Environmental Modelling & Software 57 (2014) 76–89

Contents lists available at ScienceDirect

Environmental Modelling & Software

journal homepage: www.elsevier.com/locate/envsoft

Data assimilation framework: Linking an open data assimilation library (OpenDA) to a widely adopted model interface (OpenMI)

Marc E. Ridler [a,d,*], Nils van Velzen [b,e], Stef Hummel [c], Inge Sandholt [d], Anne Katrine Falk [a], Arnold Heemink [b], Henrik Madsen [a]

[a] DHI, Agern Allé 5, DK-2970 Hørsholm, Denmark
[b] TUDelft, Postbus 5, 2600 AA Delft, The Netherlands
[c] Deltares, P.O. Box 177, 2600 MH Delft, The Netherlands
[d] University of Copenhagen, Øster Voldgade 10, 1350 København K, Denmark
[e] VORtech, P.O. Box 260, 2600 AG Delft, The Netherlands

ARTICLE INFO

Article history:
Received 8 August 2013
Received in revised form
10 February 2014
Accepted 10 February 2014
Available online 11 March 2014

Keywords:
OpenDA
OpenMI
Data assimilation
Hydrological modeling
Kalman filter
Uncertainty

ABSTRACT

Data assimilation optimally merges model forecasts with observations taking into account both model and observational uncertainty. This paper presents a new data assimilation framework that enables the many Open Model Interface (OpenMI) 2.0 .NET compliant hydrological models already available, access to a robust data assimilation library. OpenMI is an open standard that allows models to exchange data during runtime, thus transforming a complex numerical model to a 'plug and play' like component. OpenDA is an open interface standard for a set of tools, filters, and numerical techniques to quickly implement data assimilation. The OpenDA–OpenMI framework is presented and tested on a synthetic case that highlights the potential of this new framework. MIKE SHE, a distributed and integrated hydrological model is used to assimilate hydraulic head in a catchment in Denmark. The simulated head over the entire domain were significantly improved by using an ensemble based Kalman filter.

Software availability

The OpenDA–OpenMI Framework described in this article is currently available for download in the development version of OpenDA and will be available in the release version soon. OpenDA software is available under the GNU Lesser General Public License from: www.openda.org. OpenMI Software Development Kit (SDK) is available under the GNU Lesser General Public License from www.openmi.org. The .NET Framework is available for download under a proprietary license from www.microsoft.com/net.

1. Introduction

Recent advances in hydrological modeling have led to improved water resource management practices, greater crop production and better flood forecasting systems. However, uncertainty is inherent in all deterministic numerical models; stemming from improper model structure, parameters values, initial conditions and forcing data, ultimately leading to imperfect forecasts. It remains a crucial challenge to account for system uncertainty, so as to provide model outputs accompanied by a quantified confidence interval. Properly characterizing and reducing uncertainty opens-up the opportunity for risk-based decision making and more effective emergency and disaster management.

Data assimilation optimally merges information from independent observations with a deterministic model, taking into account the uncertainty of both the model and the observations. It is a powerful and versatile means to improve prediction accuracy while providing uncertainty estimates. Data assimilation has been used extensively in the geosciences, notably for meteorological forecasting, characterizing oil and gas reservoirs, and estimating land surface, ocean and atmospheric conditions. Recent computational advances and the burgeoning of observational data has spurred research into data assimilation and the implementation of operational real-time assimilation systems.

Traditionally, in-situ observations of discharge, soil moisture, hydraulic head and snowpack have been used to improve hydrological predictions of stream flow and other hydrological variables (Komma et al., 2008; Hendricks Franssen and Kinzelbach, 2008; Hendricks Franssen et al., 2011; Camporese et al., 2009;

* Corresponding author. University of Copenhagen, Øster Voldgade 10, 1350 København K, Denmark.
E-mail address: mer@dhigroup.com (M.E. Ridler).

http://dx.doi.org/10.1016/j.envsoft.2014.02.008
1364-8152/

Moradkhani et al., 2005b). In-situ measurements have the advantage of being accurate and timely (when automated), but lack spatial support. Recently, the quality and abundance of satellite data has made possible the assimilation of remotely sensed variables; soil moisture, snow water equivalent and/or snow cover area or extent, surface water elevation, terrestrial water storage and land surface temperature. For a more in depth discussion, see review papers (Liu et al., 2012; Moradkhani, 2008).

The aforementioned in-situ and satellite variables have been assimilated (in various combinations) with numerous different types of environmental models. Some notable examples include: physically-based land surface, distributed hydrologic, conceptual rainfall-runoff, hydraulic, groundwater, coupled surface-subsurface, biogeochemical, and sediment transport models (Liu et al., 2012; Moradkhani, 2008). Although these models represent different scales and processes, they can benefit from the incorporation of observation data.

Thus far, many of the implementation efforts have been model specific, which require coding both complex algorithms as well as access to the numerical core of the models. A means to deal with the level of interaction between model and assimilation algorithm is the use of an Open Model Interface (OpenMI) (Moore and Tindall, 2005; Gregersen et al., 2007). This open standard interface allows models to exchange data during runtime, thus transforming a complex numerical model to a *plug and play* like component. The standard has been implemented by twelve major model providers for simulating diverse processes such as urban water, groundwater, rivers, catchments, river-runoff, and sediment transport (OpenMI). OpenMI was designed with data assimilation in mind (Gijsbers et al., 2010), providing clear procedures for data exchange and model control.

Intense research over the last few years has also focused on developing new and increasingly sophisticated data assimilation algorithms, from simple rule-based direct insertion methods to advanced smoothing and sequential techniques. Some algorithms, such as variational methods (for example 3D-VAR) popular in meteorological science, require an adjoint of the observation operator, which can be difficult to develop. On the other hand, ensemble based algorithms are model generic, for example the Ensemble Kalman Filter (EnKF) (Evensen, 2003) and its deterministic formulation, the Ensemble Square-Root (EnSR) (Tippett et al., 2003) filter. For non-linear models and non-Gaussian observations, Particle Filters (PF) (Moradkhani et al., 2005a) evolves a sample of the state space forward using a sequential Monte Carlo method. Numerical techniques are often necessary in data assimilation to avoid filter divergence (insufficient spread in the ensembles) or spurious correlations between observations and state variables (Hamill et al., 2001). To minimize spurious correlations, a distance based regularization techniques can reduce the spatial domain of influence of observations during the update, also improving the forecast error covariance estimate (Hamill et al., 2001; Anderson, 2001; Sakov and Bertino, 2011). To avoid filter divergence, an inflation factor can be selected empirically (Hamill et al., 2001; Anderson, 2001).

Open source data assimilation libraries are available which implement these algorithms and numerical techniques. Notably, the Data Assimilation Research Testbed (DART) (Anderson et al., 2009) and The Parallel Data Assimilation Framework (PDAF) (Nerger and Hiller, 2013) are based on Fortran code compilable in the Linux/Unix environment. DART is a community facility for ensemble data assimilation, developed and maintained by the Data Assimilation Research Section (DAReS) at the National Center for Atmospheric Research (NCAR). It provides modelers, observational scientists, and geophysicists with powerful, flexible assimilation tools that are easy to implement and use and can be customized to support efficient operational data assimilation applications. Similarly, PDAF provides fully implemented and optimized data assimilation algorithms for ensemble based Kalman filtering. PDAF is optimized for large-scale models usually run on big parallel computers, but can also be used for smaller models.

OpenDA is another framework launched in 2010 (OpenDA). OpenDA is an open interface standard for (and free implementation of) a set of tools, filters, and numerical techniques to quickly implement data assimilation. The library is actively developed and includes efficient implementation of the algorithms just mentioned. OpenDA was chosen for the OpenMI bridging as it is the most similar to OpenMI in terms of data exchange structure (IExchangeItem) and time stepping. Thanks to IKVM (IKVM.NET), the OpenDA was converted from Java to the .NET Common Intermediate Language, so that the framework and MIKE SHE could all be run together. Furthermore, no changes within the MIKE SHE model engine was possible, making OpenMI compliance the only option.

This paper describes a new data assimilation framework that enables the many OpenMI 2.0 .NET compliant hydrological models already available, to have access to a robust and flexible data assimilation library. Popular hydrological models that simulate rivers, groundwater, land surface and many other processes, can now take advantage of observation data for model optimization and uncertainty analysis. The framework is designed to grant users access to numerous complex assimilation algorithms, uncertainty models, and distance based regularization techniques, for state or joint state-parameter updating, with a minimal amount of programming. Furthermore, the framework is flexible enough to be applied to any deterministic environmental model, provided it is compliant with the OpenMI 2.0 standard. The OpenDA association is actively adding new filters and features, making it an ideal solution for both research and real-time application purposes.

This paper is organized as follows. The theoretical and mathematical basis of data assimilation and some key filters commonly used, is described in Section 2. In Section 3, OpenDA is described and in Section 4, OpenMI is presented. Finally, in Section 5 the framework that ties these two open-source initiatives together is described along with various components of the framework and the challenges overcome. A synthetic test case in Section 6 highlights the potential of this new framework. MIKE SHE, a distributed and integrated hydrological model is used to assimilate hydraulic head in the Karup catchment in Denmark. The simulated head over the entire domain were significantly improved by using an ensemble based Kalman filter. Furthermore, the river discharge estimates (which was not itself assimilated) were also corrected. Discussion and conclusions are presented in Section 7.

