Gebruikershandleiding
Calibriv-in-OpenDA

Rijkswaterstaat
Ministerie van Infrastructuur en Waterstaat

Gebruikershandleiding Calibriv-in-OpenDA

Version number : versie 1.1, 5 mei 2011
Maintenance : see www.helpdeskwater.nl/waqua

Log-sheet

document version	date	Changes with respect to the previous version
1.0	28-06-2010	Aanpassen M09.039 voor release 2010
1.1	05-04-2011	m3554 Memo in Simona style omgezet

Inhoudsopgave

Hoofdstuk 1

Inleiding

Het programma Calibriv-in-OpenDA is bedoeld voor het calibreren van Waqua/Triwaq riviermodellen, waarbij vooral de bodemruwheid in het zomerbed wordt aangepast om het model beter te laten overeenstemmen met metingen. De methode is overgenomen van het voormalige SIMONA-programma Calibriv. Deze is echter aangepast en opnieuw geïmplementeerd via het open-source systeem OpenDA. Hiervoor is een koppeling gemaakt tussen Waqua/Triwaq en OpenDA.

In paragraaf 2 wordt een globale introductie van "calibratie" gegeven en wordt globaal uitgelegd wat OpenDA is en kan. Vervolgens wordt in paragraaf 3 de opzet en werking van Calibriv-in-OpenDA uitgelegd. Paragraaf 4 gaat dieper in op het gebruik van Calibriv-in-OpenDA. We maken hierbij gebruik van een voorbeeld waarvoor de calibratie stap-voor-stap wordt uitgewerkt. De installatie van OpenDA wordt uitgelegd in paragraaf 4.6.

Hoofdstuk 2

Overzicht van calibratie met OpenDA

2.1 Calibratie

Fysische processen kunnen worden gesimuleerd met een *model*. Zo'n model zal in het algemeen de achterliggende wiskundige vergelijkingen die bij deze fysische processen horen (zo goed mogelijk) oplossen. Bij deze vergelijkingen horen, voor elke specifieke situatie die moet worden gesimuleerd, allerlei begin- en randvoorwaarden, en *parameters* die in de vergelijkingen moeten worden ingevuld maar kunnen afhangen van de specifieke situatie.

Vaak is al ongeveer bekend wat deze parameters zouden kunnen zijn. Deze schattingen worden dan ook in eerste instantie gebruikt in het model.

Het model levert bepaalde uitkomsten, die in het algemeen *predicties* worden genoemd. Uiteraard is gewenst dat deze predicties zo goed mogelijk overeenkomen met de werkelijkheid. Om dat te kunnen beoordelen, moeten deze predicties vergeleken kunnen worden met metingen (*observaties*) van de werkelijke situatie waar de simulatie naar gemodelleerd is.

Het vergelijken van observaties en predicties geeft aan hoe goed de parameters gekozen zijn. We kunnen vervolgens aan de hand van de discrepantie systematisch, procesmatig, de parameters dusdanig aanpassen dat de modelpredicties beter gaan passen bij de metingen. Dit noemen we *afregelen* of *calibreren* van een model.

We formuleren het wiskundig aldus. Er is een model M dat afhangt van een initiële toestand x, randvoorwaarden u, parameters p en de tijd t en dat predicties y_{pred} oplevert:

$$y_{pred} = M(x, u, p, t) \tag{2.1}$$

Hierin is p een vector met $npar$ elementen: $p \in \mathbb{R}^{npar}$. Deze parameters zijn tot op zekere hoogte onzeker. Er is een zekere, onbekende "echte" waarde p_{true} en we hebben een schatting van de onzekerheid in de gegeven waardes p:

$$p = p_{true} + \epsilon_{par}, \quad \epsilon \sim N(0, \sigma) \tag{2.2}$$

Hierin staat ϵ voor de onzekerheid van p en is $N(\mu, \sigma)$ de standaardnormaalverdeling. In principe kan er ook met andere kansverdelingen worden gewerkt. Vanwege de onzekerheid van p wordt het

model (2.1) *stochastisch* genoemd. Dit in tegenstelling tot een deterministisch model waarin geen onzekerheden voorkomen.

Daarnaast zijn er metingen y_{obs} met onzekerheden ϵ_{obs}. Het aantal metingen wordt aangegeven met $nobs$. Gevraagd wordt nu om die instellingen voor de parameters p te bepalen waarvoor de overeenstemming tussen model en metingen wordt gemaximaliseerd. Dit gebeurt met een *doelfunctie* of *kostenfunctie* J:

$$\min_{p \in \mathbb{R}^{npar}} J(y_{obs}, \epsilon_{obs}, y_{pred}(p), p, p_{ref}, \epsilon_{par}) \tag{2.3}$$

Het afregelproces bestaat uit het systematisch zoeken naar waarden voor p waarvoor de rekenresultaten y_{pred} zo goed mogelijk lijken op de metingen y_{obs}. Dit wordt typisch wiskundig uitgewerkt via de som van de kwadraten van de verschillen $y_{obs}(i) - y_{pred}(i)$. Verder moeten de parameters vaak aan aanvullende voorwaarden voldoen: realistisch patroon, of kleine aanpassingen ten opzichte van de initiële waarden die al goed waren ingevuld. Dit wordt in de kostenfunctie verwerkt via een *penalty*-term waarin referentieparameters p_{ref} worden gebruikt. De onzekerheden ϵ_{obs} en ϵ_{par} worden gebruikt voor de weging van de verschillende bijdragen aan J.

Voor het "systematisch zoeken" kunnen verschillende *calibratiealgoritmen* worden gebruikt. Er zijn veel keuzes voor dergelijke algoritmes mogelijk. De overeenkomst tussen al deze algoritmen is:

- Er wordt een modelsimulatie met een specifieke keuze van parameters uitgevoerd.
- De predicties van de simulatie worden vergeleken met de metingen, de kostenfunctie wordt geëvalueerd.
- Aan de hand van de verschillen tussen predicties en metingen wordt een nieuwe keuze van parameters geconstrueerd.
- De simulatie wordt met deze nieuwe parameterkeuze herhaald.
- Zodra de waarde van de kostenfunctie voldoende klein is stopt dit herhalingsproces en kunnen we zeggen dat we het model hebben gecalibreerd, en wel met de laatste keuze van de parameters.

Voor een gebruiker die een model wil calibreren zijn er in principe twee mogelijkheden om dit te bereiken:

1. De maker of gebruiker van een model schrijft zelf een programma om calibratie met dit model uit te voeren. Het model en het calibratiealgoritme worden *nauw verweven* met elkaar. Het voordeel is dat alle details van het model al bekend zijn bij de gebruiker en kunnen worden benut in het calibratieproces. Bijvoorbeeld hoe de te calibreren parameters in het model zitten en predicties worden opgehaald en vergeleken met observaties. Het nadeel van deze aanpak is dat het gemaakte calibratiealgoritme niet herbruikbaar is voor andere modellen, misschien niet het meest geschikte algoritme is, en nog fouten kan bevatten.
2. De tweede aanpak is het *aansluiten* van het model *op bestaande calibratieprogrammatuur*. Aan deze programmatuur moet dan wel uitgelegd worden wat de parameters en predicties van het model zijn, en hoe ze verkregen of gemanipuleerd kunnen worden. De koppeling tussen

het model en de metingen moet formeel en expliciet worden gemaakt. Dat is vergeleken met de eerste aanpak een nadeel. Het voordeel van deze aanpak is echter dat we dan meteen wel een geschikt, en beproefd, calibratiealgoritme "van de plank" kunnen pakken. Bovendien kan er dan eenvoudig met verschillende algoritmen worden geëxperimenteerd.

OpenDA is een voorbeeld van bestaande calibratieprogrammatuur.

2.2 OpenDA

OpenDA kan het beste worden gezien als een gereedschapskist voor het uitvoeren van data-assimilatie en calibratie van modellen. Een gebruiker levert aan OpenDA een (beschrijving van een) *model* en een verzameling *metingen*. OpenDA beschikt zelf over allerlei *methoden*, ofwel *algoritmen* om een calibratie of data-assimilatie uit te voeren. OpenDA hoeft niet zo veel van het model af te weten. Het moet de mogelijkheid hebben om bepaalde dingen met een model te doen, zoals:

- Stop een bepaalde keuze van parameters in het model in plaats van de oorspronkelijke verzameling parameters;
- Voer een simulatie uit met het model;
- Vraag aan het model wat de predicties zijn (nadat het model is uitgevoerd);
- Vraag de bij deze predicties horende observaties op.

