Rijnland RTC-Tools 2 Optimalisatie - Architectuur

Dit document beschrijft de Rijnland Optimalistie structuur.

Allereerst wordt een korte beschrijving gegeven hoe de optimalisatie run gestart kan worden op Windows/Linux.

Vervolgens wordt een uitgebreid overzicht gegeven van de classes en functies, die samen met de modellen het Optimalisatie probleem beschrijven. Dit is opgebroken in de volgende elementen:

- Basis Model/Class definities voor het Rijnland model
- Structuur van de Rijnland_Boezem_Optimization script stappen
    - In 'Optimization' mode
    - In 'Post-processing' mode

Daarna wordt een voorstel gedaan hoe het hoofd script geherstructureerd kan worden, voor betere toekomstige onderhoudbaarheid.

Als laatste wordt een overzicht gegeven van de RTC-Tools directory structuur voor Rijnland. Hierbij wordt (de functie/inhoud van) de volgende directories uitgebreid beschreven:

- Input
- Model
- Src (Source)
- Output

Ook wordt er nog ingegaan op de verschillende elementen die gebruikt zijn in de Modelica schematisatie van het Rijnland systeem.

Start van Optimalisatie run

Het Rijnland Optimalisatie model draait door het aanroepen van het Optimalistie script:
/src/Rijnland_Boezem_Optimization.py

Windows:
Het script wordt aangeroepen door *RTC2_Python_silent.bat* (Onderdeel van de RTC-Tools Windows installatie)

```
call "d:\Software\RTCTools2\system\JModelica\RTC2_Python_silent.bat" .\Rijnland_Boezem_Optimization.py
```

Linux:

```
. scl_source enable python27
. scl_source enable devtoolset-6  # For FMU compilation
# Activate virtual environment in which the optimization runs. The venv has acces to the RTC-Tools routines
. /opt/rtc-tools/venv/bin/activate

cd src/
exec python Rijnland_Boezem_Optimization.py
```

Basis Model/Class definities voor het Rijnland model

In de class *Rijnland_base(object)* class, in *Rijnland_Boezem_Optimization.py*, worden eigenschappen van het model/de optimalisatie opgeslagen, zodat ze gebruikte kunnen worden voor verschillende run modi:

1. Definiëren van Modelica model id: *'NetworkOptimization*'
2. Zetten van de *modelica_library_folder* waarin de definitie van de model elementen opgenomen is:
    1. */model/rtc-tools-channel-flow*
3. Opgeven van alle rtcParameterConfig files die door RTC ingelezen moeten worden:
    1. */input/rtcParameterConfig_Boezem*
    2. */input/rtcParameterConfig_Boezemgemalen*
    3. */input/rtcParameterConfig_Kanalen*
    4. */input/rtcParameterConfig_Goals*
    5. */input/rtcParameterConfig_Polders*
    6. */input/rtcParameterConfig_Knopen*
    7. */input/rtcParameterConfig_Boezem-PID*
4. Naast de standaard output van RTC-Tools (de optimalisatie variabelen) willen we meer tijdseries wegschrijven naar de output. Voor zowel de knopen (storage_nodes) als de kanalen (reaches) definieren we aanvullende output tijdseries in *additional_output*
5. We definiëren lijsten met de gemalen, in groepen:
    1. Emergency_pumps
    2. One_way_pumps
    3. Two_way_pumps

    Afhankelijk van de groep classificatie, wordt een gemaal wel/niet gebruikt voor bepaalde optimalisatie routines.

6. Voor de boezem gemalen wordt de relatie tussen pompstand en discreet debiet vastgezet in de dictionary *pump_stages*. Deze zal in de de 'Post-processing' mode van de optimalisatie gebruikt worden om de geoptimaliseerde continue gemaal inzet te vertalen naar een discrete pompstand.

Structuur van de Rijnland_Boezem_Optimization script stappen

In eerste instantie draait het script het Rijnland model in 'optimization' mode. Dit is de optimalisatie run (met alle geprioriteerde doelen) inclusief alle benodigde initialisatie en pre-processing runs.

Als de 'Optimization' mode klaar is, is er een optimaal resultaat berekend voor de inzet van de gemalen in 'continue' vorm. Vervolgens wordt het model in de 'Post-processing' mode gedraaid, om de continue gemaal inzetten naar discrete pompstanden te vertalen.

"Optimization" mode

De eerste stap is het draaien van het ApplyEmergencyClosure script.

1. */src/ApplyEmergencyClosure.py*

a) leest */input/rtcParameterConfig_Noodwaterkeringen.xml* om de gesloten branches (kanalen) te bepalen

b) past */model/Network.mo* aan

c) output */input/closed_reaches.csv*

Daarna wordt het InitializeStates script gedraaid, om de model states te creëren op basis van de waardes (gemaal afvoer/waterstanden/chloride concentraties) vanuit Delft-FEWS

2. /src/InitializeStates.py

a) laad initial_state waarde van */input/timeseries_import.py*

b) draait een optimalisatie, gebruikmakend van de RTC-Tools InitialStateEstimationMixin. Initiele waardes (states) voor de optimalisatie run die zo dicht mogelijk liggen bij de (gemeten) waardes, zonder de model vergelijkingen te schenden.

c) outputs */input/timeseries_import-initialized.xml*

Vervolgens wordt er een volledige simulatie gedraaid (geen speciale regels) om een eerste schatting van linearization parameters te krijgen.