2. Data assimilation

The basic principle of data assimilation is to incorporate measurement information into a numerical model with the aim to improve model results by error minimization. It is a feedback process where model forecasts are conditioned to the observations. As depicted in Fig. 1, the information gained from assimilating an observation can be used to correct (Refsgaard, 1997b; Madsen and Skotner, 2005):

1. Input forcing (**u**). The expectation is that minimizing model input error will ultimately improve model results. The simulation model is then run with the corrected forcing.
2. State variables (**x**) are the internal variables of the mathematical model. Kalman filtering or Bayesian estimation techniques are the most widely applied data assimilation procedures. The main

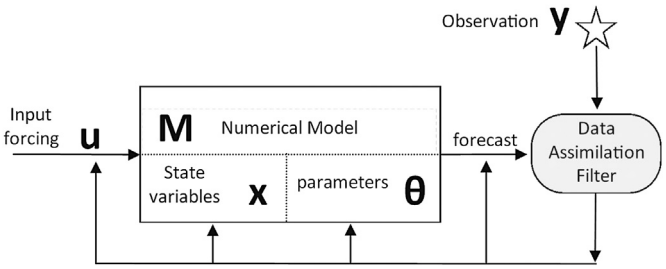


Fig. 1. Data assimilation filters can update the (1) input forcing **u**, (2) state variables **x**, (3) model parameters θ, and (4) model output forecast (Refsgaard, 1997b).

advantage of these methods is that they provide estimates of the model states and an estimate of model forecast uncertainties.

3. Parameters (θ). Model parameters can be updated as a stand-alone procedure or in combination with state updating.
4. Model output can be corrected by deriving a prediction error model by fitting the deviation between model forecasts and observations. Errors are often correlated in time making it possible to forecast future error values using a time series model such as autoregressive moving average (ARMA).

The state-based formulation for a numerical model without assimilation is,

$$\mathbf{x}_k = \mathbf{M}(\mathbf{x}_{k-1}, \mathbf{u}_k, \theta, \varepsilon_k) \tag{1}$$

where $\mathbf{M}(\bullet)$ is the model operator that solves the mathematical equations necessary to bring the model forward from time step $k-1$ to k. The variable $\mathbf{x}$ is the state vector, $\mathbf{u}$ the model forcing, and θ represents model parameters (assumed in this formulation constant over time). The ε_k is the process noise term encompassing all model uncertainty which can stem from: (1) model input ($\mathbf{u}$) uncertainty such as precipitation data, (2) imperfection in model formulation, structure, and resolution ($\mathbf{x}_k$) and parameterization (θ), and (3) uncertainty in model initial conditions ($\mathbf{x}_{k-1}$). According to Equation (1), a model will provide an estimate of the state of the system at time k given the state at the previous step, a set of model parameters and forcing data.

When an observation measurement vector $\mathbf{y}$ is available, the model state can be related to that observation using the measurement operator $\mathbf{H}$ such that

$$\mathbf{y}_k = \mathbf{H}_k\mathbf{x}_k + \eta_k \tag{2}$$

with measurement error η_k. Spatial mapping is sometimes required to translate between point measurements to a region of the model and can be a non-linear process.

With sequential data assimilation, observations are used as soon as they are available to correct the present state of the model. As illustrated in Fig. 2, the forecast f (Fig ◦) of the state vector is updated to the analysis a (Fig •) whenever an observation $\mathbf{y}$ (Fig ★) is available. In state-space formulation, an analysis equation can be formed from Equations (1) with (2) under the condition that the best posterior estimate is a linear combination of the prior best estimate and the observations:

$$\mathbf{x}_k^a = \mathbf{x}_k^f + \mathbf{K}_k(\mathbf{y}_k - \mathbf{H}_k\mathbf{x}_k^f) \tag{3}$$

The gain matrix $\mathbf{K}$, weights the forecast and the forecast-observation misfit, and reflects how much emphasis should be placed on the observations or the model forecast. From Equation (3), the greater the $\mathbf{K}$ value, the more observations are trusted over the model forecast leading to a model analysis close to the observations.

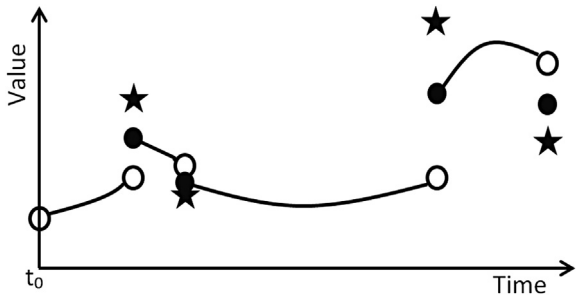


Fig. 2. Sequential data assimilation approach. Whenever an observation is available (★), the model forecast (◦) is updated (•) to a value closer to the observation. The updated state is then the starting point for the next model forecast.

The gain matrix $\mathbf{K}$ also reflects how corrections at measurement locations are transferred to the entire state vector. The fundamental task of the filtering algorithm is to determine the optimal gain taking into account model as well as measurement uncertainty.

The Kalman Filter (KF) is a comprehensive filtering scheme where the gain (Kalman gain) is determined by least square minimization of the expected error of the updated state. The gain is given by

$$\mathbf{K}_k = \mathbf{P}_k^f\mathbf{H}_k^T(\mathbf{H}_k\mathbf{P}_k^f\mathbf{H}_k^T + \mathbf{R}_k)^{-1} \tag{4}$$

where $\mathbf{P}_k$ is the error covariance matrix, a measure of the estimated accuracy of the state estimate, and $\mathbf{R}_k$ the measurement error.

The KF's main strength is to explicitly take model and measurement uncertainties into account in the updating process as well as provide an estimate of the uncertainty of the system state. But important caveats exist.

- An initial guess of both the model uncertainty and measurement uncertainty must be provided, which can be tricky to determine depending on the complexity of the system.
- KF assumes that the errors in the measurement and model are Gaussian.
- KF requires the system model to be linear.
- Propagation of the stochastic model can be computationally expensive.

Luckily, variations to the KF have been developed to partially overcome these limitations, rendering data assimilation feasible for operational forecasting systems. Some notable filters often used for environmental applications and available in OpenDA are outlined below. For a review of data assimilation in operational hydrological forecasting, see (Liu et al., 2012) and for a review of hydrologic remote sensing and land surface data assimilation, see (Moradkhani, 2008).

2.1. Ensemble Kalman Filter (EnKF)

The ensemble Kalman Filter (EnKF) (Evensen, 2003) is a popular Monte Carlo approximation of the KF, which avoids evolving the covariance matrix of the probability density function of the state vector $\mathbf{x}$. The probability density of the state is estimated by a finite number N of randomly generated system states (Ensemble) such that $\mathbf{X} = [\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_N}]$ with randomly generated system noise. The update of the state then becomes

$$\mathbf{X}_k^a = \mathbf{X}_k^f + \mathbf{K}_k(\mathbf{Y}_k - \mathbf{H}_k\mathbf{X}_k^f) \tag{5}$$

where $\mathbf{Y} = [\mathbf{y} + \upsilon_1, \mathbf{y} + \upsilon_2, \ldots, \mathbf{y} + \upsilon_N]$ denotes a matrix with realizations of the observed values, and υ_i realization of the

observation noise. Thus with the EnKF, observations are perturbed in order to prevent underestimation of the model error covariance. The disadvantage is that this introduces sampling error, which makes the filter suboptimal particularly for small ensembles (Whitaker and Hamill, 2002). EnKF has been extensively used in hydrological applications as outlined in the reviews papers (Liu et al., 2012; Moradkhani, 2008).

2.2. Ensemble Square-Root filter (EnSR)

With EnKF, small ensembles have only a few degrees of freedom available to represent errors, and suffer from sampling errors in the observations that can further degrade the forecast error covariance representation. In EnSR, the addition of the noise υ in the update of the ensemble is omitted rendering the solution deterministic; see (Tippett et al., 2003) for details. EnSR updates the ensemble in two steps: (1) the ensemble mean is updated with the standard Kalman filter analysis equation, and then (2) the ensemble anomalies are transformed so that the ensemble-based analysis error covariance matches the theoretical solution given by Kalman filter theory (van Velzen and Segers, 2010). This transformation is not unique and depending on the implementation could lead to bias in the solution. The EnSR method implemented in OpenDA is the one presented by Sakov and Oke (2008) that is symmetric and mean preserving. EnSR has been extensively applied in hydrological models as outlined in the review papers (Liu et al., 2012; Moradkhani, 2008).

2.3. Particle Filter (PF)

The Particle Filter (PF) is also known as a sequential Monte Carlo (SMC) method. Similar to the EnKF, particle filtering evolves a sample of the state space forward using the SMC method to approximate the predictive distribution. Unlike the EnKF, the PF performs updating on the particle weights instead of the state variables, which has the advantage of reducing numerical instability especially in physically-based or process-based models (Moradkhani et al., 2005a; Van Leeuwen, 2009; Liu et al., 2012). Another key advantage is that PF can be applied to systems and models that are non-linear and non-Gaussian. However an efficient PF can be computationally complex and often PF requires more particles (larger ensembles) than other filtering methods.

3. OpenDA

OpenDA is an open-source toolbox for developing and applying data assimilation and calibration algorithms to dynamic models (OpenDA). The design philosophy of OpenDA is to breakdown data assimilation into a set of building blocks using the principles of object oriented programming. The OpenDA toolbox consists of two parts; a description of the building blocks with their interfaces, and a library of default implementations of those building blocks. It stems from the merger of two data assimilation initiatives: COSTA (van Velzen and Verlaan, 2007; van Velzen and Segers, 2010; Heemink et al., 2010) and DATools (Weerts et al., 2010).

OpenDA is programmed in Java with the possibility to perform the core calculations in other languages like C, C#, and Fortran 77/95. Once a model is prepared for OpenDA, it is possible to try out various off-the-shelf data assimilation algorithms without any further programming. OpenDA was designed for speed and efficiency to open up the possibility of operational data assimilation. It has been shown that the performance is close to dedicated implementations (van Velzen and Verlaan, 2007) and furthermore can automatically propagate the ensemble of models either serially or concurrently.