OpenDA ziet het model derhalve als een "zwarte doos". De beperkte lijst acties die OpenDA met het model moet uitvoeren wordt voorgeschreven door OpenDA's *modelinterface*. Die interface wordt typisch ingevuld door het schrijven van een *wrapper* rondom een bestaand model. In de volgende paragraaf wordt nader ingegaan op de behandeling van het model door OpenDA.

2.3 OpenDA's "Black Box Stochastic Model Factory"

In Calibriv-in-OpenDA wordt OpenDA's "Black Box Stochastic Model Factory" gebruikt. Het gaat hier om een zeer lange naam met een groot aantal mogelijk onbekende woorden. Hieronder volgt een poging om de naam te verklaren.

Model Factory: OpenDA vindt dat elk gebruik van andere parameters een ander "model" oplevert. De basisschematisatie (simulatie-invoerfile) wordt daarom een "modellenfabriek" genoemd. Dit wordt een model zodra alle onderdelen van de siminp-file (m.n. ruwheidsbeschrijving) precies zijn ingevuld. Een model wordt ook wel instantiatie van de modellenfabriek genoemd. Een calibratiealgoritme maakt steeds nieuwe modelinstantiaties aan.

Stochastic Model: Een "stochastisch model" is een model dat stochastische elementen bevat, met name de te calibreren parameters. In OpenDA bestaat een stochastisch model uit het onderliggende deterministische model (zonder stochastische elementen), de lijst van parameters,

de onzekerheden van de parameters, en de lijst van predicties die in de calibratie worden gebruikt.

De onzekerheden in de parameters worden alleen gebruikt in de zogenaamde penalty-term in de doelfunctie (2.3). Wanneer het niet uitmaakt hoeveel de parameters worden aangepast (wat overeenkomt met zeer grote onzekerheden), dan kan dat te kennen worden gegeven aan het algoritme, zie paragraaf 4.2.

Black Box Model: De term "black box" geeft aan dat de *koppeling* tussen de generieke programmatuur van OpenDA en het concrete model is gerealiseerd *via files*. Hiervoor hoeft de modelprogrammatuur zelf niet te worden aangepast. Het alternatief is dat er een wrapper wordt geprogrammeerd waarin OpenDA het model aanstuurt via functie/subroutine-calls.

OpenDA bevat een generieke "black box model wrapper". Dit is een stuk programmatuur waarin via een invoerfile de aansturing van een concreet model kan worden ingevuld. Deze invoer bestaat uit een lijstje van commando's voor interne of externe programmatuur voor

- het omzetten van parameterwaarden in modelinvoer;
- het opstarten van simulaties;
- het ophalen van simulatieresultaten en omzetten in predicties.

Deze black box model wrapper is gebruikt voor het realiseren van Calibriv-in-OpenDA.

2.4 Aansturing en gebruik van OpenDA

OpenDA is geschreven in Java en heeft dan ook een sterk object-georienteerd karakter. In principe hoeft de gebruiker niets aan de source-code van OpenDA te veranderen. Er is reeds een hoofdapplicatie aanwezig die alleen hoeft te worden opgestart met een bestandsnaam als argument (zie hiervoor ook paragraaf 4.6). Dit door de gebruiker te maken hoofdbestand vertelt OpenDA:

- welke *methode* (algoritme) voor de calibratie gebruikt moet worden,
- welk *model* moet worden gecalibreerd, en
- welke *metingen* er moeten worden gebruikt.

De metingen staan beschreven in een object dat *(stochastic) observer* is genaamd.

Het hoofdbestand en de invoerfiles behorende bij deze drie onderdelen zijn allemaal in XML-formaat. Voor elke XML-file is vastgelegd wat er allemaal gespecificeerd mag en moet worden. Bevatten de XML-files geen fouten, dan treedt middels een druk op de knop het calibratiealgoritme in werking. Uiteindelijk wordt dan het eindresultaat afgedrukt: een lijst van getunede parameters, en hoe goed de modelpredicties met deze parameters aansluiten bij de observaties.

Er is dus een hierarchie van XML-bestanden. Deze hierarchie wordt ook weerspiegeld in de directory-structuur die een gebruiker voor zijn calibratieprobleem moet opzetten.

De taak van de gebruiker is derhalve:

- Een directorystructuur in te richten met invoerfiles van het model en files waarin de metingen staan;
- De XML-files te beschrijven die precies aan OpenDA vertellen hoe en wat er moet worden gecalibreerd.

Hoe dit precies moet wordt allemaal in de volgende paragrafen uitgelegd.

Hoofdstuk 3

Gedetailleerd overzicht van Calibriv-in-OpenDA

Calibriv-in-OpenDA is een calibratieprogramma voor Waqua/Triwaq riviermodellen dat is gerealiseerd via een koppeling tussen Waqua/Triwaq en OpenDA.

3.1 Afregelen van riviermodellen middels calibratie

Voor het simuleren van rivierstromingen binnen Waqua/Triwaq moet de simulatie-invoerfile, samen met bijbehorende afgeleide files, zo volledig mogelijk ingevuld worden. De meeste benodigde informatie voor de invoer (zoals de fysieke afmetingen van de rivier) is min of meer bekend. Het is veel moeilijker om a priori een goed idee te hebben van de bodemruwheid. Dit is ook een reden om deze bodemruwheid dan maar meteen als *stelparameter* te gebruiken: het is een vergaarbak van alle onzekerheid waarmee we het hele model pogen te fine-tunen.

De modellering van de bodemruwheid valt in de `siminp`-invoerfile onder het sub-hoofdstuk FRICTION binnen FLOW. Afhankelijk van de keuze van modellering komt het er uiteindelijk vaak op neer dat van bepaalde fysische parameters de waarde zo goed mogelijk moet worden vastgesteld.

Calibriv-in-OpenDA is gericht op riviermodellen waarin de ROUGHCOMBINATION methode met White-Colebrook formulering wordt gebruikt. Hierbij worden zogeheten “ruwheidscodes” gebruikt, met bijbehorende parameters a tot en met d. De bepaling van deze parameters gebeurt door calibratie. De predicties zijn in dit geval de waterstanden in checkpoints. De checkpoints komen overeen met de locaties waar de waterstand daadwerkelijk is gemeten.

Belangrijke aspecten hierbij zijn:

- Hoe goed (nauwkeurig/betrouwbaar) zijn de metingen?
- Hoe goed is de beginkeuze van parameters? Hoe zeker zijn we daarvan?
- Kunnen de metingen en de predicties goed met elkaar worden vergeleken (zijn ze op dezelfde tijd en plaats gedefinieerd)?

- Zijn de metingen relevant? Het variëren van de bodemruwheid in een bepaald riviertraject heeft in het algemeen weinig effect op observaties van waterstandsstations die stroomafwaarts gelegen zijn.

3.2 Conceptuele beschrijving van het calibratieproces

In een calibratierun worden metingen in de vorm van tijdreeksen gebruikt op verschillende lokaties in de rivier. De rivier wordt opgedeeld in trajecten met constante ruwheidsparameters (a-d) waarbij aan elk traject een of meer meetlokaties gekoppeld worden. Hierbij wordt gebruik gemaakt van de aanname dat de ruwheid met name invloed heeft op de waterstand van het direct aangelegen bovenstroomse gebied, hoe verder weg hoe kleiner de invloed wordt.

De eenvoudigste aanname is dat bij elke meetlokatie precies één traject met één te calibreren parameter hoort. Deze aanname van de beperkte ruwheidsinvloed is wellicht te strict. Zo is het gewenst om een meetlokatie te koppelen aan een aantal parameters in plaats van een enkele te calibreren parameter.