3. Draaien van de *Rijnland_Full_Presim* class (onderdeel van */src/Rijnland_Boezem_Optimization.py*)

a) leest */input/timeseries_import-initialized.xml*

b) draait een volledige rule-based simulatie run om een eerste schatting van linearization parameters te krijgen

c) output */input/timeseries_import-full_initialization.xml*

Nu het 'voorbereidende werk' qua initialisatie voltooid is, kan de Optimalisatie run gedraaid worden

7. Draaien van de *Rijnland_Optimization* class (onderdeel van */src/Rijnland_Boezem_Optimization.py*)

a) Leest */input/timeseries_import-initialized.xml* en */input/timeseries_import-full_initialization.xml*

b) Een *collocated-integrated optimalisatie* over alle tijdstappen, waarbij in het proces verschillende routines gedraaid worden:

i) Opzetten en vullen van diverse 'dictionaries' en 'lists' die gebruikt zullen worden in de optimalisatie

ii) Het zetten van een aantal 'boolean' parameters naar true/false om opties van de run aan/uit te zetten.

iii) *pre()* routine

(1) Zetten van verschillende variable die later gebruikt worden

Voor goal variabelen (het wel/niet draaien van een bepaald doel) wordt er op dit moment nog direct goal parameters ingelezen uit de parameterConfig file. Met de introductie van 'regimes' zullen de parameters door een apart script naar de parameterConfig file geschreven worden, waarna ze ingelezen kunnen worden

(2) Adjust Timesteps

Er is een blok code om de tijdstappen aan te passen, mits die optie geactiveerd is door de gebruiker.

(3) Bounds

Inlezen en bepalen van bounds tijdseries die gebruikt worden in de optimalisatie:

- Waterstanden van kanalen
- Gemaal debieten
- Chloride concentratie

(4) Polders

Inlezen van polder gerelateerde informatie uit input/lookuptables/*csv files en het creeeren van polder bounds tijdseries

(5) Gouda: *use_Cl_flushing_goal*

Er is een serie met bewerkingen die als doel hebben om Q_req tijdseries voor te bereiden voor de M&T polder en gemaal inlaat bij Gouda – indien de *use_Cl_flushing_goal* True is.

iv) Specifieke RTC-Tools routines om waardes benodigd voor de optimalisatie in te lezen/te definieren

(1) def parameters

(2) def path_variables

(3) def bounds

(4) def constraints

(5) def goals

(6) def path_goals

(a) in de path_goals functie worden de optimalisatie doelen met verschillende prioriteiten toegevoegd. De algemene opzet is dat er een if-statement is die checkt of een bepaald doel toegevoegd moet worden op basis van de ingelezen parameters/ de regime keuze.

(b) Er wordt gebruik gemaakt van verschillende gedefinieerde goal classes.

(i) class WeighPumps

(ii) class MinimizePumpCost

(iii) class WaterVolumeRangeGoal

(iv) class WaterLevelRangeGoal

(v) class WaterLevelSetpointGoal

(vi) class PumpDownstreamWaterLevelRangeGoal

(vii) class ChlorideRangeGoal

(viii) class DischargeGoal

(ix) class MinimizeQpumpGoal

(x) class MinimizeInitialDerQ2

Als alle doelen gezet zijn, wordt het optimalisatie probleem uitgevoerd door de RTCTools HomotopyMixin

Van theta = 0 (gelineariseerde (convexe) versie van het optimalisatie probleem) met stappen delta_theta opbouwend naar theta = 1 (niet-convexe vorm van het optimalistie probleem). Voor elke theta wordt de GoalProgrammingMixin gebruikt voor het uitvoeren van de optimalisatie run per prioriteit – van laag naar hoog.

c) De *priority_started()* functie draait de *cost minimizations* (minimaliseren pompkosten/prioritering) optimalisatie stappen na elke prioriteit (als theta = 1) slaat de resultaten (geminimaliseerde pompkosten) per pomp op in */output/pump_costs.csv*

d) De *priority_completed()* slaat de complete resultaten van de optimalisatie op en logt een samenvatting in */output/solution_path.csv*

   i) Er wordt gebruik gemaakt van de *process_sol_path_output()* functie (onderdeel van */src/Rijnland_Boezem_Optimization.py*)

e) *post()* routine

   i) Draait *discrete_setpoint_controller()* functie (onderdeel van */src/Rijnland_Boezem_Optimization.py*) om discrete pompstanden te bepalen, zo dicht mogelijk bij de geoptimaliseerde continue resultaten

   ii) Draaien van de *Rijnland_PostSimulation* class om het systeem te modelleren met de discrete gemaal afvoeren, output naar */output/ timeseries_export-postprocessed.xml*

"Post-processing" mode

1. *Rijnland_Post_Process* class (onderdeel van */src/Rijnland_Boezem_Optimization.py*)

a) Laden van *timeseries_import-rerun.xml*

   i) gebruikt discrete_setpoint_controller() functie (onderdeel van */src/Rijnland_Boezem_Optimization.py*) om discrete pompstanden te bepalen, zo dicht mogelijk bij de geoptimaliseerde continue resultaten

   ii) Draaien van *Rijnland_Rijnland_Post_Process* class om het systeem te modelleren met de discrete gemaal afvoeren, output naar */output/timeseries_export.xml*

Voorstel voor refactoring Rijnland_Boezem_Optimization.py

Rijnland_Boezem_Optimization.py bevat op dit moment veel verschillende classes en functies, die de file lang en onoverzichtelijk maken. De code is daardoor op de langere termijn minder goed onderhoudbaar. Ook bij toekomstige uitbreidingen is er daardoor een risico dat er zaken mislopen.