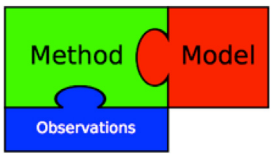


Fig. 3. OpenDA defines three building blocks: *method* (the data assimilation or calibration algorithms), *observations* (the stochastic observer for handling the observations), and *model*.

3.1. OpenDA components

As illustrated in Fig. 3, OpenDA defines three major building blocks: *method* (the data assimilation or calibration algorithms), *observations* (the stochastic observer for handling the observations), and *model*. The end user can make arbitrary combinations of these building blocks to set up an experiment without the need of any programming. The system is configured using an XML file where a user defines options regarding: algorithm type (EnSR, EnKF,.. etc.) and ensemble size; the variables to assimilate and grid location within the model; and associated uncertainties of the model, forcing data, parameters, and observations. Using the default implementations, observations can be stored in various formats like CSV, NOOS or in an SQL database. OpenDA can produce output in various formats such as ASCII, Matlab, and NetCDF to easily analyze filtering performance and graph results.

3.2. Model specific implementation

The OpenDA model interface has already been implemented for over 20 dynamic model codes (van Velzen et al., 2013). One implementation approach is to use the *Black-Box* interfaces provided with OpenDA, but can be tricky as it requires core model access to perform tasks such as, 'perform a time step', 'deliver the model state' or 'accept a modified state'. Furthermore, OpenDA is compiled in Java which might require the model to implementation the Java Native Interface (JNI) framework to exchange data between other languages and Java. There are two main elements a user needs to implement to render a model OpenDA compliant (see Fig. 4); the *Model Factory* and the *Model Instance* which must implement the following tasks:

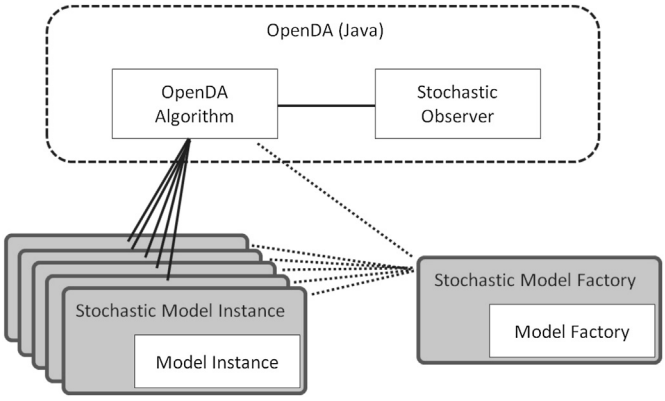


Fig. 4. Structure of OpenDA operational framework. The *OpenDA Algorithm* is in charge of the workflow; each algorithm performs its own typical steps, in which it compares model computations with observations. The *Stochastic Observer* handles reading observational data. When the algorithm needs a new *Stochastic Model Instance*, it will ask the *Stochastic Model Factory* for it. The stochastic part of the model factory and instances are provided by OpenDA, so the model provider only needs to implement the deterministic part; the *Model Factory* and *Model Instance*.

1. *Model Factory*
   - Create a model instance (ie, let the model clone itself).
2. *Model Instance*
   - Propagate model (to a given date/time).
   - Get variables, parameters or state vector.
   - Set variables, parameters or state vector.
   - Free instance (when OpenDA is finished)

For ensemble based assimilation, OpenDA calls the *Stochastic model factory* once during the initialization phase during which time an ensemble of *stochastic model instances* are spawned and initialized. The stochastic factory makes stochastic instances out of deterministic model instances. For N number of ensembles, there will be $N + 1$ instances where the extra one is the *assimilated* (or *main*) model, with corrected state equal to the ensemble mean and represents the filter's best estimate. Once the factory finishes, each one of those instances must be accessible directly by OpenDA by means of an identifier (process id). OpenDA controls the propagation of each one of the $N + 1$ instances until there is an observation available. Once an observation is available, the algorithm gets the observation information and gets the states of each of the N ensemble members, does the assimilation calculations and finally sets the new state to the N instances and the best guess to the *main* state. The algorithm proceeds in this manner until the end time of the model instances.

3.3. Data exchange

Data in OpenDA is defined with the interface *IExchangeItem* for which the values can be retrieved from or provided to the model. The model tells which exchange items it has, after which the exchange item is used to access the values. The *IExchangeItem* contains property information regarding the variable quantity, location, value type usually in double array, times in Modified Julian Day and identifier. The getting and setting of the state vector **x** in OpenDA is done with one of the *IExchangeItem* implementations: such as *getValues*, or *setValues*. OpenDA handles all the logic to adapt the data objects to the algorithmic state vector **x**.

4. OpenMI

OpenMI is an open standard interface that defines general model functionality and operation, independent of specific programming language or model domain (Knapen et al., 2013). This open standard allows models (primarily hydrological) a general means to exchange data during runtime, thus facilitating the interactions between models and data sources (OpenMI). The interface is flexible enough so that models can interact even if the model is coded in a different language, represent processes from a different domain or have different spatial and temporal resolutions. OpenMI source code is open and available under the Lesser GPL license conditions. Once a model component is compliant with the standard, it can, without any programming, be configured to exchange data during computation. Thus, transforming a complex numerical model to a *plug and play* like component as depicted in Fig. 5.

The OpenMI Association maintains a list on their website of compliant hydrological models from 12 major model providers. A non exhaustive list of OpenMI 2.0 compliant models can be found in Table 1.

A key component of the OpenMI 2.0 standard is the *ITimeSpaceComponent* which is the main interface to the model engine component through which data inside the component is accessible and from which the model can be controlled (see Fig. 5). The *ITimeSpaceComponent* requires a number of methods to be implemented to direct the model engine to *Prepare* itself, *Initialize*, and *Update* (perform a model time step). For data assimilation purposes, the *ITimeSpaceComponent* provides the means to control the model (Initialize, Prepare, Update, Validate, and Finish).

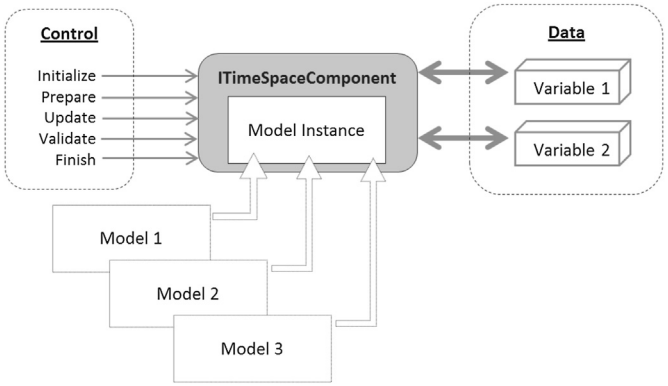


Fig. 5. OpenMI conceptualization where a numerical environmental model is wrapped to implement an *ITimeSpaceComponent*. The white boxes in the center of the *ITimeSpaceComponent* are interchangeable with any other OpenMI compliant model depicted below. The core control calls (Initialize, Prepare, Update, Validate, and Finish) are then model independent. The data is defined by the *IExchangeItem* interface and is also model independent.

Data assimilation requires that data (variables, parameters, and forcing) forming the state vector **x** be exchangeable between the model and the assimilation algorithm. OpenMI provides standardized interfaces to define, describe and transfer data between software components that run sequentially. In OpenMI, so-called 'exchange items' (*IExchangeItem*) are defined to describe the data that a component can provide (the outputs) or accept (the inputs). Each input or output item that can be exchanged contains information on the quantity it represents, optionally the geometry where it applies, and optionally the time frame for which it applies. For model output, the *IBaseOutput* is the exchange item from which the accessor *Values* provides data in an array of doubles. Conversely, *IBaseInput* is the exchange item for providing the model with data via the accessor *Values*. During the model initialization stage, output exchange items must be connected to input exchange items by means of the *Consumer* and *Provider* property.

Thus, OpenMI defines a standard way to access a model, control it, get and set values within the model with spatial and temporal definitions as well as get information regarding the current time step. For further information, see (Gregersen et al., 2005, 2007; Moore and Tindall, 2005) and for details regarding the newest 2.0 release (Donchyts et al., 2010).

Table 1
List of OpenMI 2.0 models and providers. For an up to date list and for OpenMI 1.4 compliant models, see the OpenMI website.

Provider	Model
Deltares	Sobek RR
	Sobek CF
	Sobek RTC
	DeltaShell-Flow1D
DHI	Mike 11
	Mike 1D
	Mike Urban
	Mike SHE
	Mike 21
	FEFLOW
	WEST
	MOUSE
Innovyze	InfoWorks-CS
	InfoWorks-RS
HR Wallingford	FluidEarth

5. OpenDA–OpenMI framework

The main function of the OpenDA–OpenMI framework is to tie these two open source components together. In this way, an OpenMI compliant model can have access to the features of OpenDA without much programming and customization. This framework is designed to work for all OpenMI 2.0 .NET compliant models. The components of this framework are outlined below and visualized in Fig. 6 and the functions of the components are visualized as a flow diagram in Fig. 7.

5.1. Components

5.1.1. ITimeSpaceComponent Factory

Similar to the *Stochastic model factory* defined in Section 3.2, this component is model specific and needs to be programmed by the user. The *ITimeSpaceComponent Factory* will create $N + 1$ OpenMI instances of the model (where N is the ensemble size). Depending on the model and setup chosen, each instance might require its own disk space for computation or output files. The factory returns an *ITimeSpaceComponent* with a model prepared and initialized along with exchange items of interest initialized according the OpenMI standard as described in Section 4. Depending on the setup, a user can create instances that are pre-perturbed or can configure OpenDA to perturb the instances during initialization and runtime. The *Factory* corresponds to the column (1)Initialize in Fig. 7.