Het huidige concept vor Calibriv-in-OpenDA houdt rekening met bovenstaande overwegingen en is als volgt:

- Er wordt aangegeven in welke trajecten de parameters a tot en met d (of een deel daarvan) moeten worden gecalibreerd. Dit gebeurt door per vak/traject/sectie een eigen ROUGHCODE te gebruiken;
- De predicties worden middels checkpoints aangegeven en de lokaties moeten overeenkomen met lokaties van metingen;
- Er zijn a priori geen aannames over afhankelijkheden. Wel kan precies worden voorgeschreven welke afhankelijkheden er *wel* bestaan. Per predictie/meting wordt een lijst van ruwheidsparameters gevraagd die volgens de gebruiker van invloed zijn.
- Het calibratiealgoritme, "SparseDud" genaamd, houdt rekening met de afhankelijkheden. Hoe minder afhankelijkheden er zijn, en hoe beter die zijn aangegeven, des te minder simulaties er in het calibratieproces nodig zijn.

Voor de ervaren gebruiker geven we hierbij de volgende historische toelichting. De oude implementatie van Calibriv in SIMONA is gebaseerd op de eerste aanpak van paragraaf 2.1, en was specifiek gemaakt/nauw verweven met de Waqua-programmatuur. Vanwege de al langer levende wens om alles rondom data-assimilatie en calibratie in SIMONA onder te brengen in OpenDA, vanwege de uitbreidingen van Waqua met waterstands/afvoerafhankelijke ruwheden, en vanwege de hiervoor benodigde aanpassingen aan Calibriv, is Calibriv opnieuw gerealiseerd via OpenDA. Dat is dus conform de tweede aanpak van paragraaf 2.1, op basis van bestaande calibratieprogrammatuur.

In OpenDA bestonden al algemene calibratiemethodes zoals "DuD". De DuD-methode bleek voor ons doel (opnemen van Calibriv in OpenDA) weliswaar voldoende en altijd correct werkend maar traag. Dat komt omdat er in DuD op voorhand helemaal niet van wordt uitgegaan dat bepaalde afhankelijkheden tussen observaties en parameters verwaarloosd kunnen worden. Daarom is

er in OpenDA een afgeleide calibratiemethode genaamd "SparseDuD" geïntroduceerd. Deze methode combineert de specifieke gedachten van de methode van Calibriv, rekening houden met (on)afhankelijkheden tussen metingen en calibratieparameters, met de generieke werking van het DuD-algoritme. Door de uitbreidingen ten opzichte van de oorspronkelijke methode van Calibriv kunnen ook waterstandsafhankelijke ruwheden worden gecalibreerd.

3.3 Inrichten van directories

In deze en de volgende paragraaf wordt uitputtend doch globaal uitgelegd hoe de gebruiker een calibratie kan uitvoeren met Calibriv-in-OpenDA. De details worden verder uitgelegd in paragraaf 4.

In de uitleg maken we gebruik van een voorbeeld, een Waqua-model met de naam `rough_stat9`. Er is ook een tweede model `rough_fase2`. In dat model wordt gebruik gemaakt van de waterstandsafhankelijke ruwheden in Waqua. Daarbij kunnen er per traject (ruwheidscode) meerdere ruwheidsparameters $a - d$ worden gebruikt, die ieder gelden binnen een bepaalde range voor een stuurparameter. In de uitleg zal vermeld worden waar de behandling van dit tweede model afwijkt van het eerste model.

Er wordt verondersteld dat Calibriv-in-OpenDA al is geïnstalleerd; zie hiervoor paragraaf 4.6.

Onder `trunk/openda` bevinden zich directories als `src`, `bin`, `doc` en `tests`. Een aangeraden aanpak is de volgende:

Maak op een willekeurige plek een directory met de naam van het Waqua-model, hierna te noemen `<modelnaam>`. In ons voorbeeld staat de directory binnen `tests`.

Daarin moet de volgende directorystructuur met vaste namen worden ingericht:

```
* stochmodel/
   * detmodel/
      * base/
* stochobserver/
* algorithm/
```

In `detmodel/base` (waarbij `detmodel` staat voor "deterministisch model") wordt het oorspronkelijke Waqua-model gezet, dus met de initiële keuze van de ruwheidsparameters. Hierin moeten de Waqua-invoerbestanden staan, dus de `siminp`-file en bijbehorende include-bestanden. Merk op dat het model in zekere zin niet deterministisch is, maar eigenlijk de beginschatting is van het stochastische model: het hele idee van calibratie is juist dat we hebben aangegeven dat het model niet deterministisch is.

Later zorgt OpenDA ervoor dat in `detmodel` naast de `base` directory allerlei werkdirectories komen te staan; per calibratie-run één werkdirectory. Elke werkdirectory bevat een aangepaste kopie van `detmodel`. De aanpassing bestaat eruit dat in de invoerfiles in deze directories andere keuzes voor de parameters staan. Het complete model wordt dus voor elke evaluatie gekopiëerd. Dat is weliswaar duidelijk, maar niet zo efficiënt; later wordt dit wellicht beter aangepakt. (*TODO*).

In de directory `stochobserver` worden de metingen gezet. De directory heet zo ("stochastisch") omdat behalve de metingen ook hun onzekerheden worden weergegeven.

Wellicht staan de metingen waarover de gebruiker beschikt nog niet in het door OpenDA gewenste formaat. Hier komen we later, in paragraaf 4.4 op terug.

3.4 De XML-bestanden

Nu het Waqua-model is ingericht, moet aan OpenDA worden verteld wat er moet gebeuren. Dat gebeurt middels XML-files.

- Maak in de hoofddirectory `<modelnaam>` een hoofdbestand met een verwijzing naar een *model*, een *observer* en een *algoritme (methode)*.

  Kies een geschikte bestandsnaam. De naam geeft aan dat het gaat om een calibratie van een zeker model, mogelijk met een aanduiding van de variant of iets dergelijks.

  Een voorstel voor een naam in ons voorbeeld is `calib_waqua_rough_stat9.xml`. Het betreft hier klaarblijkelijk een calibratie van een Waqua-model dat `rough_stat9` (met wellicht negen checkpoints) heet.

  De specificatie van zo'n hoofdbestand staat in paragraaf 4.1.

- Maak een XML-beschrijving van het stochastisch model. Dit bestand komt in de `stochmodel` directory.

  In het voorbeeld heet dit `stochmodel_rough_stat9.xml`.

  Voor meer details verwijzen we naar paragraaf 4.2.

- Maak een XML-beschrijving van het deterministisch model. Dit bestand komt in de `detmodel` directory.

  In het voorbeeld heet dat `detmodel_rough_stat9.xml`.

  Het is een goed idee dat deze naam erg lijkt op die van het stochastische model.

  Bij het deterministisch model wordt de configuratie gespecificeerd: invoerfiles, werkdirectories, simulatienamen. Tevens wordt alvast aangegeven welke verbindingen van het model met de buitenwereld interessant zijn. Dit worden in OpenDA "*Exchange Items*" genoemd.

  In deze directory hoort ook de `WaquaWrapper.xml` te staan. Dit is geen invoerfile voor het calibratieprobleem maar kan worden beschouwd als een (niet veranderend) stuk van de programmatuur van Calibriv-in-OpenDA.

  In paragraaf 4.3 wordt hier nader op ingegaan.

- Maak een XML-beschrijving van de stochastic observer. Dit bestand komt in de directory `stochobserver`.

  Een goede bestandsnaam geeft aan dat het gaat om metingen in een bepaalde rivier, en in welke tijdsperiode. In ons voorbeeld heet het bestand `obs_rough_20000707.xml`.

  De waarnemingen zelf komen als tijdreeksen in afzonderlijke Noos-bestanden per station.

Meer uitleg over de observer staat in paragraaf 4.4.

- Kies een algoritme en maak daarvoor een invoer-file. Binnen Calibriv-in-OpenDA is dit typisch de SparseDud-methode, maar er kan ook met andere algoritmen worden geëxperimenteerd.

  De namen van de invoerfiles verwijzen naar het soort algoritme met eventueel de varianten, zoals `DudAlgorithm.xml` en `sparseDudAlgorithm.xml`.

  In paragraaf 4.5 wordt de praktische kant van het algoritme (de XML-invoerfile) uitgebreid besproken. Voor de theorie verwijzen we naar de Appendices.