De volgende voorstellen voor code restructuring kunnen gedaan worden:

1) Verplaats 'helper functions' naar aparte file: helper_functions.py
   - *discrete_setpoint_controller()* functie
   - solution path functies (*process_sol_path_output()* en *sol_path_entry()*)

en roep deze in het hoofd script aan:
*from helper_functions import **

2) Alle 'goal' classes kunnen naar een afzonderlijk python file verplaatst worden

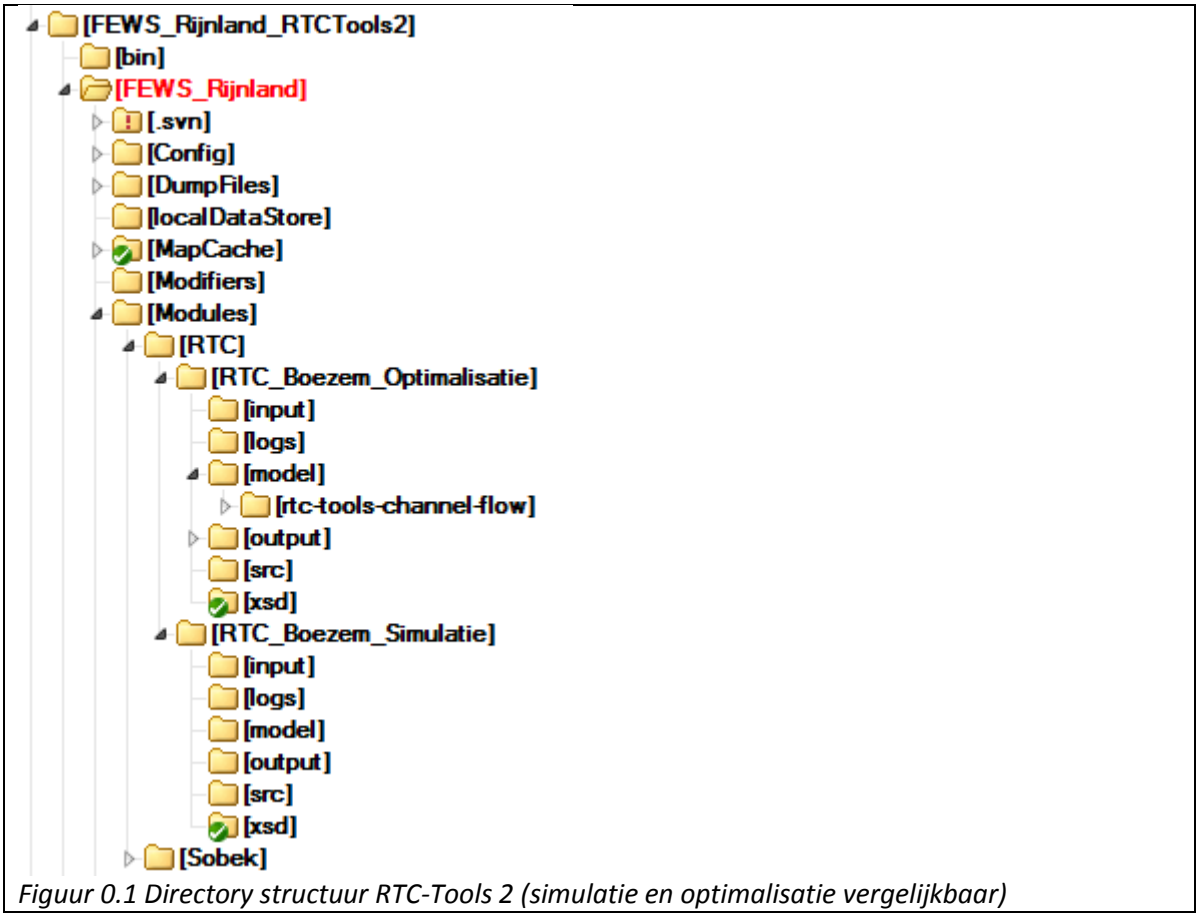
3) De functie waarmee de regime keuze vertaald wordt naar goal parameters kan ook in een afzonderlijke file geplaatst worden, en uitgevoerd worden in de pre() routine – als onderdeel van de Optimization mode

De Optimalisatie en Simulatie classes kunnen moeilijker verplaatst worden, ten gevolge van de class overerving. Er wordt voorgesteld deze in het hoofdscript te houden.

RTC-Tools directory structuur

Hierbij een overzicht van de RTC-Tools directory structuur. In dit project is de basis locatie voor de RTC-Tools runs onder de Delft-FEWS hoofddirectory, in de *Modules**RTC* folder.

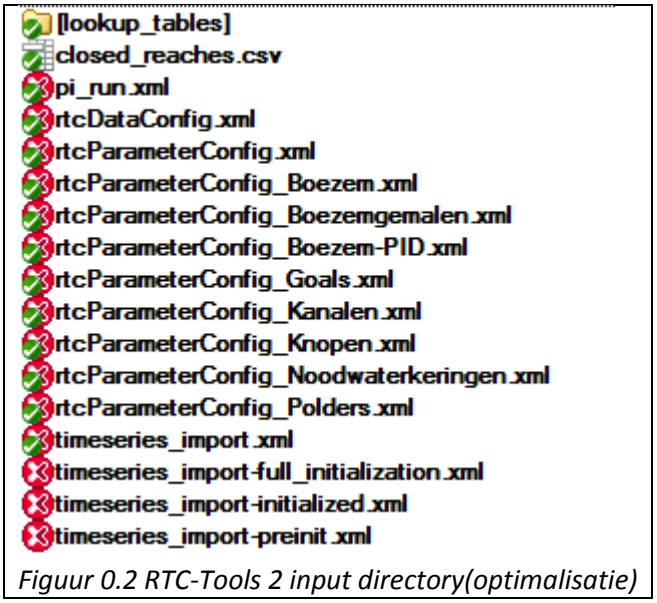
Binnen deze mappen wordt de RTC-Tools run opgezet in een relatief starre directory structuur, die weergegeven is in *Figuur 0.1*. RTC-Tools gaat ervan uit dat deze structuur bestaat. Het is dus niet mogelijk om de namen van de verschillende directories naar eigen inzicht te veranderen.



