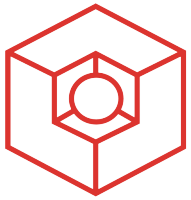
PDAF's Data Model

Lars Nerger

Alfred Wegener Institute for Polar and Marine Research
Bremerhaven, Germany

and
Bremen Supercomputing Competence Center BremHLR
Bremen, Germany

lars.nerger@awi.de



BremHLR
Kompetenzzentrum für Höchstleistungsrechnen Bremen

About PDAF

A computational framework for ensemble data assimilation

- Aimed at large-scale problems
- Optimized for optimal compute performance and memory usage
- Naturally parallel using MPI
  (Compilation without MPI possible using dummy-MPI)
- Applied with between 1 and 4096 processor cores

Further

- Easy connectivity to models
  (extending a model by a few subroutine calls)
- Provide parallel ensemble integration for non-parallel models

Core part of PDAF

"The analysis step" (for various filters)

Operate entirely on vectors and matrices

Independent from model and observations

Dimensions: (INTEGER)	size of state vector
	size of observation vector
	size of ensemble
Arrays: (REAL)	state vector
	ensemble matrix
	observation vector
	(plus internal temporary arrays)
Parallelization:	Assume domain decomposition (sub-states, mode-decomp. variant exists)

Connection of PDAF with model

Single executable (DA-extended model)
(typical, but separate executables possible)

In-memory information transfer

2 routines (implemented by user):

get_state: Initialize model fields from an ensemble state vector

put_state: Initialize ensemble state vector from model fields

Routine to analyze ensemble:

prepoststep

(e.g. for compute variances, write fields, etc.)

Handling of Observations

User-implemented routines:

init_dim_obs: Initialize size of observation vector

init_obs: Initialize observation vector

obs_op: Apply observation operator to a state vector

Observation error covariance matrix R – depends on filter

SEIK/ETKF/SEEK/ESTKF:

prodRinvA: Product of inverse of R with some matrix

EnKF

add_obs_error: Add R to some matrix.

Additional routines for localization

Implementation up to the user (for optimal performance)