Zodra de verschillende invoerfiles zijn klaargezet, kan de calibratie worden uitgevoerd.

Hoofdstuk 4

Gebruik van Calibriv-in-OpenDA

In de vorige paragraaf hebben we laten zien wat Calibriv-in-OpenDA is en wat er allemaal klaargezet moet worden opdat Calibriv-in-OpenDA gebruikt kan worden.

In deze paragraaf wordt verteld wat er in de XML-invoerfiles moet staan. Omdat we steeds van onze voorbeeld-calibratie uitgaan, volstaat het in de meeste gevallen voor de gebruiker om deze invoerfiles (het zijn er vijf) over te nemen van het voorbeeld en aan te passen.

4.1 Overzicht van een calibratie met OpenDA

Voor een calibratie zijn de volgende drie onderdelen nodig:

1. een *model* om te calibreren;

   dit model moet *parameters* (invoer) bevatten die invloed hebben op de rekenresultaten, en *predicties* (uitvoer) genereren, die vergeleken kunnen worden met metingen.

   De manier waarop een Waqua-model wordt aangeboden (de *schematisatie*) aan Calibriv-in-OpenDA wordt beschreven in paragraaf 4.3. De typische OpenDA-terminologie die hier bij hoort is reeds uitgelegd in 2.3.

2. *metingen* die kunnen worden vergeleken met modelresultaten;

   Een manier waarop metingen kunnen worden aangeboden aan Calibriv-in-OpenDA wordt beschreven in paragraaf 4.4.

3. een *calibratiemethode*, die op een efficiënte wijze die waarden kan vinden voor de parameters, zodat de predicties van het model zo goed mogelijk overeen komen met de metingen.

   Enkele calibratiemethoden die beschikbaar zijn in Calibriv-in-OpenDA worden beschreven in paragraaf 4.5.

Een voorbeeld van een hoofdbestand met model, observer en algoritme is weergegeven in Tabel 4.1. In de volgende paragrafen wordt dit voorbeeld verder uitgewerkt.

```
<?xml ... ?>

<openDaApplication xmlns="http://www.openda.org" xmlns:xsi=
                   "http://www.w3.org/2001/XMLSchema-instance"
                   xsi:schemaLocation="http://www.openda.org
                   http://www.openda.org/schemas/openDaApplication.xsd">

  <stochObserver className=
             "org.openda.exchange.timesseries.NoosTimeSeriesStochObserve
     <workingDirectory>./stochobserver</workingDirectory>
     <configFile>obs_rough_20000707.xml</configFile>
  </stochObserver>

  <algorithm className="org.openda.algorithms.Dud">
     <workingDirectory>./algorithm</workingDirectory>
     <configString>sparsedudAlgorithm.xml</configString>
  </algorithm>

  <stochModelFactory className=
                          "org.openda.blackbox.wrapper.BBStochModelFactor
     <workingDirectory>./stochmodel</workingDirectory>
     <configFile>stochmodel_rough_stat9.xml</configFile>
  </stochModelFactory>

  <resultWriters>
     <resultWriter className="org.openda.resultwriters.MatlabResultWrite
        <workingDirectory>.</workingDirectory>
        <configFile>results_gewonedud.m</configFile>
     </resultWriter>
     <resultWriter className="org.openda.resultwriters.TextTableWriter">
        <workingDirectory>.</workingDirectory>
        <configFile>results_gewonedud.csv</configFile>
     </resultWriter>
  </resultWriters>

</openDaApplication>
```

Tabel 4.1: *XML-code voor de beschrijving van een calibratie in Calibriv-in-OpenDA. Verkorte weergave van het bestand* `calib_waqua_rough_stat9.xml`

Duidelijk te zien is dat de drie onderdelen (model, metingen, en methode) precies terugkomen in dit hoofdbestand. Verder is het ook mogelijk hier aan te geven welke uitvoer gewenst is (`resultWriters`); dit onderdeel is echter niet verplicht. Het is mogelijk om Matlab-uitvoer te geven of uitvoer in het zogeheten CSV-formaat. In dit laatste formaat worden de belangrijkste resultaten van de calibratie weergegeven in de vorm van een tabel.

De methode is in het algemeen onafhankelijk van het model en de metingen. Ze kan (in theorie) vrijelijk worden vervangen door een alternatief. Een voorbeeld waarbij dit niet helemaal geldt is als er gekozen wordt voor SparseDuD. Daarin wordt namelijk expliciet opgegeven welke afhankelijkheden er bestaan tussen observaties en calibratieparameters; zie hierover later meer (paragraaf 4.5).

De beschrijving van het model en van de metingen zijn in het algemeen wel sterk aan elkaar gerelateerd. Er moet namelijk altijd een associatie tussen de observaties en predicties worden gemaakt. Welke grootheden moeten, in welke eenheden, op welke plaats, in welke tijdsaanduiding worden vergeleken? Informatie betreffende deze vragen is te halen uit de XML-informatie van het model en de observer, en uit de meta-informatie in de files waar predictie en metingen zijn opgeslagen. Hier komen we later op terug, in paragraaf 4.4.

4.2 Het stochastisch model

Het eerste onderdeel voor calibratie, het *model*, wordt binnen OpenDA beschreven met de “Black Box Stochastic Model Factory”. Dit is uitgelegd in paragraaf 2.3.

Qua administratie bestaat het onderdeel “model” uit drie schillen, in de gedaante van drie XML-files. Zoals in paragraaf 3 al kort is uitgelegd, is de buitenste schil het stochastische model. Daarbinnen zit het deterministisch model. De binnenste schil is de “Waqua-wrapper”. We werken van buiten naar binnen en beginnen dus met het stochastische model.

Een voorbeeld is het bestand `stochmodel_rough_stat9.xml` (Tabel 4.2). Deze XML-beschrijving kan voor de gebruiker in eerste instantie wat onduidelijk zijn. Het is belangrijk het volgende voor ogen te houden:

- Het gaat in deze beschrijving niet om het complete stochastische model, maar meer om het stochastische gedeelte van het model volgens het principe “stochastisch model is deterministisch model plus onzekerheid”. Het deterministische model wordt behandeld in paragraaf 4.3.
- Vanwege de “black box”-benadering hoeven alleen de in- en uitgangen van het model aan OpenDA te worden gespecificeerd.
- De onzekerheid (penalty-term) wordt opgelegd via de calibratieparameters en is dus een ingang van het model.
- OpenDA moet weten wat het model oplevert, de predicties, dit is de uitgang van het model.

De gebruiker moet binnen de aangegeven XML-structuur op enkele plekken iets invullen. De XML-structuur is een “boom” en deze is strict voorgeschreven.

```
<?xml ... ?>
<blackBoxStochModel ...>

    <modelConfig>
        <file>./detmodel/detmodel_rough_stat9.xml</file>
    </modelConfig>

    <vectorSpecification>
        <parameters>
            <regularisationConstant>
                <stdDev value=".02"  transformation="identity"/>
                <vector id="A-411"/>
            </regularisationConstant>
            ...
            <regularisationConstant>
                <stdDev value=".02"  transformation="identity"/>
                <vector id="A-419"/>
            </regularisationConstant>
        </parameters>

        <predictor>
            <vector id="M230.waterlevel"/>
            ...
            <vector id="M440.waterlevel"/>
        </predictor>
    </vectorSpecification>

</blackBoxStochModel>
```

Tabel 4.2: *XML-code voor de beschrijving van een Waqua-schematisatie in Calibriv-in-OpenDA, gebaseerd op OpenDA's black box stochastic model factory. Verkorte weergave van het bestand* `stochmodel_rough_stat9.xml`

- Het stochastische model, gezien als boom, bestaat uit twee hoofdtakken: een verdere beschrijving van het deterministische (Blackbox-) model (zie hiervoor paragraaf 4.3) en een stochastische specificatie, welke aangeeft wat Calibriv-in-OpenDA met het model kan.

  De gebruiker moet in de hoofdtak `modelConfig` de lokatie en naam van de XML-invoerfile van het deterministische model invullen.