5.1.2. OpenDA assimilation

The selected assimilation routine then begins where the ensemble of *ITimeSpaceComponent* are stepped forward to a time when a stochastic observation is present. The ensemble members can either propagate serially or concurrently, depending on the XML settings. OpenDA will calculate the new state for the main model, and will send the new updated values to the model via the OpenDA–OpenMI bridge. The states of the ensemble might also be updated during this time. Once updated, the ensemble members propagate forward until a new observation is present or until the model simulation time is complete.

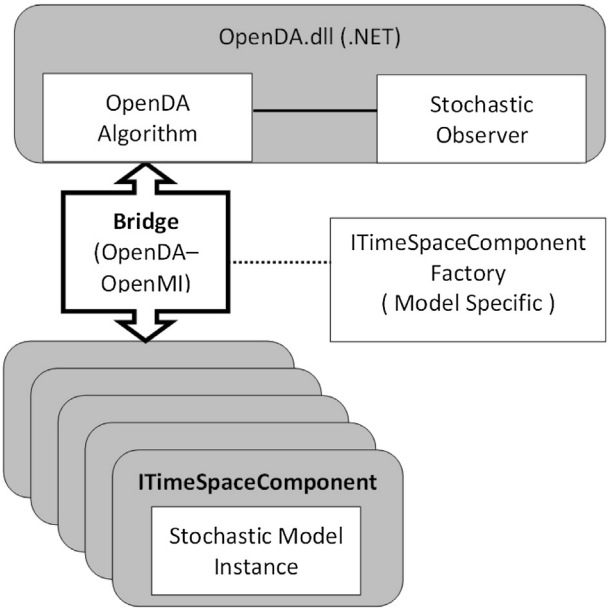


Fig. 6. OpenDA–OpenMI Framework. Both *OpenDA algorithm* and *Stochastic observer* components (within the top shaded area) are identical to the OpenDA components previously described and shown in Fig. 4, except that the byte code has been compiled to .NET using IKVM. The *ITimeSpaceComponentFactory* spawns *ITimeSpaceComponent* instances of the model using the OpenMI standard. Afterward, communication, control and data exchange between the model instances and the *OpenDA algorithm* is done via the *Bridge (OpenDA–OpenMI)*.

5.1.3. OpenDA in .NET

As OpenDA is compiled in Java whereas OpenMI is defined in the .NET environment, IKVM is used to link the two frameworks (IKVM.NET). IKVM is an implementation of Java for the Microsoft .NET Framework with a Java Virtual Machine implementation in .NET. The open source IKVM includes a 'IKVMC' compiler tool that takes compiled java bytecode and converts it to .NET Common Intermediate Language (CIL). Thus a fully debuggable Dynamic-link library (.dll) of OpenDA is produced.

5.1.4. OpenDA–OpenMI bridge

The OpenDA–OpenMI bridge compiled in .NET 4.0 was designed to be flexible enough to link any OpenMI 2.0 compliant model with the IKVM converted OpenDA. The bridge converts method calls and data structures from OpenDA to/from *ITimeSpaceComponent* function calls and *ExchangeItems* in .NET. In this bridge implementation, very little additional programming is necessary for communication between OpenMI and OpenDA (see Section 5.2 for details). The Bridge is broadly grouped into four layers (see Fig. 8):

1. OpenDA .NET to/from Java (OpenDaDotnet2Java). Implements some of the interfaces expected by OpenDA in .NET and is the entrance point for OpenDA.
2. Functions and conversions from Java to .NET (UsingOpenDAInDotNet). A Software Development Kit (SDK) with conversion methods from Java standards to .NET.
3. OpenMI to OpenDA (OpenMIModelsUsingOpenDA). The bridge between the OpenDA .NET compliant calls to OpenMI.
4. Model Specific Implementation. This layer is model specific and implements the interfaces defined in Section 5.2.

The bridge is responsible for the five primary tasks listed in the Fig. 7.

1. **Initialize.** This is model specific where the user must create a model Factory (as described in Section 5.1.1) that returns fully initialized *ITimeSpaceComponent* instances. A string of arguments is passed to the model factory which could be blank, or could contain whatever further information is needed for initialization.
2. **Time Step.** OpenDA calls Compute(EndTime) when it requires the model instances to be propagated to particular time. The EndTime is most often the 'time of next available observation', or EndTime could be the final assimilation time. The bridge converts the Java *DateTime* to an OpenMI *ITime*. The logic required to time step an *ITimeSpaceComponent* to the given time is implemented by the bridge (so the user does not need to do anything here).
3. **Get Data.** Two different get data functions are called by OpenDA. *ExchangeItem.getValues(ID)* gets an array of data (which is used to form the state vector in OpenDA) of a variable identified by a string ID. Whereas the *ExchangeItem.getObservedValues(ObservationDescription)* gets an array of data from the defined variable at the observation locations. In this case, the *ObservationDescription* is Metadata containing information regarding the observation string ID, time, and locations defined by x, y, z coordinates. The bridge converts the information passed from Java to .NET, and performs the appropriate OpenMI calls to retrieve the information requested from OpenDA. The data from

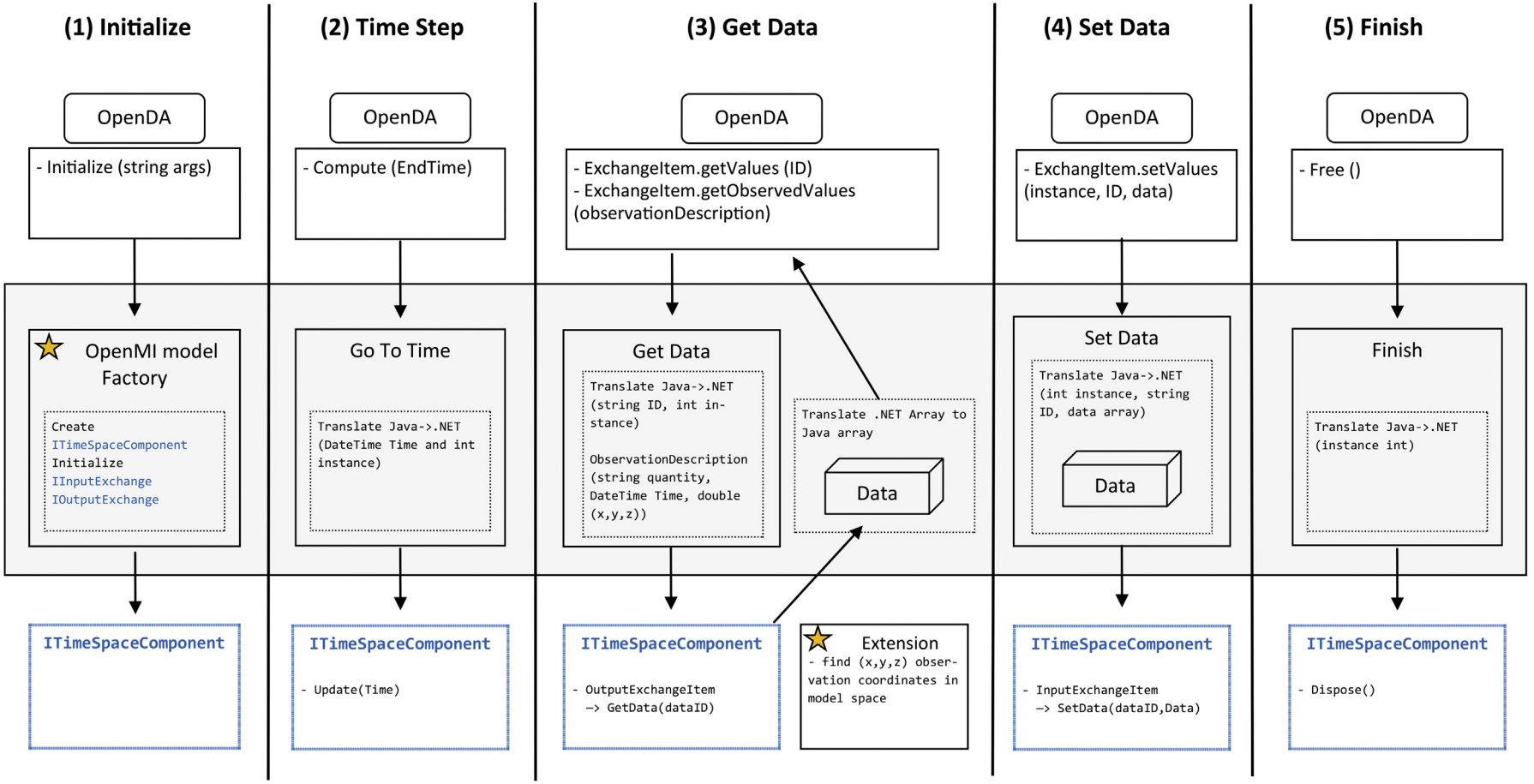


Fig. 7. A flow diagram of the framework's operations. The upper portion of the diagram represents the OpenDA domain, the bottom part is the OpenMI domain, and the middle region in gray is the OpenDA–OpenMI framework linking them together. The framework is responsible for five main functions as listed at the top. For each, the calls and function data must be translated from Java to .NET and back. The two boxes with stars are the only model specific components to be implemented when adapting a new model to this framework.

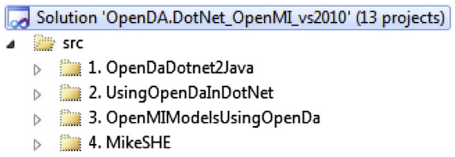


Fig. 8. A view of the framework from Visual Studios 2010 where the projects are broadly grouped into four layers.

