Building Ensemble-Based Data Assimilation Systems for High-Dimensional Models with the Parallel Data Assimilation Framework PDAF

Lars Nerger, Paul Kirchgessner, Wolfgang Hiller
Alfred Wegener Institute for Polar and Marine Research
Bremerhaven, Germany
Overview

How to simplify to apply data assimilation?
- simplify building a data assimilation application

Structure data assimilation application into
- generic part
- case-specific part (model and observations)

Provide
- software for generic part (e.g. filter methods, incl. methods like localization & inflation)
- code templates and documentation for case-specific part
Example: Forecast model for North and Baltic Seas

**Model** surface temperature

**Satellite** surface temperature

Focus on ensemble-based assimilation
- Ensemble Kalman filters
- Particle filters

PDAF: A tool for data assimilation

PDAF - Parallel Data Assimilation Framework

- a program library for data assimilation
- provide support for ensemble forecasts
- provide fully-implemented filter and smoother algorithms (EnKF, LETKF, LSEIK, LESTKF … easy to add more)
- easily useable with (probably) any numerical model (applied with NEMO, MITgcm, FESOM, HBM, TerrSysMP, …)
- makes good use of supercomputers (Fortran, MPI & OpenMP)
- allows for separate development of model and assimilation algorithms
- first public release in 2004; continued development

Open source:
Code and documentation available at
http://pdaf.awi.de

Framework Considerations
3 components of an assimilation system

Model
initialization
time integration
post processing

mesh data/coordinates

DA method
initialization
analysis step
ensemble transformation

Observations
obs. vector
obs. operator
obs. error

state

time

Ensemble-based Kalman Filter

Kalman filter: express probability distributions by mean and covariance matrix

EnKF (Evensen, 1994): Use ensembles to represent probability distributions

- Ensemble forecasting
- Initial sampling
- State estimate
- Forecast
- Analysis
- Ensemble transformation
- Observation

Time 0, Time 1, Time 2
Offline coupling – separate programs

Model

Start

Initialize Model
generate mesh
Initialize fields

Do i=1, nsteps

Time stepper
consider BC
Consider forcing

Post-processing

Stop

Model error

Assimilation program

Start

read ensemble files

analysis step

write model restart files

Stop

generic

For each ensemble state
• Initialize from restart files
• Integrate
• Write restart files

+ Simple to implement
- Inefficient:
  • file reading/writing
  • model restarts

• Read restart files (ensemble)
• Compute analysis step
• Write new restart files

Lars Nerger et al. – Building Ensemble DA Systems with PDAF
Ensemble filter analysis step

Analysis operates on state vectors (all fields in one vector)

Filter analysis
1. update mean state
2. ensemble transformation

Ensemble of state vectors
X

Vector of observations
y

Observation operator
H(...)

Observation error covariance matrix
R

For localization:
Local ensemble
Local observations
Filter analysis implementation

Operate on state vectors

- Filter doesn’t know about ‘fields’
- Computationally most efficient
- Call-back routines for
  - Transfer between model fields and state vector
  - Observation-related operations
  - Localization operations

For forecast

- Transfer data from state vector to model fields
Logical separation of assimilation system

Single program

Filter
Initialization
analysis
re-initialization

Core of PDAF

Model
initialization
time integration
post processing

modify parallelization

Observations
quality control
obs. vector
obs. operator
obs. error

state
time

state
observations

mesh data

Explicit interface

Indirect exchange (module/common)

2-level Parallelism

1. Multiple concurrent model tasks
2. Each model task can be parallelized
   - Analysis step is also parallelized
   - Configured by “MPI Communicators”
2 compartment system – strongly coupled DA

might be separate programs
Extending a Model for Data Assimilation

Model

Start

Initialize Model
generate mesh
Initialize fields

Do $i=1$, $n_{steps}$

Time stepper
consider BC
Consider forcing

Post-processing

Stop

Extension for
data assimilation

ensemble forecast
enabled by parallelization

also possible for coupled models
with several executables

plus:
Possible model-specific adaption.
e.g. Euler time step after assimilation
(NEMO model)
Framework solution with generic filter implementation

Model with assimilation extension

Core-routines of assimilation framework

Case specific callback routines

Start

init_parallel_DA

Initialize Model

Init_DA

Do $i=1, nsteps$

Time stepper

Assimilate

Post-processing

Stop

Generic

DA_Init
Set parameters
Initialize ensemble

DA_Analysis
Check time step
Perform analysis
Write results

DA_Model_Error

Dependent on model and observations

Read ensemble from files

Initialize state vector from model fields

Initialize vector of observations

Apply observation operator to a state vector

multiply R-matrix with a matrix

Subroutine calls or parallel communication

Lars Nerger et al. – Building Ensemble DA Systems with PDAF
PDAF interface structure

- Defined calls to PDAF routines and to call-back routines
- Model and observation specific operations:
  elementary subroutines implemented in model context
- User-supplied call-back routines for elementary operations:
  - transfers between model fields and ensemble of state vectors
  - observation-related operations
  - filter pre/post-step to analyze ensemble
- User supplied routines can be implemented as routines of the model (e.g. share common blocks or modules)
PDAF originated from comparison studies of different filters

**Filters**
- EnKF (Evensen, 1994 + perturbed obs.)
- ETKF (Bishop et al., 2001)
- SEIK filter (Pham et al., 1998)
- SEEK filter (Pham et al., 1998)
- ESTKF (Nerger et al., 2012)
- LETKF (Hunt et al., 2007)
- LSEIK filter (Nerger et al., 2006)
- LESTKF (Nerger et al., 2012)

**Smoothers** for
- ETKF/LETKF
- ESTKF/LESTKF
- EnKF

Not yet released:
- serial EnSRF
- particle filter
- EWPF
- NETF
Features of online program

- minimal changes to model code when combining model with filter algorithm
- model not required to be a subroutine
- no change to model numerics!
- model-sided control of assimilation program (user-supplied routines in model context)
- observation handling in model-context
- filter method encapsulated in subroutine
- complete parallelism in model, filter, and ensemble integrations
Global ocean model

FESOM (Finite Element Sea-ice Ocean model, Danilov et al. 2004)

• Uses unstructured triangular grid

Global configuration

- $1.3^\circ$ resolution, 40 levels
- horizontal refinement at equator
- state vector size $10^7$

Setup used for assimilation of sea surface height data
Parallel Performance

Use between 64 and 4096 processor cores of SGI Altix ICE cluster (HLRN-II)

94-99% of computing time in model integrations

**Speedup**: Increase number of processes for each model task, fixed ensemble size

- factor 6 for 8x processes/model task
- one reason: time stepping solver needs more iterations

**Scalability**: Increase ensemble size, fixed number of processes per model task

- increase by ~7% from 512 to 4096 processes (8x ensemble size)
- one reason: more communication on the network
• Simulate a “model”
• Choose an ensemble
  • state vector per processor: \(10^7\)
  • observations per processor: \(2 \times 10^5\)
• Ensemble size: 25
• 2GB memory per processor
• Apply analysis step for different processor numbers
  • 12 – 120 – 1200 – 12000

• Very small increase in analysis time (~1%)
• Didn’t try to run a real ensemble of largest state size (no model yet)
Summary

• Simplify building data assimilation systems
• Efficient online coupling with minimal changes to model code
• Generic model interface and case-specific call-back routines
• Parallelization allows for ensemble forecasts
• Data assimilation framework PDAF (http://pdaf.awi.de) supports high-dimensional models
• Coding you own Ensemble Kalman filter or Particle Filter usually not necessary

Thank you!

Lars.Nerger@awi.de - Building ensemble DA systems