- De specificatie is onderverdeeld in een invoergedeelte `parameters` en een uitvoergedeelte `predictor`.

- Voor parameters staat het XML-schema alleen de `regularisationConstant` toe. We definieren hier de parameters die we willen variëren. (Dat hoeven dus niet *alle* parameters te zijn. Wel moet er minstens één parameter worden genoemd, anders kan er natuurlijk niet gecalibreerd worden.)

  Hierbij geven we de naam (`id`), de standaardafwijking met betrekking tot de onzekerheid in de invoerparameters `stdDev` en we geven aan dat we additioneel gaan variëren met parameters: zet daartoe `transformation` op `'identity'`. De standaarddeviatie is soms van belang voor het calibratiealgoritme (zie paragraaf 4.5). De `id` kan in deze fase nog vrij worden gekozen, wel is het natuurlijk handig dat de naam lijkt op de parameternaam in de Waqua-invoerfile.

- Het gedeelte `predictor` bestaat uit een lijst van grootheden die door het model uitgevoerd wordt. Deze verzameling van modeluitkomsten moet, als we gaan calibreren, nog wel gekoppeld worden aan metingen op dezelfde plaatsen en tijden. In ons geval betreft het (een tijdreeks van) de waterstand in een achttal punten. De `id` kan in deze fase nog vrij worden gekozen, wel is het natuurlijk handig dat de naam lijkt op de checkpoint-naam in de Waqua-invoerfile.

4.3 Het deterministische model

Het stochastische model, dat is beschreven in paragraaf 4.2, bestond onder andere uit een deterministisch model. Deze paragraaf gaat over de manier waarop dit deterministische model kan worden beschreven aan Calibriv-in-OpenDA. In ons voorbeeld staat deze beschrijving in het bestand `detmodel_rough_stat9.xml` (zie Tabel 4.3).

Het deterministische model bestaat uit vier onderdelen:

`waquaWrapper` Nadere beschrijving van de manier waarop Waqua werkt c.q. door OpenDA kan worden aangestuurd. Deze beschrijving is onderdeel van Calibriv-in-OpenDA en hoeft niet te worden aangepast door de gebruiker.

In de `waquaWrapper` staat beschreven dat een Waqua-run bestaat uit de volgende stappen:

1. Genereer de juiste invoer voor Waqua door
   - de invoer-directory te kopiëren;
   - SIMONA BOX files te maken voor ruwheidsparameters in u- en v-punten en voor de bodemligging. Dit onderdeel wordt niet gebruikt in Calibriv-in-OpenDA.

```
<?xml ...?>
<blackBoxModelConfig ...>

   <!-- OpenDA-description how Waqua works -->
   <wrapperConfig> <file>waquaWrapper.xml</file> </wrapperConfig>

   <aliasValues>
      <alias key="templateDir"       value="base"/>
      <alias key="SIMONADIR"         value=
                              "/v3/bo_omgeving/releases/simona2008-01/"
      <alias key="instanceDir"       value="work"/>
      <alias key="runid"             value="rough_stat9"/>
      <alias key="netcdfoutput"      value="waterlevelseries.nc"/>
      <alias key="inputFile"         value="siminp.stat_rough9"/>

      <alias key="roughParamsFile"   value="ruw.karak"/>
      <alias key="depthFile"         value=""/>
      <alias key="roughness-U-File"  value=""/>
      <alias key="roughness-V-File"  value=""/>
   </aliasValues>

   <exchangeItems>
      <vector id="A-411"
       ioObjectId="roughParams"    elementId="Rough_CODE_411_A"/>
       ....
      <vector id="A-419"
       ioObjectId="roughParams"    elementId="Rough_CODE_419_A"/>

      <vector id="M230.waterlevel"
       ioObjectId="%netcdfoutput%" elementId="M230.waterlevel"/>
       ...
      <vector id="M440.waterlevel"
       ioObjectId="%netcdfoutput%" elementId="M440.waterlevel"/>
   </exchangeItems>

   <doCleanUp>false</doCleanUp>

</blackBoxModelConfig>
```

Tabel 4.3: *XML-code voor de beschrijving van een deterministisch Waqua-model (complete Waqua schematisatie zonder onzekerheden) in Calibriv-in-OpenDA. Verkorte weergave van het bestand* `detmodel_rough_stat9.xml`.

- een include file te maken met ruwheidskarakteristieken.

2. Draai achtereenvolgens

   waqpre om de invoer te lezen;

   waqpro om de simulatie uit te voeren;

   getdata om de simulatieresultaten te vertalen naar NetCDF-formaat.

`AliasValues` Een lijstje met argumenten/parameters waarmee het model gespecifeerd wordt.

- `templatedir`

  De directory waar de invoerfiles staan (met startwaarden van de parameters, dus vóórdat het model is gecalibreerd). In de vorige paragraaf, bij het opzetten van de directorystructuur, hebben we deze directory al de naam `base` gegeven, dus dat moet hier worden ingevuld.

- `SIMONADIR`

  Dit heeft Waqua nodig, tenzij het al als environment variable is gezet.

- `inputFile`

  De Waqua invoerfile (`siminp`-file). Deze staat in de `detmodel/base` directory.

- `runid`

  De Waqua run-id. Deze wordt onder andere gebruikt voor de diagnosefiles zoals `waqpro-m.<runid>`.

- `Instancedir`

  Dit is het eerste deel van de naam van de werkdirectories waar een simulatie wordt uitgevoerd. Het tweede deel bestaat uit het nummer van de instantiatie. In elke instantiatie verschilt de inhoud van de file met de te calibreren invoer. Deze werkdirectories komen naast de `base` directory te staan. Deze naam is verder vrij te kiezen.

- `roughParamsFile`, `depthFile`, `roughness-U-File`, `roughness-V-File`

  De files waarin de te calibreren invoerparameters staan. In ons voorbeeld wordt alleen de invoer in de `roughParamsFile` gecalibreerd. Deze file is te vinden in de `base`-directory en heet in het voorbeeld `ruw.karak`. In de `siminp`-file wordt deze file ook als `include` gebruikt in het hoofdstuk NIKURADSE - GLOBAL - ROUGH_CHAR.

  In het tweede voorbeeld met de nieuwe ruwheidsparameters is de inhoud van `ruw.karak` iets anders; deze wordt als `include` gebruikt onder ROUGHCOMBINATION - GLOBAL - ROUGH_CHAR. Het nieuwe deel ROUGH_PARAMETER blijft in de `siminp`-file staan.

  Hoewel de `depthFile`, `roughness-U-File` en `roughness-V-File` niet worden gebruikt, moeten ze toch worden opgegeven: in ons geval met een lege string. In toekomstige versies zal dat niet meer nodig zijn.

- `netcdfoutput` Deze is voor de gebruiker meestal niet van belang, maar dient voor de koppeling tussen Waqua en OpenDA.

`exchangeItems` Een lijst van alle verbindingen van het deterministische (black-box) model met de buitenwereld. In het voorbeeld zijn het de (binnengaande) parameters voor de ruwheid en de (uitgaande) waterstanden zoals het model die berekent.

- De `id`'s moeten gelijk zijn aan de `id`'s van de parameters en predictors van het stochastische model.

  Merk op dat hier *alle* parameters moeten worden vermeld, terwijl in de XML-file voor het stochastische model alleen de *te calibereren* parameters hoefden te worden genoemd.

- Met de `ioObjectId` van de `vector` wordt aangegeven waar de exchange-items zich in bevinden. Zo bevinden de predicties zich in het bestand `%netcdfoutput%`, waarin de uitvoer van de simulatie staat. De ruwheidsparameters bevinden zich in het object `roughParams`, dat ruwheidsinvoer kan lezen, manipuleren en schrijven. Het valt op dat men alleen hieraan kan zien of een exchange-item invoer of uitvoer bevat.

  De gebruiker kan de `ioObjectId` niet vrij kiezen (en hoeft deze dus ook niet te veranderen). De `ioObjectId` van de parameters is van belang voor de calibratie. Deze moet namelijk altijd `roughParams` heten, omdat dit de `Id` van het object is die bij de ruwheidsparametersfile hoort (dit is zo gedefinieerd in de `WaquaWrapper`). Verder is de `ioObjectId` van de predicties gelijk aan de alias van de netcdf-file.

