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

Lars Nerger

Alfred Wegener Institute
Helmholtz Center for Polar and Marine Research
Bremerhaven, Germany

OceanPredict’19, Halifax, Canada
Motivation

For efficient application of data assimilation we like to

- simplify the steps to set up a data assimilation system and to apply it
- ensure high computational efficiency
- be able to easily take up new developments in data assimilation
- profit from experience from various data assimilation applications

→ Community data assimilation software allows for this
PDAF: A tool for data assimilation

PDAF - Parallel Data Assimilation Framework

- a program library for ensemble data assimilation
- provides support for parallel ensemble forecasts
- provides filters and smoothers - fully-implemented & parallelized (EnKF, LETKF, LESTKF, NETF … easy to add more)
- easily useable with (probably) any numerical model
- run from laptops to supercomputers (Fortran, MPI & OpenMP)
- Usable to study assimilation methods and to apply real assimilation
- first public release in 2004; continued development
- ~370 registered users; community contributions

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

Ensemble Filters

Ensemble Kalman Filters & Particle Filters

→ Use ensembles to represent probability distributions (uncertainty)
→ Use observations to update ensemble

There are many possible choices!
What is optimal is part of our research
Different choices in PDAF

Lars Nerger et al. – Efficient Ensemble DA with PDAF
3 Components of Assimilation System

- **Model**
  - initialization
  - time integration
  - post processing

- **Ensemble Filter**
  - Initialization
  - analysis
  - ensemble transformation

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

- **single program**

- **Core of PDAF**
  - state time
  - observations

- **Explicit interface**

- **Indirect exchange (module/common)**

Augmenting a Model for Data Assimilation

Model
- single or multiple executables
- coupler might be separate program

revise parallelization enables ensemble forecast

Extension for data assimilation

plus:
- Possible model-specific adaption
e.g. in NEMO: treat leap-frog time stepping

Lars Nerger et al. – Efficient Ensemble DA with PDAF
2-level Parallelism

1. Multiple concurrent model tasks
2. Each model task can be parallelized
   - Analysis step is also parallelized
   - Configured by “MPI Communicators”

Lars Nerger et al. – Efficient Ensemble DA with PDAF
Minimal changes to NEMO

Add to mynode (lin_mpp.F90) just before init of myrank

```fortran
#ifdef key_USE_PDAF
   CALL init_parallel_pdaf(0, 1, mpi_comm_opa)
#endif
```

Add to nemo_init (nemogcm.F90) at end of routine

```fortran
#ifdef key_USE_PDAF
   CALL init_pdaf()
#endif
```

Add to stp (step.F90) at end of routine

```fortran
#ifdef key_USE_PDAF
   CALL assimilate_pdaf()
#endif
```

Modify dyn_nxt (dynnxt.F90) – for Euler time step

```fortran
#ifdef key_USE_PDAF
   IF((neuler==0 .AND. kt==nit000) .OR. assimilate)
#else
   Aaaaaaaa
   Aaaaaaaa
   aaaaaaaaa
   Start
   Stop
   Initialize Model
   generate mesh
   Initialize fields
   Time stepper
   consider BC
   Consider forcing
   Post-processing
   WHILE istp ≤ nitend
   init_pdaf
   assimilate_pdaf
   For MITgcm: calls included in MITgcm release version. PDAF provides the model-binding routines
   PDAF
   Parallel Data Assimilation Framework
  init_parallel_pdaf
   init_pdaf
   assimilate_pdaf
   Assimilation
   Post-processing
   Stop
```
Why Online coupling?

Offline-coupling is simple to implement but can be very inefficient.

**Example:**
Timing from atmosphere-ocean coupled model (AWI-CM) with daily analysis step:

- Model startup: 95 s
- Integrate 1 day: 28 s
- Model postprocessing: 14 s
- Analysis step: 1 s

Restarting this model is \(~3.5\) times more expensive than integrating 1 day

→ avoid this for data assimilation
PDAF interface structure

- Defined calls to PDAF routines and to call-back routines
- Model und 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 (observation operator, …)
  - 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: User-friendliness

Assumption: Users know their model
- let users implement assimilation system in model context

For users, model is not just a forward operator
- let users augment their model for data assimilation

Keep simple things simple:
- Define subroutine interfaces to separate model and assimilation based on arrays
- No object-oriented programming
  (most models don’t use it; most model developers don’t know it; not many objects would be involved)
- Users directly implement observation-specific routines
  (no indirect description of e.g. observation layout)
- Users run their model as without data assimilation, but with additional options
PDAF Capability: Very big test case

- 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 core
- Apply analysis step for different processor numbers
  - 12 – 120 – 1200 – 12000
- Very small increase in analysis time (~1%)
  (Ideal would be constant time)
- Didn’t try to run a real ensemble of largest state size (no model yet)

Timing of global SEIK analysis step

State dimension: 1.2e11
Observation dimension: 2.4e9
**PDAF Application Examples**

**HBM:** Coastal assimilation of SST, in situ and ocean color data (S. Losa, M. Goodliff)

**AWI-CM:** coupled atmos.-ocean assimilation (project ESM, Qi Tang)

**MITgcm-REcoM:**
- global ocean color assimilation (Himansu Pradhan)
  - Talk tomorrow

+ external applications & users, like
  - MITgcm sea-ice assim (NMEFC Beijing)
  - Geodynamo (IPGP Paris, A. Fournier)
  - TerrSysMP-PDAF (hydrology, FZ Juelich)
  - CMEMS Baltic-MFC (DMI/BSH/SMHI)
  - CFSv2 (J. Liu, IAP-CAS Beijing)
  - NEMO (U. Reading, P. J. van Leeuwen)
Current algorithms in PDAF

PDAF originated from comparison studies of different filters

Filters and smoothers
- EnKF (Evensen, 1994 + perturbed obs.)
- (L)ETKF (Bishop et al., 2001)
- SEIK filter (Pham et al., 1998)
- ESTKF (Nerger et al., 2012)
- NETF (Toedter & Ahrens, 2015)

All methods include
- global and localized versions
- smoothers

Model binding
- MITgcm

Toy model
- Lorenz-96

Not yet released:
- serial EnSRF
- particle filter
- EWPF
- Generate synthetic observations (V1.14)

Not yet released:
- NEMO
Summary

- **PDAF**: Software framework simplifies building data assimilation systems
- Efficient online assimilation coupling with minimal changes to model code
- Setup of data assimilation with coupled model
  1. Configuration of parallelization
  2. Implementation of user call-back routines
     - for interfacing with model code and
     - observation handling

Open source:
Code, documentation, and tutorial available at

http://pdaf.awi.de

Lars.Nerger@awi.de – Efficient Ensemble DA with PDAF
References

- http://pdaf.awi.de

Thank you!

Lars.Nerger@awi.de – Efficient Ensemble DA with PDAF