Figuur 0.1 Directory structuur RTC-Tools 2 (simulatie en optimalisatie vergelijkbaar)

Input directory

Alle inputs en parameters bevinden zich in de *input* directory, zie . Deze directory bevat een xml file met alle tijdreeksen, *timeseries_import.xml*. Deze file wordt door FEWS geëxporteerd.



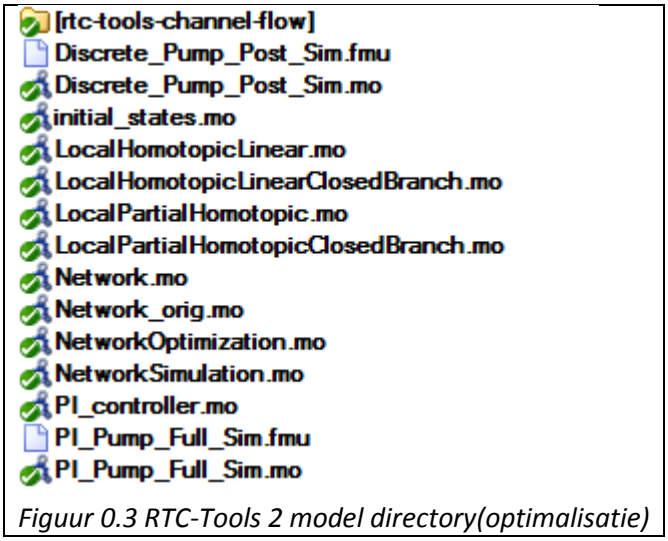
Figuur 0.2 RTC-Tools 2 input directory(optimalisatie)

In de optimalisatie run zijn de initiële waterstanden niet afkomstig uit een state file, en worden de waterstanden en chloride concentraties voor de model knopen op T0 (metingen) en de initiële gemaal debieten geëxporteerd naar de *timeseries_import.xml*. In de optimalisatie case wordt er vooraf een initialisatie gedraaid waarbij de initiële condities zo bepaald worden dat ze consistent zijn met alle vergelijkingen zoals ze op de 1^{e} tijdstap binnen het totale netwerk gelden en zo dicht mogelijk bij de geëxporteerde waardes liggen. Het resultaat van deze initialisatie slag is *timeseries_import-initialized.xml* en het is deze file die voor de echte optimalisatie run gebruikt wordt.

Voor de gebruikersinstellingen met betrekking tot de optimalisatie zijn meerdere ParameterConfig files betrokken. De attributen in deze files zijn afkomstig uit de Modifier schermen van de optimalisatie run

Model directory

Het Rijnland boezemnetwerk is gemodelleerd in Modelica. De Modelica hoofdklasse van het Rijnland netwerk is *Network.mo*.



Figuur 0.3 RTC-Tools 2 model directory(optimalisatie)

In de optimalisatie runs wordt een specifieke Modelica Optimalisatie klasse (*NetworkOptimization.mo*) opgezet, die eveneens werkt onder de Modelica hoofdklasse van het Rijnland netwerk (*Network.mo*).

Binnen de modelica files wordt gebruik gemaakt van de RTC-Tools (Deltares) Channel Flow libraries. De default libraries zijn beschikbaar binnen de RTC installatie. Hier wordt voor de simulatie gebruik van gemaakt. In de optimalisatie case worden de channel flow biblliotheek lokaal meegeleverd (zodat er geen afhankelijkheid van de versie is). Ook biedt dit de mogelijkheid om bepaalde formuleringen specifiek voor dit project aan te passen.

In het geval van Rijnland wordt er gebruik gemaakt van een *LocalHomotopicLinear.mo* en *LocalPartialHomotopic.mo* formulering voor de beschrijving van de kanalen (branches) fysica. Deze specifieke lokale formuleringsversies geven wat extra controlemogelijkheden voor het toepassen van de inertia term in de vergelijkingen die de wet van behoud van impuls beschrijven. Deze nieuwe formulering maakt het namelijk mogelijk om te 'switchen' tussen de volledige set van 1D Saint-Venant vergelijkingen en een diffusive wave benadering. Ondanks dat deze mogelijkheid toegevoegd is wordt hier niet actief gebruik van gemaakt in de huidige opzet en acteert deze als de standaard versie.

De *Network.mo* file wordt in het geval van de optimalisatie run in een pre-processing stap gegenereerd vanuit de *Network_orig.mo* file. In deze stap wordt er op basis van *rtcParameterConfig_Noodwaterkeringen.xml* een keuze gemaakt welke Modelica formulering er voor de kanalen (branches) gebruikt moet worden.

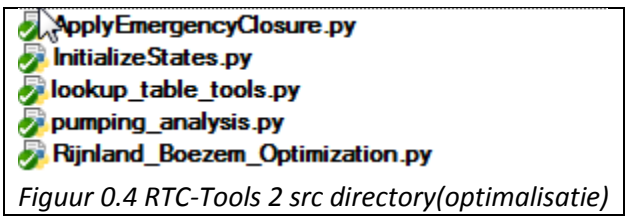
LocalHomotopicLinearClosedBranch.mo en *LocalPartialHomotopicClosedBranch.mo* worden gebruikt in het geval van de inzet van een Noodwaterkering. De afvoer en massatransport in het kanaal worden in deze formulering op 0 gezet. Effectief betekent dit dat er een blokkade gesimuleerd wordt in het kanaal, waarbij boven en benedenstroomse waterstanden en concentraties hydraulisch ontkoppeld zijn.