- De betekenis van de `elementID` is wat onduidelijk.

  Betreffende de parameters is het noodzakelijk dat de `elementID` begint met `rough_CODE`, omdat OpenDA expliciet deze string wil gebruiken. Daarna moet de code volgen zoals deze in de `roughParamsFile` staat (in dit geval heet deze `ruw.karak`), met tenslotte de parameter `a`, `b`, `c` of `d`.

  Iets soortgelijks geldt voor de `elementID` van de predicties. Ook dit luistert nauw. Deze string bestaat uit twee delen. Het eerste deel moet de naam van het station zijn zoals deze als checkpoint in de `siminp`-invoerfile is gegeven. Het tweede deel moet van OpenDA altijd de string `waterlevel` zijn.

  In een Waqua-model met de nieuwe ruwheidsformulering moet de `elementID` nog weer anders genoemd worden. Per ruwheidscode kan er nu sprake zijn van meerdere deelparameters. In het gedeelte `ROUGH_PARAM` in de `siminp`-invoerfile wordt aangegeven dat bij een ruwheidscode een parameter hoort (bijvoorbeeld de waterstand in punt P9: `r_code 202 WATERLEVEL P9`).

  In `ruw.karak` staat dan bijvoorbeeld

  ```
  r_code 202  PARAM = 0.221   a = 0.22
  r_code 202  PARAM = 0.225   a = 0.60
  ```

  Dit betekent dat er voor ruwheidscode 202 twee (deel)parameters zijn. Deelparameters worden in de `elementID` aangeduid met `rough_CODE_<rcode>_dp<parameterwaarde>_<par>`. In het voorbeeld wordt het exchange-item dus aangegeven met

  ```
  <vector id="A-202_1" ioObjectId="roughParams"
          elementId="Rough_CODE_202_dp0.221_A"/>
  <vector id="A-202_2" ioObjectId="roughParams"
          elementId="Rough_CODE_202_dp0.225_A"/>
  ```

  Verder worden deze deelparameters op dezelfde wijze behandeld als andere parameter exchange-items.

```
<?xml ...?>
<noosObserver ...>

    <timeSeries status="use" location="M230" standardDeviation="0.5" >
        M230.waterlevels
    </timeSeries>
        ...
    <timeSeries status="use" location="M440" standardDeviation="0.5" >
        M440.waterlevels
    </timeSeries>

</noosObserver>
```

Tabel 4.4: *XML-code voor de beschrijving van waterstandsmetingen in een zogenaamde stochastic observer. Verkorte weergave van het bestand* `obs_rough_20000707.xml`.

`doCleanUp` Het model wordt tijdens het calibratieproces meerdere keren uitgevoerd, telkens in een andere werkdirectory. Met deze optie kunnen de werkdirectories na afloop van de simulatie weer worden verwijderd. Voor de gebruiker is dit niet heel belangrijk.

4.4 De Stochastic Observer

De metingen worden in OpenDA aangeduid als een "stochastic observer", omdat zowel de metingen zelf, als hun onzekerheden worden weergegeven. Eén van de manieren waarop de metingen (en hun onzekerheden) kunnen worden aangeboden aan OpenDA is middels een "NoosTimeSeriesStochObserver".

In een NoosTimeSeriesStochObserver wordt voor elke tijdreeks een standaarddeviatie gegeven plus de naam van het bestand waarin de metingen staan, in "NOOS-formaat". Voor ons huidige voorbeeld ziet de stochastic observer er uit als in Tabel 4.4.

Hierbij merken we op dat voor de `status="use"`, ook gekozen kan worden voor `"validate"` of `"ignore"`. In deze gevallen wordt de betreffende tijdreeks respectievelijk alleen ter vergelijking dan wel helemaal niet gebruikt.

De OpenDA-programmatuur bevat zelf een duidelijke uitleg van het NOOS-formaat. We herhalen deze uitleg in gewijzigde vorm. In de uitleg wordt namelijk een voorbeeld gegeven, die we hebben vervangen door een daadwerkelijk door de stochastic observer gebruikte file: `M230.waterlevels`.

> *The NOOS format was created for the exchange of observations and forecasts at individual locations between the countries around the North Sea. See also* `http://www.noos.cc`. *This format is also the standard format for the Matroos database.*
>
> *Here is an example of this simple format:*
>
> `#-----------------------------------------------`

```
# Timeseries retrieved from timeseries_true
# Created at Fri Jun 26 2009
#-------------------------------------------------------
# Location    : M230
# id          : 230.waterlevel
# Unit        : waterlevel
# Timezone    : GMT
#-------------------------------------------------------
200007070000    -18.9000
200007070005    -18.8951
200007070010    -18.8785
```

The rules are:

- *lines with a '#' contain comments and metadata.*
- *lines with '#* $< property >$ *:* $< value >$*' contain the metadata. If the line does not fit this format it is a comment.*
- *data lines contain one* $< dateTime >$ $< value >$ *pair.*
- *the* $< dateTime >$ *format is 'yyyymmddHHMM'.*

Note that the property 'quantity' for the TimeSeries-object is labeled 'Unit' in the files.

Met deze uitleg tot onze beschikking komen we terug op de in het begin gestelde vraag hoe OpenDA de observaties en predicties precies moet koppelen.

De verbinding tussen exchange-item `M230.waterlevel` in de uitvoer van het model en deze NOOS-file met naam `M230.waterlevels` wordt gemaakt door de '#location' (in dit voorbeeld: "M230") te combineren met de '#unit' (in dit voorbeeld: "waterlevel") in de specifieke NOOS-file. Deze wordt dan gematcht met de `id` van de exchange-item, zoals in het deterministische model opgegeven. De `id` die in de NOOS-file staat hoeft dus *niet* overeen te komen met de `id` van de exchange-items.

Als de gebruiker deze constructie (dat de `id` van het exchange-item gelijk moet zijn aan de '#location '.'#Unit') ongewenst vindt, kan er ook worden gekozen dit te overtroeven in de XML-beschrijving van de Stochastic Observer (`obs_rough_20000707.xml`). De 'location' en/of 'quantity' kunnen hier ook gezet worden, bijvoorbeeld:

```
<timeSeries status="use" location="M230" standardDeviation="0.5" >
    M230.waterlevels
</timeSeries>
```

Of zelfs:

```
<timeSeries status="use" location="meetpunt_CCXXX" quantity="watersta
            standardDeviation="0.5" >
    M230.waterlevels
</timeSeries>
```

In dit laatste geval moet het bijpassende exchange-item `meetpunt_CCXXX.waterstand` heten.

Tenslotte zijn er nog twee manieren om een selectie van de observaties te maken:

- Het is mogelijk om de waarden in de meetreeks te limiteren. Een vorobeeld maakt dit duidelijk:

```
<timeSeries status="use" standardDeviation="0.5"
                              minValue="0.0" maxValue="10.0">
     stat2_waterlevel.river1d
</timeSeries>
```

  Hiermee worden alleen obervaties met waterstanden tussen 0 en 10 meter meegenomen.

- Ook is het mogelijk om het tijdsinterval te beperken:

```
<timeSeries status="use" standardDeviation="0.05"
        minDateTime="199101030000" maxDateTime="199101030020">
      stat1_waterlevel.river1d
</timeSeries>
```

  Een calibratie met deze selecties is te vinden in de test `waqua_river1d`.

4.5 Calibratiemethoden van OpenDA

De combinatie van een Stochastisch Model (paragraaf 4.2) en een Stochastische Observer (paragraaf 4.4) levert een optimalisatie-probleem op.

OpenDA kent verscheidene calibratiemethoden. Voor Calibriv-in-OpenDA hoort in principe de Dud methode te worden gebruikt, maar de geïnteresseerde gebruiker kan daar naar eigen inzicht van afwijken.

We bespreken eerst de gewone Dud-methode. SparseDud is een verfijning daarvan.

- De DuD-methode.

