimaal omdat de Q volledig in stapjes gaat (en dus niet in bepaalde mate continu is). Typisch rekentijd is ~20 sec.

Als laatste nog enkele losse opmerkingen:

- De pomp is ook vreemd gemodelleerd is met de debietstappen. Frequentiestappen zou beter zijn (maar geen gegevens over welke frequenties dan), maar het liefst natuurlijk gewoon "staploos" tussen een bepaalde minimum en maximum frequentie. Dat reduceert het aantal integers sterk.
- Fit doe je normaal op iets van >50% efficiency, 40-100% rpm. Omdat de debietstapjes wat raar zijn, en omdat je wil voorkomen dat RTC zijn water niet meer kwijt kan, heb ik nu maar een fit van de pomp (volgend uit de Excel sheet o.b.v Q0 en H0) gedaan op Q >= 7 m3/s, en H >= 0.05. De working area is dan ook meteen een vierkant (met nog een iets ruimere dH van >= 0.001).