De *.fmu files zijn gecompileerde versies van de simulatie modellen. Deze files kunnen 'on the fly' tijdens de run uitgerekend worden, maar kunnen ook meegeleverd worden. Deze files zijn OS afhankelijk en voor een vergelijkbare situatie zal een Linux run een andere fmu file generen dan op een Windows systeem. Dit betekent dat het pre-compilen van een fmu welliswaar performance winst oplevert, maar een risico vormt als er op meerdere platformen gerund kan worden.

De Modelica files bevatten eigen annotatie die de opzet in meer detail duiden dan in dit document mogelijk is.

Source (src) directory

Een Python script dat draait in een speciale RTC-Tools omgeving (met toegang tot alle RTC-Tools methodes) bevat de aansturing van de model run.



Figuur 0.4 RTC-Tools 2 src directory(optimalisatie)

Voor de optimalisatie modus is een optimalisatie Python script opgezet (*Rijnland_Boezem_Optimization.py*), dat de doelstellingen en prioritering voor de optimalisatie run bevat. Dit is het 'hoofd script' dat aangeropen wordt. De overige python files (*.py) worden door het hoofd scrpt aangeroepen.

ApplyEmergencyClosure.py is verantwoordelijk voor het updaten/aanpassen van de Network.mo file in het geval van de inzet van een noodwaterkering (zoals beschreven in paragraaf 0).

InitializeStates.py is verantwoordelijk voor een speciale InitialStateEstimation run voor de eerste tijdstap, waarbij de states van het hele netwerk op de eerste tijdstap bepaald worden, op basis van de meetwaardes vanuit Delft-FEWS. Dit script schrijft de file *timeseries_import-initialized.xml* (zie paragraaf 0).

Lookup_table_tools.py is verantwoordelijk voor het inlezen van polder eigenschappen (uit */input/lookup_tables/*.csv*).

Pumping_analysis.py bevat de functies waarmee de impact (Solution Path) tijdseries bepaald worden.

Output directory

Als het model succesvol in simulatie modus gedraaid heeft dan wordt de output weggeschreven in een xml file (*timseries_export.xml*) die vervolgens door Delft-FEWS geïmporteerd kan worden en waarvan de tijdreeksen (evt.) in post-processing bewerkt kunnen worden. Delft-FEWS wordt gebruikt om de tijdreeksen te analyseren.

De volgende tijdreeksen worden in de optimalisatie modus weggeschreven en naar FEWS geïmporteerd:

- Waterstanden en Afvoeren van alle storage nodes
- Waterstanden en Afvoeren van alle bovenstroomse en benedenstroomse punten van de branches
- Representatief Boezem Peil (RBP)
- Debiet Boezemgemalen
- Pompstanden

Figuur 0.5 RTC-Tools 2 output directory(optimalisatie)

timeseries_export.xml bevat de (continue) resultaten van de optimalisatie run. In een post processing simulatie run worden de continue boezemgemaal debieten omgevormd naar discrete debieten die passen bij de mogelijke pompstanden. Doordat de resulterende afvoeren en waterstanden door het hele systeem iets zullen veranderen in deze discretisatiestap, worden de resulterende tijdseries weggeschreven naar *timeseries_export_postprocessed.xml*

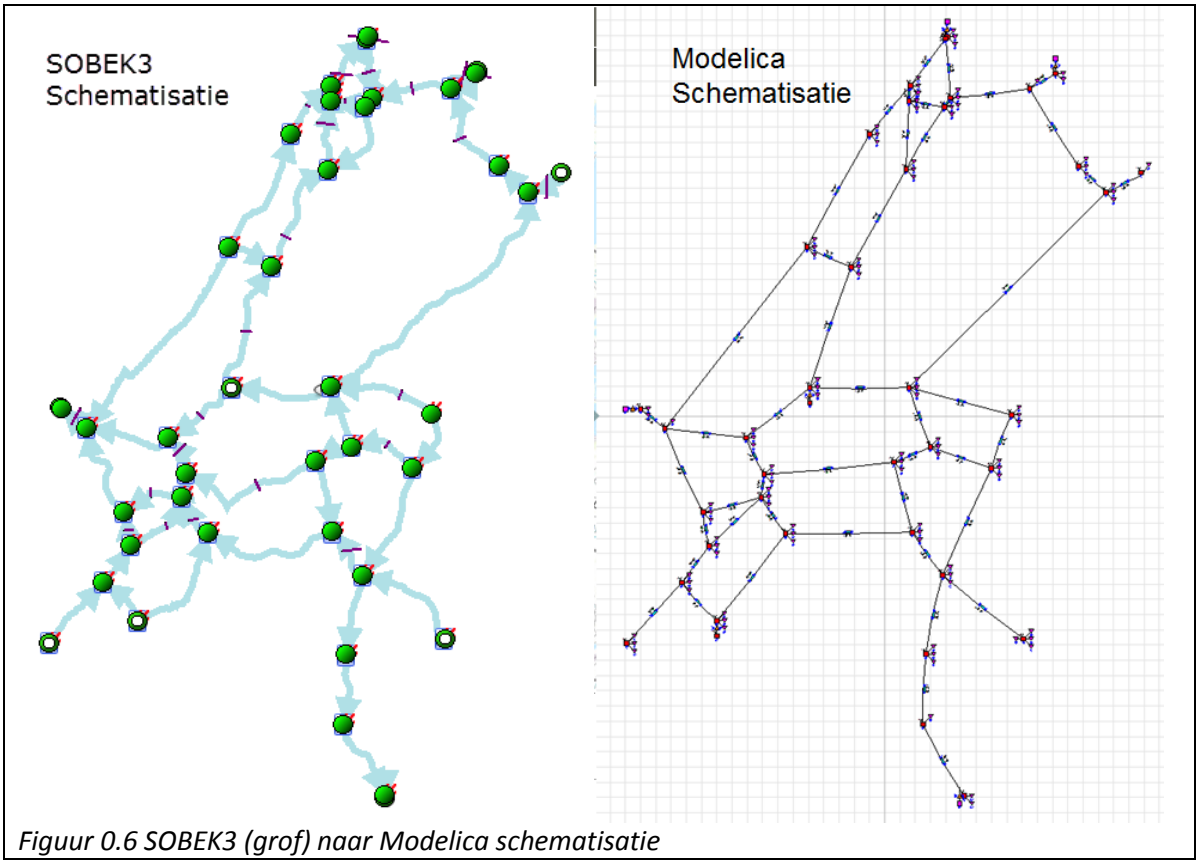
Verder wordt er informatie met betrekking tot de behaalde resultaten van de gedefinieerde doelen op een gestandaardiseerde manier samengevat in de file *sol_path_summary.csv*. Dit bestand kan