  Heel globaal is de werking als volgt. Eerst wordt voor de uitgangsinstellingen van de parameters bekeken hoe goed de predicties met de observaties overeen komen. Vervolgens wordt voor een aantal “zoekrichtingen” in de parameterruimte vastgesteld wat het effect is van de parameters op de predicties. Dan wordt in een aantal *buiteniteraties* systematisch telkens een “betere” instelling van de parameters gemaakt. Als de parameters goed genoeg zijn, eindigt de methode. Soms blijkt de nieuwe instelling opeens slechter; dan wordt in een zogeheten *binneniteratie* relaxatie toegepast.

  Deze methode heeft vrij veel rekentijd nodig om op te starten. Voor elke calibratieparameter moet een simulatie worden uitgevoerd om zijn effect vast te stellen. Zodra de methode op gang is, hoeft er echter maar één simulatie te worden uitgevoerd voor iedere volgende iteratie.

```
<?xml ...?>
<DudConfig>
   <costFunction weakParameterConstraint="false"
       class="org.openda.algorithms.SimulationKwadraticCostFunction" />

   <outerLoop maxIterations="10"
              absTolerance="0.01" relTolerance="0.01" />

   <lineSearch maxIterations="5" maxRelStepSize="10.0" >
       <backtracking shorteningFactor="0.5"
                     startIterationNegativeLook="3" />
   </lineSearch>
</DudConfig>
```

Tabel 4.5: *XML-code voor de beschrijving van de DuD calibratiemethode in Calibriv-in-OpenDA. Verkorte weergave van het bestand* `DudAlgorithm.xml`.

Een voorbeeld van de configuratiefile voor de DuD-methode staat in Tabel 4.5.

De volgende instellingen kunnen worden gekozen door de gebruiker (helaas vergt het gebruik van deze instellingen van de gebruiker tamelijk goede kennis van de werking van het algoritme):

- `weakParameterConstraint`

  De calibratie bestaat eruit dat we parameters zoeken waarmee de observaties en predicties zo dicht mogelijk bij elkaar liggen. We kunnen met deze optie tevens eisen dat deze parameters ook niet teveel van onze beginkeuze van parameters mogen afwijken. Dit betreft het aanzetten van de penalty-term. In dat geval gaat ook de standaarddeviatie van de parameters, zoals opgegeven in het stochastisch model, een rol spelen.

- `outerLoop`

  De gebruiker stelt in hoe vaak er maximaal een nieuwe keuze voor een parameterverzameling wordt gemaakt. De twee toleranties beïnvloeden het stopcriterium: zolang de huidige set parameters nog niet goed genoeg is (en het maximaal iteraties nog niet bereikt), volgt een nieuwe iteratie.

- `maxRelStepSize`

  Dit geeft aan hoeveel de parameters maximaal mogen veranderen van de ene op de andere buiteniteratie.

- `lineSearch`

  Dit heeft betrekking op de binneniteratie, waarin relaxatie wordt toegepast. Binneniteraties komen in het algemeen niet voor in een calibratie. Alleen in de eerste (buiten)iteraties, wanneer de parameters nog erg ver van hun optimum af zijn, kan binneniteratie nodig zijn.

  Het maximumaantal binneniteraties is gegeven door `maxIterations`. Telkens wordt de aanpassing aan de parameters met een factor `shorteningFactor` aangepast. Moch-

ten deze aanpassingen aan de parameters nog steeds een slechter resultaat tot gevolg hebben, dan wordt vanaf iteratienummer `startIterationNegativeLook` gekeken of een aanpassing van de parameters in omgekeerde richting wellicht beter werkt.

- de SparseDuD-methode.

  Deze methode is een verfijning van de DuD-methode. In de gevallen dat observaties maar van een beperkt aantal parameters afhangen, kunnen we het aantal zoekrichtingen op een slimme manier beperken. Dat betekent dat er minder simulaties nodig zijn, waardoor het calibratieproces veel sneller klaar is (met hetzelfde resultaat) dan met de DuD-methode. Een voorbeeld van de configuratiefile voor de SparseDuD-methode staat in Tabel 4.6.

  De invoerfile is een uitbreiding van de invoerfile voor de gewone DuD-methode. De uitbreiding bestaat eruit dat de afhankelijkheden expliciet worden opgegeven. Met `depends_on` wordt opgegeven dat een zekere predictie/observatie afhangt van een lijst parameters. Op deze manier definieren we eigenlijk een soort koppelingsmatrix met veel nullen (een ijle matrix), waarbij een niet-nul op plaats i, j betekent dat observatie i afhangt van parameter j.

4.6 Installatie en gebruik van Calibriv-in-OpenDA

- Een versie van OpenDA is te vinden bij de Simona-release. Dit bevat de gehele broncode van OpenDA. Naast allerlei unit tests bevinden de voor Simona relevante calibratie-toepassingen zich in de directory `tests`.
- Voor LINUX:
  - De locale settings moeten worden aangepast. We vermelden hier de instructies op de website (www.openda.org):

```
At the moment, scripts for csh and related shells are not included
with OpenDA.
If is possible to use OpenDA in conjunction with for instance tcsh ,
but you will have to convert the scripts yourself.

 In the bin directory of your OpenDA installation, copy the
settings_local_base.sh file to a new file named
settings_local_<hostname>.sh (unless that file already
exists). You can check your hostname with the hostname  command.

Then edit that file: enable the relevant lines and change the values
of the environment variables.

In the .bashrc in your home directory, add the following two lines:
export OPENDADIR=<bindir>, with <bindir> the location of the
bin directory of your OpenDA installation.

. $OPENDADIR/settings_local.sh $HOSTNAME
```

```
<?xml ...?>
<SparseDudConfig>
   <costFunction weakParameterConstraint="false"
         class="org.openda.algorithms.SimulationKwadraticCostFunction" /

   <outerLoop maxIterations="10"
              absTolerance="0.01" relTolerance="0.01" />

   <lineSearch maxIterations="5" maxRelStepSize="10.0" >
       <backtracking shorteningFactor="0.5"
                     startIterationNegativeLook="3" />
   </lineSearch>

   <dependencies>
       <obs id="M230.waterlevel">
           <depends_on>
               <par id="A-411"/>
               <par id="A-412"/>
           </depends_on>
       </obs>
       ...
       <obs id="M260.waterlevel">
           <depends_on>
               <par id="A-412"/>
           </depends_on>
       </obs>
   </dependencies>
</SparseDudConfig>
```

Tabel 4.6: *XML-code voor de beschrijving van de SparseDud calibratiemethode in Calibriv-in-OpenDA. Verkorte weergave van het bestand* `SparseDudAlgorithm.xml`.

```
Note that the '.' is significant in the latter of these lines.
```

We merken op dat, bij de keuze van `SIMONADIR`, voor een simulatie met waterstands-afhankelijke ruwheden een recente SIMONA-versie moet worden gebruikt.

– Nu kan een calibratie worden uitgevoerd. Voor een voorbeeld-calibratie, waarbij alle invoerfiles in deze paragraaf zijn gegeven, kan dat middels

```
% Application.sh -gui tests/waqua_usedoc_rough_stat9/Dud.oda
```

(*Merk op dat in deze release van OpenDA het hoofdbestand* `calib_waqua_rough_stat9.xml` *is hernoemd naar* `Dud.oda`.)
Gebruik voor het voorbeeld met de nieuwe ruwheidsformulering

```
% Application.sh -gui tests/waqua_usedoc_rough_fase2/Dud.oda
```

Gebruik voor een voorbeeld met selecties in de observaties

```
% Application.sh -gui tests/waqua_river1d/Dud.oda
```

Ook is het mogelijk om te experimenteren met andere invoerfiles, zoals voor SparseDud (`SparseDud.oda`).
In het scherm dat verschijnt, kan middels een druk op 'Start' de calibratie worden gestart. Het laatste deel van de uitvoer geeft onder andere de optimale keuze van de parameters.
Voor het uitvoeren van een eigen calibratie moet de gebruiker de directory-structuur opzetten zoals uitgelegd in paragraaf 3, de invoerfiles en metingen klaarzetten, de XML-files gereedmaken, en dan

```
% Application.sh -gui <modelnaam>/main.xml
```

Voor `main.xml` moet de gebruiker het hoofdbestand invullen dat is besproken in paragraaf 4.1.