OpenMI is then converted from a .NET array to a Java array. The OpenDA–OpenMI framework provides all the logic for the get data operations, except for the mapping from observation space to model space. Specifically, where the observation *x*, *y*, *z* coordinates are located within the model is user specific and must be programmed.

4. **Set Data.** When OpenDA performs a filter update, a new state vector is set to all the models. OpenDA converts the state vector to *ExchangeItems* of values to be set for each variable. The data array and variable string ID is passed from OpenDA to OpenMI via the bridge (that converts it from Java to .NET), and the new values are set to each corresponding variable in OpenMI. All the logic for setting values is handled within the bridge, and no additional programming is necessary.
5. **Finish.** When OpenDA finishes, all the model instances are instructed to finish. A function call is passed via the bridge to call OpenMI Dispose().

5.2. Model specific implementation

The OpenDA–OpenMI framework is as general and versatile as possible. Some model specific implementation is, however, unavoidable but it is very minimal and grants the user the flexibility to configure the system optimally for their needs. The three methods to be implemented (in .NET) are described in Fig. 9 which correspond the stared boxes in Fig. 7. Two of the three methods deal with the model factory and the initialization of the *ITimeSpaceComponent*, whereas the third function maps the observation space to the model space. Corresponding to Fig. 9 the following methods are required: (1) A void *Initialize* method for passing information to the framework. (2) A model instance factory that returns an initialized *ITimeSpaceComponent* every time the method is called. The user can, when returning an *ITimeSpaceComponent*, configure their own system to store the model results on disk or just in memory, run the models over different computers, add diagnostic or backup capabilities for the assimilation system. (3) A **H** measurement operator (Equation (2)) must be implemented to translate the model variables to observation space.

A step-by-step procedure to get an OpenMI compliant model to perform assimilation with this framework can be summed up as follows:

1. Download the OpenDA–OpenMI Framework which includes all the libraries (OpenMI, OpenDA,.., etc) necessary.
2. Implement the three functions described in Fig. 9.
3. Format the observation data to be compliant with one of the OpenDA formats (NOOS, CSV, or SQL database). Information and examples for this task are included in OpenDA.
4. The OpenDA configuration is XML based, with information for the 'algorithm', 'stochModel', 'results', and 'stochObserver'. The XML schema is described in the OpenDA documentation. One OpenDA–OpenMI framework specific addition to the XML configuration is the addition of the model factory reference to the class name containing the user implemented model factory.
5. Model uncertainty can either be defined within the Model Factory, where there might be *N* ensemble models pre-perturbed (with different parameters, forcing, or initial conditions), or OpenDA can add a noise model via the XML schema.
6. Execute by creating a model Factory instance and calling ApplicationRunnerSingleThreaded.

An example is provided with the OpenDA–OpenMI framework to help guide the user through the steps.

6. Test case

A twin-test (also known as synthetic) data assimilation experiment was performed using the OpenMI–OpenDA framework. A river catchment was modeled in MIKE SHE, an OpenMI compliant distributed hydrological model. The ensemble was created

```
/// <summary>
/// Initializes the ITimeSpaceComponent Class if needs be.
/// </summary>
/// <param name="workingDirPath">Nothing is passed here (not used).</param>
/// <param name="args">One argument. Full file name to the OpenMI model</param>
public void Initialize(string workingDirPath, params string[] args)

/// <summary>
/// Able to generate a new and unique OpenMI ITimeSpaceComponent instance
/// every time it is called.
/// Returns a new OpenMI model instance.
/// </summary>
public ITimeSpaceComponent CreateInstance()

/// <summary>
/// The measurment operator (H) that relates the model state to the observation.
/// The IObservationDescriptions contains information about the observations such as latitude,
/// longitue or other observation information.
/// For each observation in the IObservationDescriptions, this function returns the corresponding model result.
/// Here the user has the flexibility to derive a model relating model results to observations.
/// </summary>
/// <param name="IObservationDescriptions">OpenDA style class with a description of the observation.</param>
public double[] getObservedValues(OpenDA.DotNet.Interfaces.IObservationDescriptions observationDescriptions)
```

Fig. 9. Three model specific classes must be implemented for the OpenMI–OpenDA framework.

representing the uncertainty in the model and input by perturbing model parameters and forcing data. One of those perturbed realizations was considered to be the "truth" and hydraulic head at 35 observation wells were assimilated. The assimilation and forecast performance of the ensemble average was compared to the true model. The test case detailed below showcases the capability of the OpenMI–OpenDA framework.

6.1. Karup catchment

The Karup River with its approximately 20 tributaries drain an area of 440 km^2 in western Denmark. See Fig. 10. The catchment has a gently sloping topography, ranging between 20 and 100 m above sea level. The geology is relatively homogeneous with highly permeable sand and gravel deposits and small lenses of moraine clay. The aquifer is mainly unconfined and varies in thickness from about 10 m at the western and central part to more than 90 m at the upstream eastern water divide. The depth of the unsaturated zone varies from 25 m at the eastern water divide to less than 1 m in the wetland areas along the river. The land use consists of agriculture (67%), forest (18%), heath (10%), and wetland areas (5%). The catchment has been featured in a number of studies and modeled in MIKE SHE (Madsen, 2003; Refsgaard, 1997a; Styczen and Storm, 1993; Refsgaard et al., 1999; Blasone et al., 2008). The meteorological and hydrological data are described in detail in past studies (Refsgaard, 1997a; Madsen, 2003; Blasone et al., 2008). The data

Table 2
Parameters subject to perturbation for ensemble generation. *The vertical hydraulic conductivity (Kv) was tied to the horizontal conductivity (Kh) by the equation Kv = 0.1 × Kh.

Process	Variable	Mean	Standard deviation	Note
Overland flow (m$^{1/3}$/s)	Manning – n	5	0.2	Varied
Saturated zone conductivity (m/s)	Horizontal – Meltwater sand	5×10^{-4}	1×10^{-5}	Varied
	Vertical – Meltwater sand			Tied*
	Horizontal – Clay	8×10^{-7}	3×10^{-7}	Varied
	Vertical – Clay			Tied*
	Horizontal – Quartz sand	9×10^{-4}	3×10^{-5}	Varied
	Vertical – Quartz sand			Tied*
	Horizontal – Mica sand	5×10^{-5}	2×10^{-6}	Varied
	Vertical – Mica sand			Tied*
	Horizontal – Mica clay/silt	1×10^{-8}	5×10^{-10}	Varied
	Vertical – Mica clay/silt			Tied*
	Horizontal – Limestone	7.4×10^{-5}	3×10^{-6}	Varied
	Vertical – Limestone			Tied*
Unsaturated zone (–)	van Genuchten, n (JB-1)	1.445	0.07	Varied
	van Genuchten, n (JB-2)	1.382	0.07	Varied
	van Genuchten, n (JB-3)	1.337	0.07	Varied
	van Genuchten, n (JB-4)	1.304	0.07	Varied
	van Genuchten, n (JB-5)	1.251	0.07	Varied

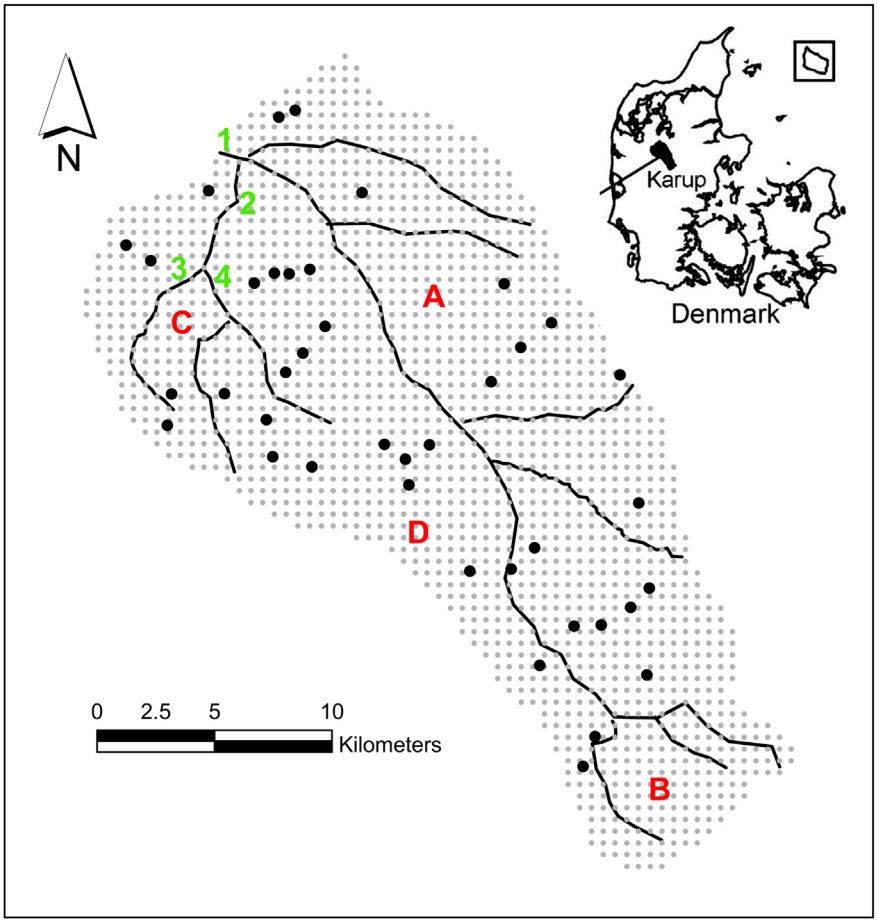


Fig. 10. Karup catchment in Denmark. The 35 groundwater elevation wells that record data every 15 days and assimilated into the model are marked by black circles (•). Validation data for hydraulic head are marked by the red letters (**A**, **B**, **C**, **D**), and discharge validation marked by green numbers (**1**, **2**, **3**, **4**). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

includes rainfall measurements from nine stations (daily), runoff at the river outlet and at three internal subcatchments (daily), groundwater elevation data from 35 wells (every 15 days) and temperature (daily). The data used in this study cover the period between the beginning of 1970 to the end of 1978.