ingelezen en getoond worden door Delft-FEWS, waarbij het mogelijk is om de informatie op handige manieren te filteren en sorteren.

Pump_costs.csv bevat de informatie die gebruikte wordt voor het bepalen van de Impact waardes van de doelen op de afzonderlijke boezem gemalen.

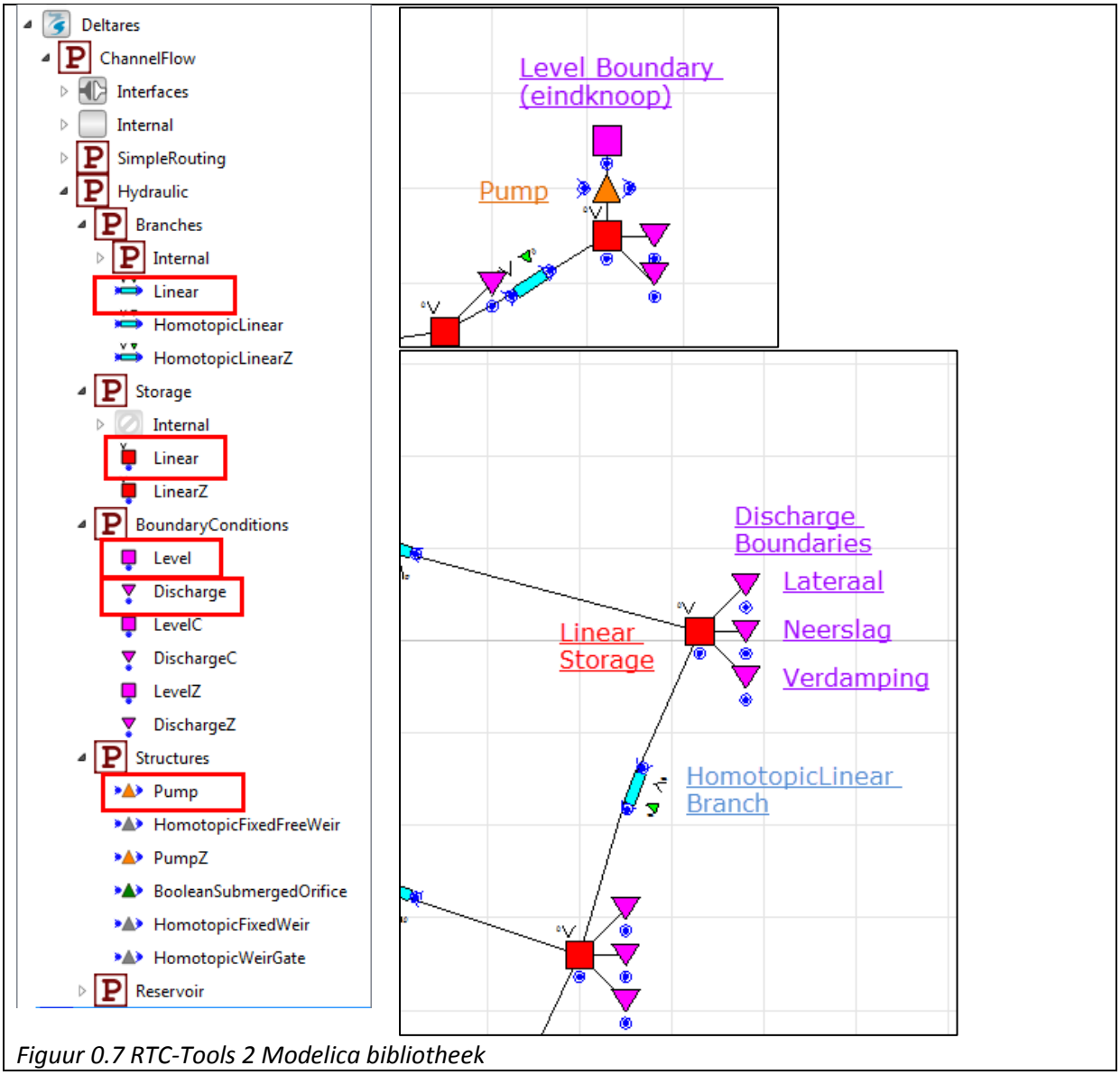
Modelica schematisatie

Het RTC-Tools 2 model van het Rijnland boezem systeem is gebouwd in Modelica en wordt aangestuurd vanuit een Python script. Dit model is gebaseerd op een grof SOBEK3 model van het Rijnland boezem netwerk, waarin diverse wijzigingen zijn doorgevoerd om het model geschikt te maken voor het doel van dit project. Een overzicht van het Modelica netwerk is weergeven in *Figuur 0.6*.