- Voor WINDOWS:

  – De batch scripts `run_openda_gui.bat` en/of `run_openda_app.bat` moeten worden aangepast. Het gaat feitleijk slechts om het zetten van de `OPENDA_BINDIR`: de `bin` directory binnen de OpenDA installatie.

  – In een DOS-window kan een van de genoemde batch files worden gerund. Pas op: de `Simonadir` moet uiteraard bekend zijn, hetzij via de environment settings van Windows, hetzij middels een voorafgaand commando in het DOS-window.
  De invoerfile voor de calibratie kan als argument worden opgegeven:

```
%  run_openda_gui.bat ..\tests\waqua_usedoc_rough_stat9\Dud.oda
```

  en (voor de nieuwe ruwheidsformulering)

```
%  run_openda_gui.bat ..\tests\waqua_usedoc_rough_fase2\Dud.oda
```

  en (als voorbeeld van selecties in observaties):

```
%  run_openda_gui.bat ..\tests\waqua_river1d\Dud.oda
```

- Er verschijnt nu een scherm met vier tabbladen. In het tabblad “Input” vinden we een weergave van het gewenste bestand. De calibratie kan nu gestart worden door op de knop `Start` te drukken (de tekst is niet goed zichtbaar, het is de derde knop). Tijdens de calibratie kan er op `Stop` of `Pause` gedrukt worden maar dit werkt meestal niet goed.

  In het tabblad “Output” is een tabel die tijdens de calibratie wordt bijgewerkt. Hierbij staan voor elke simulatie van het model de resulterende kostenfunctie, en de bij deze simulatie horende parameterkeuzes. Merk op dat niet de parameterwaarden zelf maar hun afwijkingen worden vermeld. Bij “Iteration 1” (de eerste simulatie) zijn alle parameterafwijkingen nul. Als de calibratie beëindigd is volgens het stopcriterium wordt in de laatste regel van de tabel de optimale keuze van (afwijkingen van) parameters gegeven.

  In het tabblad “Costfunction” wordt de kostenfunctie als functie van het aantal simulaties grafisch weergegeven.

Bijlage A

De Waqua-wrapper van OpenDA

OpenDA beschikt zelf al over een beschrijving van de werking van het simulatiepakket Waqua. Deze beschijving is zelfs een beetje uitgebreider dan nodig is voor Calibriv-in-OpenDA. Tegelijkertijd is het de binnenste schil van het blackbox-model waarvoor in paragraaf 4.2 en 4.3 de lagen daarbuiten al zijn beschreven.

Feitelijk betreft het hier eigenlijk programmatuur (de werking van Waqua uitgelegd aan OpenDA), dus dit bestand hoeft niet te worden aangepast.

In de modelbeschrijving staat beschreven dat een Waqua-run bestaat uit de volgende stappen:

1. Genereer de juiste invoer voor Waqua door
   - de invoer-directory te kopiëren;
   - SIMONA-BOX files te maken voor ruwheidsparameters in u- en v-punten en voor de bodemligging;
     dit onderdeel wordt niet gebruikt in Calibriv-in-OpenDA.
   - een include file te maken met ruwheidskarakteristieken.
2. Draai achtereenvolgens

   waqpre om de invoer te lezen;

   waqpro om de simulatie uit te voeren;

   getdata om de simulatieresultaten te vertalen naar NetCDF-formaat.

De gebruiker hoeft over het Waqua-model alleen nog maar op te geven

- hoe de invoer-directory heet;
- hoe de verschillende te genereren invoerbestanden heten;
- welke *run-id* gebruikt moet worden;
- hoe het uitvoerbestand in NetCDF-formaat heet.

Dit gebeurt in de XML-file van het deterministisch model (zie paragraaf 4.3).

Merk op dat deze beschrijving tamelijk algemeen is. Het heeft daarom ook niet zoveel met ons specifieke voorbeeld te maken: een andere Waqua-simulatie kan deze algemene Waqua-wrapper ook gebruiken. De specifieke informatie over parameters en predicties is te vinden in de twee schillen boven deze wrapper: het deterministisch en het stochastisch model.

Nu volgt de voorbeeld-invoer voor een Waqua-model.

```
<?xml version="1.0" encoding="UTF-8"?>
<blackBoxWrapperConfig xmlns=...>

   <aliasDefinitions defaultKeyPrefix="%" defaultKeySuffix="%" >
      <alias key="SIMONADIR"/>
               ....
   </aliasDefinitions>

   <run>

       <!-- for each model instance, the template directory will be clone
            to create the instance directory -->
       <initializeActionsUsingDirClone
                  instanceDir="%instanceDir%%instanceNumber%"
                  templateDir="%templateDir%"/>

       <computeActions>
           <action exe="%SIMONADIR%/bin/waqpre.pl"
                   workingDirectory="%instanceDir%">
               <arg>-runid</arg>
               <arg>%runid%</arg>
               <arg>-input</arg>
               <arg>${inputFile}</arg>
               <arg>-isddh</arg>
               <arg>n</arg>
               <arg>-back</arg>
               <arg>n</arg>
               <checkOutput file="waqpre-m.%runid%"
                            expect="Finished Succesfully"/>
               <checkOutput file="SDS-%runid%"/> <!-- should exist -->
           </action>
           <action exe="%SIMONADIR%/bin/waqpro.pl"
                   workingDirectory="%instanceDir%">
               <arg>-runid</arg>
               <arg>%runid%</arg>
               <arg>-isddh</arg>
               <arg>n</arg>
               <arg>-back</arg>
```

```
                <arg>n</arg>
                <arg>-npart</arg>
                <arg>1</arg>
                <checkOutput file="waqpro-m.%runid%"
                             expect="Finished Succesfully"/>
                <checkOutput file="SDS-%runid%"/> <!-- should exist -->
            </action>
            <action exe="%SIMONADIR%/bin/getdata.pl"
                    workingDirectory="%instanceDir%">
                <arg>-f</arg>
                <arg>SDS-%runid%</arg>
                <arg>-v</arg>
                <arg>ZWL,NAMWL,MWL,NWL,XZETA,YZETA,ITDATE</arg>
                <arg>-o</arg>
                <arg>netcdf</arg>
                <arg>-d</arg>
                <arg>%netcdfoutput%</arg>
                <checkOutput file="%netcdfoutput%"/>
            </action>
        </computeActions>
        <finalizeActions/>
    </run>

inputOutput>
        <ioObject className="org.openda.blackbox.io.SimonaBoxFile">
            <file>${roughness-U-File}</file>
            <id>roughness-U</id>
            <arg>roughness-U</arg>
        </ioObject>
        <ioObject className="org.openda.blackbox.io.SimonaBoxFile">
            <file>/roughness-V-File/</file>
            <id>roughness-V</id>
            <arg>roughness-V</arg>
        </ioObject>
        <ioObject className="org.openda.blackbox.io.SimonaBoxFile">
            <file>%depthFile%</file>
            <id>depthFile</id>
            <arg>depth</arg>
        </ioObject>
        <ioObject className="org.openda.blackbox.io.SimonaRoughParamsFile">
            <file>${roughParamsFile}</file>
            <id>roughParams</id>
            <arg>roughParams</arg>
        </ioObject>
        <ioObject className="org.openda.blackbox.io.SimonaNetcdfFile">
```

```
            <file>%netcdfoutput%</file>
            <id>%netcdfoutput%</id>
            <arg>%refDate%</arg>
        </ioObject>

</inputOutput>

</blackBoxWrapperConfig>
```

Als toevoeging op de algemene beschrijving gaan we nog op enkele punten van deze voorbeeldinvoer in:

- de `alias`-definities zijn in het deterministisch model reeds voorgeschreven.
- Het blackbox-model bestaat dus eigenlijk uit het achtereenvolgens draaien van de drie programma's `waqpre`,`waqpro` en `getdata`, elk met hun eigen argumentenlijst. Bij sommige argumenten wordt gebruik maakt van de aliases zoals eerder gedefinieerd.
- Er zijn vier invoerfiles en een uitvoerfile. De laatste twee komen overeen met de "object-id's" zoals die in de invoerfile voor het deterministisch model zijn beschreven.