6.2. MIKE SHE

The MIKE SHE modeling system (Graham and Butts, 2006) is used to simulate the major catchment processes. MIKE SHE simulates the interactions between the surface (evaporation, overland flow and river runoff), the unsaturated zone (evapotranspiration and infiltration to the aquifer), and the saturated zone (groundwater flow and recharge). MIKE SHE includes different model descriptions for describing the different processes and their interactions. In the current setup, distributed process descriptions are used for all components with a computational grid of 1×1 km. The overland flow is simulated with the 2D Saint–Venant equations with uniform Manning number over the catchment; see Table 2. The river system is modeled in MIKE 11 using a Muskingum routing scheme and a uniform leakage coefficient regulating the river–aquifer interaction. Distributed evapotranspiration (Kristensen and Jensen, 1975) is modeled using four vegetation classes with different Leaf Area Index (LAI), root depth, and evapotranspiration parameters. Water in the unsaturated zone was simulated with the Richards' equation with 91 soil layers. Soil types are distributed using seven soil types which are described using van Genuchten parameters (van Genuchten, 1980). The saturated zone uses six different geological soils types with different horizontal and vertical conductivities, specific yield and specific storage. See Table 2 for some parameter values.

6.3. Model uncertainty

Correctly describing model uncertainty is a vital aspect of data assimilation. The uncertainty in the Karup catchment model has been analyzed using generalized likelihood uncertainty estimation (GLUE) in a previous study (Blasone et al., 2008). For ensemble filtering, an ensemble of possible stochastic realizations must be generated. Different types of uncertainty were taken into account when generating the ensemble of model realizations. Perturbations were added to the forcing time series for precipitation and reference evapotranspiration using a Gaussian error model at every time step with a relative standard deviation of 0.25. Model parameters were perturbed using a Gaussian model with mean and standard deviation as outlined in Table 2. The parameters chosen for perturbation was based on previously published MIKE SHE sensitivity and uncertainty analyses (Stisen et al., 2011; Blasone et al., 2008; Ridler et al., 2012). The perturbations were designed to capture the uncertainty of the overland flow (using Manning n), the saturated zone (using hydraulic conductivities), and the unsaturated zone (with van Genuchten n). The initial conditions for all realizations were different from each other before assimilation by spinning-up the model for 3 years with the different parameters and forcing data. As seen by the confidence intervals in Fig. 11, the spread during the spin-up period reflects the uncertainty in the system.

6.3.1. Performance measure

The assimilated model is evaluated using a number of performance criteria (Bennett et al., 2013). In this twin-test experiment, the aim is to correct the ensemble mean such that the corrected state approaches the twin's state. As well as evaluating the performance visually by comparing time series, quantitative evaluation is based on both the Root Mean Squared Error (RMSE) and the coefficient of determination R^2. A 95% confidence interval is provided to evaluate the assimilation performance and the uncertainty estimate.

6.4. Experimental protocol

In this synthetic twin-test experiment with 300 ensemble members, hydraulic head was assimilated in the Karup catchment. Observations for assimilation were drawn from the twin model taken from 35 locations (as shown in Fig. 10) every 15 days. The observations were subsequently perturbed by adding Gaussian noise with a standard deviation of 10 cm. The ensembles members were initialized with the uncertainty statistics described in Section 6.3 and Table 2. All model instances were spun-up for three years so as to have all different initial conditions, followed by three years of assimilation and three years of forecast. Various assimilation runs were carried out to test filters and localization distances, but only the results for EnKF without localization or inflation are shown.

6.5. Results

6.5.1. Hydraulic head

Four time series of hydraulic head from points situated away from an assimilation observation are shown in Fig. 11. The uncertainty in the system is measured from the spread of model realizations and is shown by the 95% confidence interval in gray. During the assimilation period, the filter is quick to converge to the true solution even though there were no observations being assimilated at these validation points. The *assimilated* model in green closely follows the *true* model for all the four cases shown in the figure. During the assimilation period there is a large reduction in the ensemble spread compared to the spin-up and forecast periods.

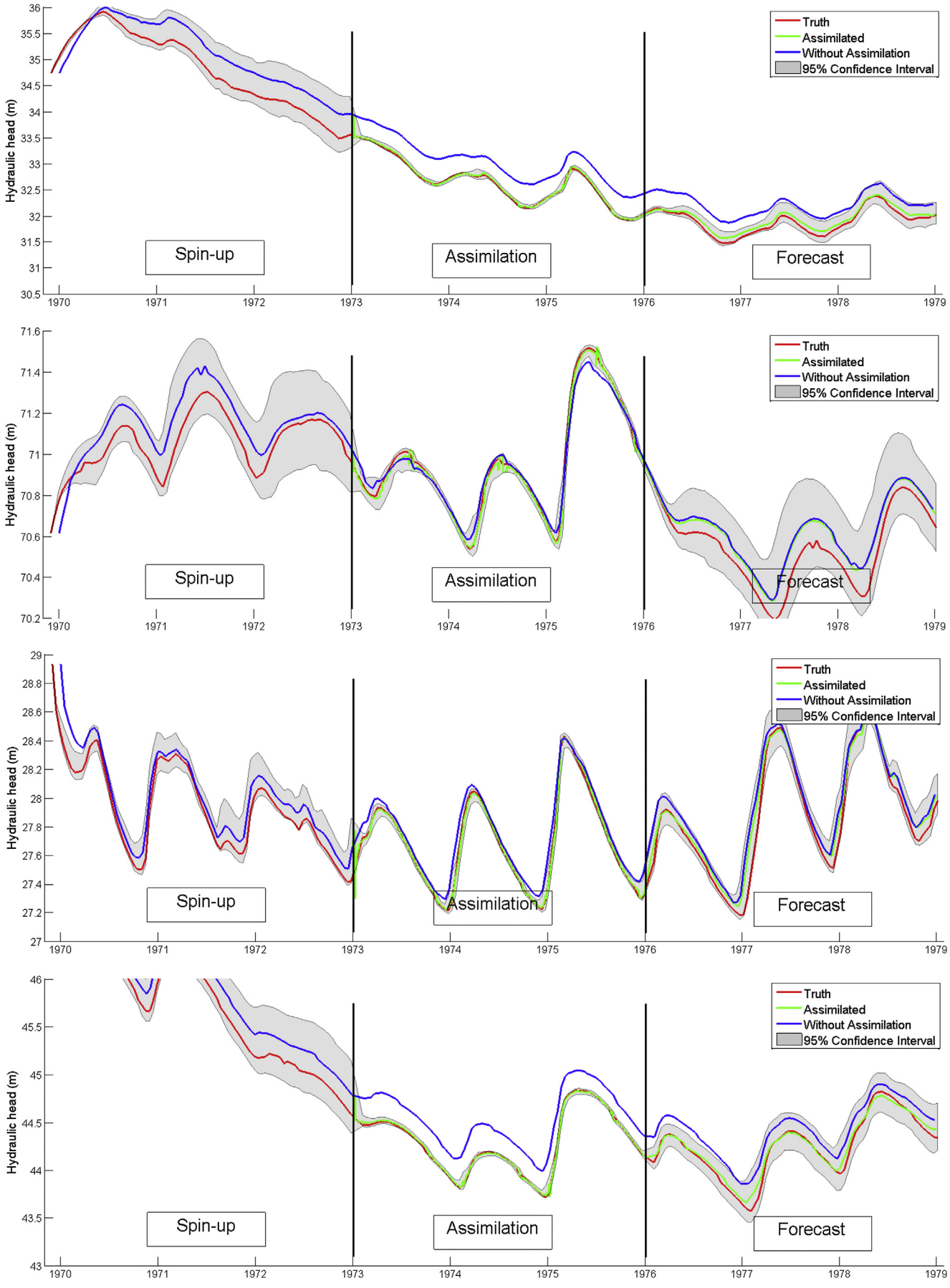
The four validation points represent different points in the catchment. Validation well **A** is bounded by four observation wells to the east with a fixed river head to the north and west and is characterized by having a thicker unsaturated zone. The absolute head difference at **A** between simulated and observed is greatest, and consequently the correction with the filter is pronounced. During the forecast period, the advantage of using EnKF is clear. Three years after assimilation has ended, the *assimilated* model still closely follows the *true* model and the spread in the confidence interval slowly increases over time from 1976 to 1979.

Validation point **B** (second time series in Fig. 10). Before assimilation, the *assimilated* and *truth* models are well within the confidence interval. Again, assimilation improved the head estimates but gains in this case were not maintained during the forecast period. The difference between assimilated and open-loop is only in the order of 20 cm at point **B**.

Both points **C** and **D** (third and fourth time series in Fig. 10) are similar in their results. The uncertainty is reduced during the assimilation period during which there are noticeable improvements in head estimates. However at **C**, improvement during the forecast period is not as pronounced.

To evaluate the overall performance of the assimilation, distributed maps of hydraulic head Root Mean Squared Error (RMSE) are shown in Fig. 12.

The top two panels are the RMSE during the assimilation period and the bottom two are during the forecast period where each period lasts three years. There is a clear reduction in RMSE from the left to right panels with the assimilation filter applied. Areas with high RMSE (in the order of 90 cm) in the north-east were improved with the EnKF leading to a RMSE in the order of 10 cm for both the assimilation period and the forecast period. This figure clearly illustrates the strengths and potential of data assimilation.


