Data assimilation with OpenDA

OpenDA course
November 2012

Alja Vrieling,
Nils van Velzen, Martin Verlaan, Stef Hummel,
Julius Sumihar, Albrecht Weerts, Ghada El Serafy, Herman Gerritsen
Outline

- Data assimilation and model calibration in one sheet
- What is OpenDA
- OpenDA architecture
- Coupling models into OpenDA
- Example of Calibration
- Example of Data assimilation
- OpenDA and Parallel computing
- Summary
Data assimilation and model calibration in one sheet

- (real time) data assimilation
- model calibration
- impact of observations
- reconstruction of sources

Data assimilation with OpenDA
OpenDA Toolbox

Motivation:
the algorithms are model/observation independent BUT
- DA methods work on top of models → serious changes to model code.
- Spaghetti code
- Difficult to change algorithm
- Maintenance problems
- Difficult to test algorithms
- Difficult/impossible to reuse code
- Expensive!
What is OpenDA

- Content:
  - Set of interfaces that define interactions between components
  - Library of data-assimilation algorithms
  - DA philosophy
  - Building blocks only need to be implemented once
What is OpenDA

- Open source (LGPL)
- Why OpenDA?
  - More efficient than development for each application
  - Shared knowledge between applications
  - Development of algorithms with e.g. universities
  - Easier to test, which should result in fewer bugs
  - Optimized building blocks
  - Development template
OpenDA Website

- www.openda.org
- Downloads
  - Users
  - Developers
- Documentation
- Wrappers for models
- OpenDA association

TU Delft
Deltare
VORtech bv

Data assimilation with OpenDA
OpenDA Main Application

Single program for

- Editing input files
- Running simulations and calibrations
- All models and observation sources
- GUI and command line
- Sequential and parallel
OpenDA architecture
Object Oriented Approach
Models in OpenDA

- Formal form of a model
  \[
  \frac{dx}{dt} = M(x(t), u(t), p, w(t))
  \]
- State of model instance \( x(t), u(t), p, w(t) \)
  \( x \): state vector, \( u \): forcings, \( p \): parameters, \( w \): stochastic part
- Object oriented concepts:
  - Data encapsulation; state cannot be directly changed only through the methods like:
    GetState, AxpyonState, Compute...
  - Multiple instances of model
  - Algorithm has no knowledge on model internals
Models in OpenDA

- Models in Data Assimilation context are stochastic!
- Normal dynamic model (codes) are not
Models in OpenDA

- A model wrapper is needed for each model:
  - Black box
  - Java
  - C/Fortran
  - C#
- Testing OpenDA models
  - Java
  - Matlab
  - C#
Models in OpenDA

OpenDA

Black Box wrapper

MyStochModel

MyWrapper

MyModel

MyModel

MyStochModel

TheirModel.exe

java

C/Fortran

other

own

3rd party

Data assimilation with OpenDA
Data assimilation with OpenDA
Example of model calibration
SWAN 3rd generation wave model

Wave breaking and interaction over a bar

\[ S_{\text{tot}} = \text{wind input} + \text{non-linear interactions (quadruplets & triads)} + \text{whitecapping} + \text{bottom friction} + \text{depth induced wave breaking} \]
Example of model calibration
SWAN 3\textsuperscript{rd} generation wave model
Example of model calibration
SWAN 3\textsuperscript{rd} generation wave model

Data assimilation with OpenDA
Example of data assimilation
Operational storm surge forecasting
Example of data assimilation
Operational storm surge forecasting

Predicted and Observed Waterlevel at CADZD
Period: 25-Apr-2002 - 30-Apr-2002

Kalman

Observed

Model

Hoek van Holland

Date: 19

Data assimilation with OpenDA
OpenDA and Parallel Computing

- Model evaluation is often the most time consuming part (90-99%)
- Often natural parallelism in the model timestepping
- Parallel time stepping for all native OpenDA models (OO)
Summary

- OpenDA: Data assimilation toolbox for Data assimilation/calibration.
- Object oriented design allows easy exchange of building blocks.
- End users do not need to do any programming for experimenting.
- Various ways for adjusting (your) model for use in OpenDA.
- Conceptual support for all kinds of parallel computing.