Figuur 0.6 SOBEK3 (grof) naar Modelica schematisatie

De structuur en belangrijkste keuzes met betrekking tot de modellering zullen in deze paragraaf besproken worden.



Figuur 0.7 RTC-Tools 2 Modelica bibliotheek

Het Rijnland netwerk in Modelica is opgebouwd uit 5 type elementen die beschikbaar zijn in de RTC-Tools model bibliotheek. Deze zijn weergegeven in *Figuur 0.7*.

- Linear Storage = oppervlakte water
- HomotopicLinear branches = boezemkanalen
- Level Boundary = buitenwaterstanden
- Discharge boundary = debiet forcings.
- Pumps = boezemgemalen

in *Figuur 0.7* is ook een detail weergave van een deel van de schematisatie van het Rijnland netwerk weergegeven. Aan de hand hiervan zullen enkele model conventies en aannames besproken worden.

Linear Storage

De *linear storage* elementen dienen als de model knopen van het netwerk. Als model eigenschap hebben ze een oppervlakte. De oppervlakte per storage knoop is bepaald door een watervlak

shapefile van het boezemsysteem te doorsnijden met een gebieds shapefile van het Rijnland Boezemnetwerk. De totale oppervlakte van deze storages bedraagt ruim 3600 hectare.

NAAM	Opp (ha)
BG_Gouda	31.70
BG_Halfweg	53.55
BG_Katwijk	103.00
BG_Spaarndam	135.15
ML_Alphen	94.65
ML_Amsterdam_NM	1.55
ML_Boskoop	21.70
ML_Heemstede_A	45.60
ML_Heemstede_C	70.50
ML_Leiden	42.05
ML_Leidschendam	16.65
ML_Lisse	125.40
ML_Nieuwe_Wetering	889.95
ML_Sloten	17.70
ML_Tolhuissluis	6.55
SN_Bodegraven	14.25

NAAM	Opp (ha)
SN_01	175.75
SN_02	44.20
SN_03	23.70
SN_04	53.25
SN_05	28.25
SN_06	148.95
SN_07	54.65
SN_08	186.00
SN_09	153.60
SN_10	42.15
SN_11	261.75
SN_12	48.40
SN_13	35.20
SN_14	18.30
SN_15	38.05
SN_16	331.50
SN_17	17.00
SN_18	212.10
SN_19	2.30
SN_20	29.50
SN_21	5.85
SN_22	56.95

Totale Oppervlakte:	3637.35 ha

Figuur 0.8 Linear Storage oppervlaktes van het Rijnland model

Een voorbeeld van alle parameters die meegegeven worden aan linear storage in de modelica file zijn:

- *HQ.C(each nominal = 1)*
- *C_nominal = fill(0.001, BG_Gouda.HQ.medium.n_substances), redeclare package medium = Medium*

De nominal voor de C (concentratie) wordt gebruikt om het optimalisatie probleem te schalen.

- *A = 317000.0*

Oppervlakte van de linear storage (m2)

- *H_b = -2.5883*

Bodempeil (m NAP)

- *HQ.H(min = -1.5, max = 0.0, nominal = 1.0)*

Minimale en maximaal toegestane waterstanden (m NAP). Dit is een harde randvoorwaarde en staat los van eventuele grenzen op waterstanden die als 'operationele grens' via de optimalisatie zelf opgelegd worden.

- *V(nominal = 317000.0)*

De nominal voor de V (volume) wordt gebruikt om het optimalisatie probleem te schalen.

- theta = theta_fudge

Theta is een term die gebruikt wordt in de homotopie benadering

HomotopicLinear Branches

HomotopicLinear branches worden gebruikt om de kanalen van het Rijnland netwerk te schematiseren. De branches zijn gemodelleerd met een aantal eigenschappen die voor de hydraulische simulatie gebruikt worden. De kanalen worden hydraulisch doorgerekend en de invloed van wind stress is meegenomen.

Branch eigenschappen

- length = *7304.4913*

De lengte van de kanalen zijn overgenomen uit de SOBEK3 definities van het grove model.

- width_up = *29.0701*, width_down = *29.0701*

Een branch in Modelica heeft in tegenstelling tot in de SOBEK3 schematisatie een rechthoekig profiel. In SOBEK3 is er een trapezoïde profiel gedefinieerd. De breedte van het watervlak in het trapezoïde profiel bij streefpeil wordt als breedte van het rechthoekige profiel in de Modelica schematisatie genomen.
Zolang de waterstanden niet te veel variëren is er beperkte afwijking van het stroomvoerend oppervlak tussen het trapezoïde en rechthoekige kanaal profiel.

- H_b_up = *-3.1178*, H_b_down = *-3.1178*, uniform_nominal_depth = *2.4928*

De ligging van de bodem wordt bepaald door in het SOBEK3 model het stroomvoerend oppervlak in het trapezoïde profiel bij streefpeil te bepalen. Met behulp van de breedte van het rechthoekige profiel kan nu de bodemligging en standaard diepte (bij streefpeil) bepaald worden, door aan te nemen dat het stroomvoerend oppervlak van het rechthoekige en trapezoïde profiel bij streefpeil gelijk moeten zijn.