Truth
Assimilated
Without Assimilation
95% Confidence Interval
Hydraulic head (m)
Spin-up
Assimilation
Forecast
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979

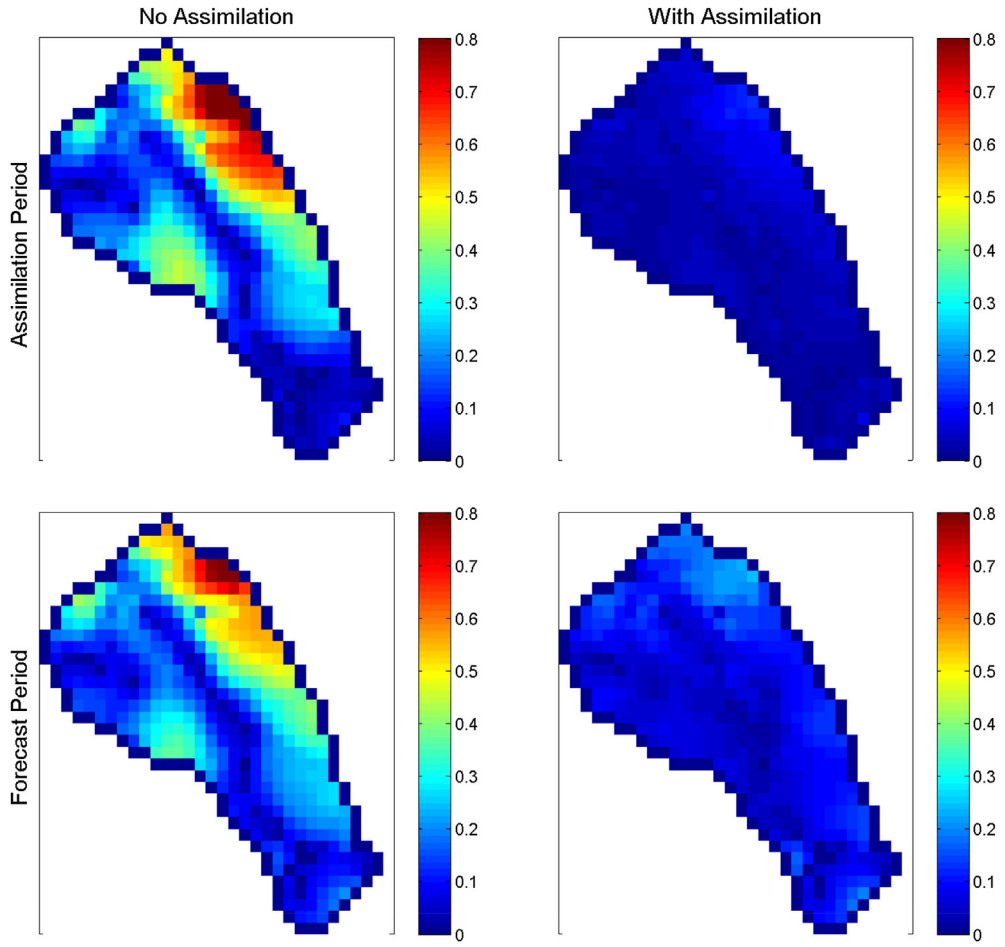


Fig. 12. Hydraulic head distributed RMSE measured in meters. The RMSE without assimilation (left) is compared to assimilated results (right) averaged over the three year assimilation period (top) and forecast period (bottom). An EnKF filter with 300 ensemble members was used for assimilation.

6.5.2. River discharge

The effect of hydraulic head assimilation on river discharge (which was not itself assimilated or part of the state vector) is analyzed here. Data from four gauging stations along the river measuring daily discharge are used to check the effect of head assimilation. Discharge results for the main reach (at station **1** Fig. 10) are shown in Fig. 13. The uncertainty is marked by the gray area measuring the 95% confidence interval. Uncertainty is reduced during the assimilation period and the bias (overestimation) is reduced with the application of the EnKF. Error statistics for the discharge during the assimilation period goes from a RMSE of 0.273 m^3/s and coefficient of determination, R^2 of 0.818 to a RMSE of 0.0737 m^3/s and R^2 of 0.985 with assimilation. In this example, there was a clear benefit to river discharge by correcting hydraulic head due to the significant ground-water river interaction. Similar results for discharge stations **2**, **3** and **4** were observed (graphs not shown). In these three stations, again the bias was reduced during the assimilation period and improvements in forecast were noted. At station **2** RMSE went from 0.056 to 0.0168 m^3/s while R^2 went from 0.908 to 0.989, at station **3** RMSE went from 0.0445 to 0.0101 m^3/s while R^2 went from 0.869 to 0.989, and finally at station **4** RMSE went from 0.0119 to 0.0077 m^3/s while R^2 went from 0.962 to 0.983.

7. Discussion and conclusion

More accurate and more reliable hydrological forecasts can reduce the impacts from water-related hazards (e.g. floods) and provide the basis for more effective water resources management, emergency management, planning of agricultural production, and environmental monitoring. Recent advances in satellite and in-situ monitoring, and faster computing facilities are opening up new opportunities to produce better forecasts. But it remains a key challenge to incorporate all this new information in a forecast model. Data assimilation is one means to optimally merge model forecasts with observations (otherwise unused) taking into account both model and observational uncertainty. Several filters exist (e.g. EnKF, EnSR, PF) each with their own characteristics, numerical techniques (e.g. localization, inflation, parameter dampening) and noise models (ARMA, spatial-temporal rainfall perturbation). With OpenDA, users can experiment with data assimilation methods without the need for extensive programming.

Fig. 11. Results from a 300 ensemble member EnKF experiment. The time series from four different head validation points (Top to bottom: **A**, **B**, **C**, **D** corresponding to the points in Fig. 10) are all situated away from an assimilated observation. The time series consists of three periods each three years long: spin-up, assimilation, and forecast. The open-loop run without assimilation is in blue, the assimilated in green, the truth in red, and the 95% confidence interval in gray. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

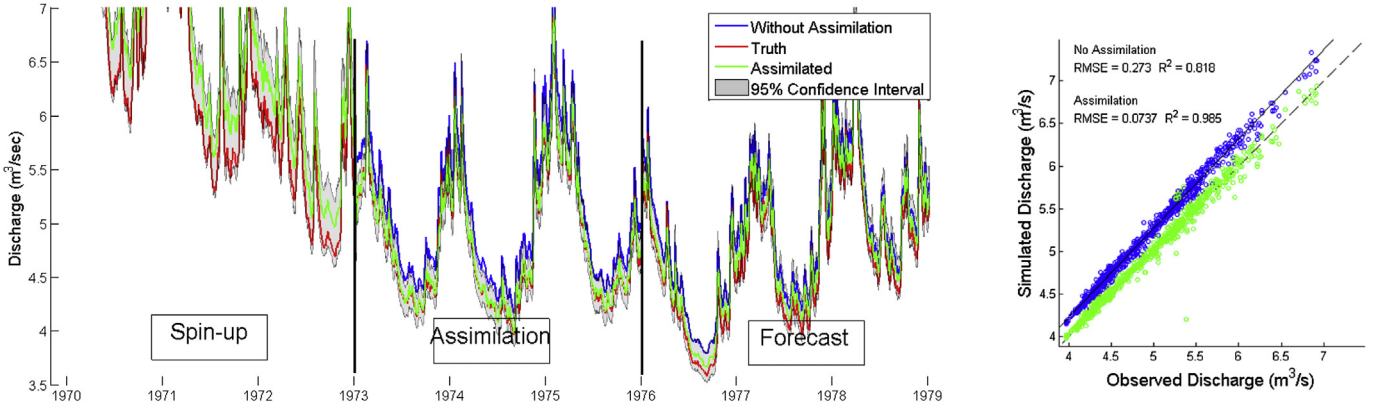


Fig. 13. River discharges from validation point (**1**) corresponding to the point on the map in Fig. 10. The time series on the left shows daily discharge for the 3 year spin-up period, 3 year assimilation period and 3 year forecast period. The open-loop run without assimilation is in blue, the assimilated optimal in green, the truth in red, and the 95% confidence interval in gray. The scatter plot on the right compares the assimilated results (green) with the open-loop results (blue) during the three year assimilation period with corresponding statistics. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

OpenDA was designed with efficiency in mind based on its predecessor COSTA. COSTA was tested on a shallow water model with the RRSQRT Kalman Filter and the time penalty was compared with a dedicated implementation (van Velzen and Verlaan, 2007). In the worst case scenario, there was an overhead in the order of 10%. In this OpenDA–OpenMI framework solution, two additional penalties are included: the OpenMI layering and the Java to .NET bridge. As far as OpenMI is concerned, one study by Castronova et al. (2013) showed that in their case the run-time data exchange was not significant (Castronova et al., 2013). Another study (Shrestha et al., 2013) found that OpenMI lead to an overhead of 14 ± 1%. With the Java to .NET bridge, the penalty with this solution is dependent on the amount of data being transferred back and forth. In this MIKE SHE experiment with 300 ensemble members, the computational overhead using this framework was found to be minimal in comparison to the cost of propagating the ensemble members. A larger model with more observation points (than the 35 used in this experiment) would likely lead to greater overhead.

In this study, a data assimilation framework was developed and tested that grants OpenMI compliant models access to a robust and flexible data assimilation library. Popular hydrological models that simulate rivers, catchments, groundwater, land surface and many other processes, can now take advantage of observation data for model optimization and uncertainty analysis. The framework is designed to grant a user access to numerous complex assimilation algorithms, noise models, and distance based regularization techniques with minimal amount of programming. The framework applies to OpenMI compliant models only and has been tested in a Windows environment. The test involved assimilating hydraulic head observations in a MIKE SHE model of the Karup catchment in Denmark. Using an EnKF with 300 ensemble members, the filter improved head simulations over the whole catchment as well as forecasts up to 3 years ahead. Because the river system is partially fed by groundwater recharge, head assimilated in the integrated model also lead to improved discharge estimates.

Acknowledgment

This work was carried out with the support of the Danish Council for Strategic Research as part of the project "HydroCast – Hydrological Forecasting and Data Assimilation", Contract No. 11-116880 (http://hydrocast.dhigroup.com/).

References

Anderson, J.L., 2001. An ensemble adjustment Kalman filter for data assimilation. Mon. Weather Rev. 129, 2884–2903.
Anderson, J., Hoar, T., Raeder, K., Liu, H., Collins, N., Torn, R., Avellano, A., 2009. The data assimilation research testbed: a community facility. Bull. Am. Meteorol. Soc. 90, 1283–1296.
Bennett, N.D., Croke, B.F., Guariso, G., Guillaume, J.H., Hamilton, S.H., Jakeman, A.J., Marsili-Libelli, S., Newham, L.T., Norton, J.P., Perrin, C., et al., 2013. Characterising performance of environmental models. Environ. Model. Softw. 40, 1–20.
Blasone, R.S., Vrugt, J.A., Madsen, H., Rosbjerg, D., Robinson, B.A., Zyvoloski, G.A., 2008. Generalized likelihood uncertainty estimation (GLUE) using adaptive Markov Chain Monte Carlo sampling. Adv. Water Resour. 31, 630–648.
Camporese, M., Paniconi, C., Putti, M., Salandin, P., 2009. Ensemble Kalman filter data assimilation for a process-based catchment scale model of surface and subsurface flow. Water Resour. Res. 45, W10421.
Castronova, A.M., Goodall, J.L., Ercan, M.B., 2013. Integrated modeling within a hydrologic information system: an openmi based approach. Environ. Model. Softw. 39, 263–273.
Donchyts, G., Hummel, S., Vaneçek, S., Groos, J., Harper, A., Knapen, R., Gregersen, J., Schade, P., Antonello, A., Gijsbers, P., 2010. OpenMI 2.0 whats new. In: International Congress on Environmental Modelling and Software, July, pp. 5–8.
Evensen, G., 2003. The ensemble Kalman filter: theoretical formulation and practical implementation. Ocean. Dyn. 53, 343–367.
Gijsbers, P., Hummel, S., Vaneçek, S., Groos, J., Harper, A., Knapen, R., Gregersen, J., Schade, P., Antonello, A., Donchyts, G., 2010. From OpenMI 1.4 to 2.0. In: International Congress on Environmental Modelling and Software.
Graham, D.N., Butts, M.B., 2006. Flexible integrated watershed modeling with MIKE SHE. Watershed Models, pp. 245–272.
Gregersen, J.B., Gijsbers, P.J.A., Westen, S.J.P., Blind, M., et al., 2005. OpenMI: the essential concepts and their implications for legacy software. Adv. Geosci. 4, 37–44.
Gregersen, J.B., Gijsbers, P.J.A., Westen, S.J.P., 2007. OpenMI: open modelling interface. J. Hydroinform. 9, 175–191.
Hamill, T.M., Whitaker, J.S., Snyder, C., 2001. Distance-dependent filtering of background error covariance estimates in an ensemble Kalman filter. Mon. Weather Rev. 129, 2776–2790.
Heemink, A.W., Hanea, R.G., Sumihar, J., Roest, M., Velzen, N., Verlaan, M., 2010. Data assimilation algorithms for numerical models. Adv. Comput. Methods Sci. Eng., 107–142.
Hendricks Franssen, H.J., Kinzelbach, W., 2008. Real-time groundwater flow modeling with the Ensemble Kalman Filter: joint estimation of states and parameters and the filter inbreeding problem. Water Resour. Res. 44, 1–21.
Hendricks Franssen, H.J., Kaiser, H.P., Kuhlmann, U., Bauser, G., Stauffer, F., Müller, R., Kinzelbach, W., 2011. Operational real-time modeling with ensemble Kalman filter of variably saturated subsurface flow including stream-aquifer interaction and parameter updating. Water Resour. Res. 47, W02532.
IKVM.NET. IKVM.NET Home Page.
Knapen, R., Janssen, S., Roosenschoon, O., Verweij, P., De Winter, W., Uiterwijk, M., Wien, J.E., 2013. Evaluating openmi as a model integration platform across disciplines. Environ. Model. Softw. 39, 274–282.
Komma, J., Blöschl, G., Reszler, C., 2008. Soil moisture updating by Ensemble Kalman Filtering in real-time flood forecasting. J. Hydrol. 357, 228–242.
Kristensen, K.J., Jensen, S.E., 1975. A model for estimating actual evapotranspiration from potential evapotranspiration. Nord. Hydrol. 6, 170–188.

Liu, Y., Weerts, A.H., Clark, M., Hendricks Franssen, H.J., Kumar, S., Moradkhani, H., Seo, D.J.J., Schwanenberg, D., Smith, P., van Dijk, A.I.J.M., van Velzen, N., He, M., Lee, H., Noh, S.J., Rakovec, O., Restrepo, P., 2012. Advancing data assimilation in operational hydrologic forecasting: progresses, challenges, and emerging opportunities. Hydrol. Earth Syst. Sci. 16, 3863–3887.

Madsen, H., 2003. Parameter estimation in distributed hydrological catchment modelling using automatic calibration with multiple objectives. Adv. Water Resour. 26, 205–216.

Madsen, H., Skotner, C., 2005. Adaptive state updating in real-time river flow forecasting a combined filtering and error forecasting procedure. J. Hydrol. 308, 302–312.

Moore, R.V., Tindall, C.I., 2005. An overview of the open modelling interface and environment (the OpenMI). Environ. Sci. Policy 8, 279–286.

Moradkhani, H., 2008. Hydrologic remote sensing and land surface data assimilation. Sensors 8, 2986–3004.

Moradkhani, H., Hsu, K.L., Gupta, H., Sorooshian, S., 2005a. Uncertainty assessment of hydrologic model states and parameters: sequential data assimilation using the particle filter. Water Resour. Res. 41.

Moradkhani, H., Sorooshian, S., Gupta, H.V., Houser, P.R., 2005b. Dual state–parameter estimation of hydrological models using ensemble Kalman filter. Adv. Water Resour. 28, 135–147.

Nerger, L., Hiller, W., 2013. Software for ensemble-based data assimilation systems implementation strategies and scalability. Comput. Geosci. 55, 110–118.

OpenDA. The OpenDA Data-assimilation Toolbox.

OpenMI. OpenMI Association.

Refsgaard, J.C., 1997a. Parameterisation, calibration and validation of distributed hydrological models. J. Hydrol. 198, 69–97.

Refsgaard, J.C., 1997b. Validation and intercomparison of different updating procedures for real-time forecasting. Nord. Hydrol. 28, 65–84.

Refsgaard, J.C., Thorsen, M., Jensen, J.B., Kleeschulte, S., Hansen, S., 1999. Large scale modelling of groundwater contamination from nitrate leaching. J. Hydrol. 221, 117–140.

Ridler, M.E., Sandholt, I., Butts, M., Lerer, S., Mougin, E., Timouk, F., Kergoat, L., Madsen, H., 2012. Calibrating a soilvegetationatmosphere transfer model with remote sensing estimates of surface temperature and soil surface moisture in a semi arid environment. J. Hydrol. 436–437, 1–12.

Sakov, P., Bertino, L., 2011. Relation between two common localisation methods for the EnKF. Comput. Geosci. 15, 225–237.

Sakov, P., Oke, P.R., 2008. Implications of the form of the ensemble transformation in the ensemble square root filters. Mon. Weather Rev. 136, 1042–1053.

Shrestha, N.K., Leta, O.T., De Fraine, B., van Griensven, A., Bauwens, W., 2013. Openmi-based integrated sediment transport modelling of the river Zenne, Belgium. Environ. Model. Softw. 47, 193–206.

Stisen, S., McCabe, M.F., Refsgaard, J.C., Lerer, S., Butts, M.B., 2011. Model parameter analysis using remotely sensed pattern information in a multi-constraint framework. J. Hydrol. 409, 337–349.

Styczen, M., Storm, B.r., 1993. Modelling of N-movements on catchment scale-a tool for analysis and decision making. Fertil. Res. 36, 7–17.

Tippett, M.K., Anderson, J.L., Bishop, C.H., Hamill, T.M., Whitaker, J.S., 2003. Ensemble square root filters*. Mon. Weather Rev. 131, 1485–1490.

van Genuchten, M.T., 1980. A closed-form equation for predicting the hydraulic conductivity of unsaturated soils. Soil Sci. Soc. Am. J. 44, 892–898.

Van Leeuwen, P.J., 2009. Particle filtering in geophysical systems. Mon. Weather Rev. 137, 4089–4114.

van Velzen, N., Segers, a.J., 2010. A problem-solving environment for data assimilation in air quality modelling. Environ. Model. Softw. 25, 277–288.

van Velzen, N., Verlaan, M., 2007. COSTA a problem solving environment for data assimilation applied for hydrodynamical modelling. Meteorol. Z. 16, 777–793.

van Velzen, N., Verlaan, M., Bos, E., 2013. The OpenDA Association Annual Report 2012.

Weerts, A.H., El Serafy, G.Y., Hummel, S., Dhondia, J., Gerritsen, H., 2010. Application of generic data assimilation tools (datools) for flood forecasting purposes. Comput. Geosci. 36, 453–463.

Whitaker, J.S., Hamill, T.M., 2002. Ensemble data assimilation without perturbed observations. Mon. Weather Rev. 130, 1913–1924.