- friction_coefficient = *38.0707*

De frictie coefficient die in RTC-Tools gebruikt wordt is Chezy. De waarde is bepaald uit de frictie term van de kanalen zoals deze in het SOBEK3 model gedefinieerd zijn

- rotation_deg = *-114.6447*

Deze term betreft de draaiing/orientatie van de branch. Dit is van belang om de wind opzet in het systeem te berekenen.

- n_level_nodes = 2

Met n_level_nodes wordt weergegeven hoeveel waterstandspunten er binnen de branch gebruikt worden in de hydraulische berekening.

De lengte en breedte voor de boezemkanalen in de RTC-Tools Modelica schematisatie zijn weergegeven in *Figuur 0.9*. De totale oppervlakte van de kanalen bedraagt ongeveer 1000 ha. De totale oppervlakte van de storage nodes bedraagt ongeveer 3650 hectare. Dit maakt de totale bergende oppervlakte van het Rijnland systeem (branches + storage nodes) rond de 4650 hectare.

NAAM	Lengte (m)	Breedte (m)	Opp (ha)
Aarkanaal_1	7304	29.1	21.23
Bennebroekervaart_1	3300	11.4	3.77
Braassemermeer_1	4335	25.2	10.93
Braassemermeer_2	2881	25.2	7.26
Buiten_Liede_1	4629	37.2	17.21
Does_1	9567	42.6	40.76
Drecht_1	3881	28.0	10.88
Drecht_2	6674	32.7	21.85
Fuikvaart_1	2311	11.2	2.58
Gouwe_2	5398	47.0	25.39
Gouwe_3	4343	47.0	20.43
Gouwe_4	4941	33.6	16.58
Halfweg_2	2173	146.1	31.74
Leidsevaart_1	14937	19.7	29.48
Leidsevaart_2	8071	17.5	14.14
Leidsevaart_3	4290	16.9	7.26
Nieuwe__Rijn_1	4143	40.1	16.61
Nieuwe_Meer_1	2927	148.1	43.34
Nieuwe_Vaart_1	4000	37.1	14.82
Oegstgeesterkanaal_2	1909	73.0	13.94
Oegstgeesterkanaal_3	5359	39.1	20.94
Oude_Rijn_1	6972	39.1	27.27
Oude_Rijn_2	3885	70.0	27.18

Totale oppervlakte:	1002 ha

NAAM	Lengte (m)	Breedte (m)	Opp (ha)
Oude_Rijn_3	8710	45.0	39.15
Oude_Rijn_4	2915	33.0	9.63
Oude_Wetering_1	3849	39.0	15.01
Rijn_1	6421	41.1	26.37
Rijn_Schiet_1	4642	36.1	16.77
Rijn_schiet_2	1418	36.9	5.24
Rijn_Schiet_3	2966	59.6	17.68
Ringvaart_1	6376	39.0	24.87
Ringvaart_2	17899	50.3	90.07
Ringvaart_3	2413	43.1	10.41
Ringvaart_4	6176	43.3	26.74
Ringvaart_5	5041	39.7	20.01
Ringvaart_6	693	36.7	2.55
Ringvaart_7	5258	37.2	19.55
Ringvaart_8	7301	37.3	27.22
Ringvaart_9	8542	36.1	30.85
Spaarne_2	144	40.0	0.58
Spaarne_3	3866	77.0	29.75
Spaarne_4	1362	31.1	4.24
Spaarne_5	4620	67.7	31.29
Vliet_1	5009	37.1	18.56
Vliet_2	2818	33.1	9.31
Vliet_3	2062	35.0	7.22
Warmonderleede_1	5459	45.6	24.90
Weipoortse_Vliet_1	8197	36.9	30.28
Woudwetering_1	4417	43.0	19.00

Figuur 0.9 HomotopicLinear Branch details van het Rijnland model

Level boundary
Level boundaries worden gebruikt om de (gemeten/voorspelde) waterstanden in de eindknopen (buiten het boezemnetwerk, aan de andere kant van de boezemgemalen) te modeleren. Het vastzetten van waterstanden met extern gedefinieerde tijdreeksen maakt het mogelijk (niet geïmplementeerd) om restricties aan de inzet van gemalen op te leggen in optimalisatie, bijvoorbeeld omdat de opvoerhoogte te groot zou worden.

Discharge boundary
Discharge boundaries worden gebruikt om afvoeren het systeem in (positief) of het systeem uit (negatief) te modeleren. De discharge boundaries worden in het Rijnland model aan de Linear Storage Nodes gekoppeld. In *Figuur 0.7* is te zien dat er voor de meeste storage nodes drie afzonderlijke debiet fluxes gedefinieerd zijn:

- Laterale debieten
- Neerslag
- Verdamping

Deze tijdreeksen worden vanuit het FEWS systeem aangeleverd.

Verder zijn er nog een aantal losse discharge boundaries op specifieke plaatsen in het netwerk. Het gaat hierbij o.a. om de inlaten bij Bodegraven en Gouda.

Pump
De Pump elementen worden ingezet om de boezemgemalen te modelleren. De pompen hebben een pomprichting. Binnen het Rijnlandmodel is een negatief debiet naar buiten gericht (het systeem uit).

De pomp debieten worden zowel in de simulatie als in de optimalisatie bepaald door RTC-Tools. De maximale waarde die het debiet aan kan nemen wordt gelimiteerd door de instelling zoals deze door de operator via het Modifiers Display is meegegeven.