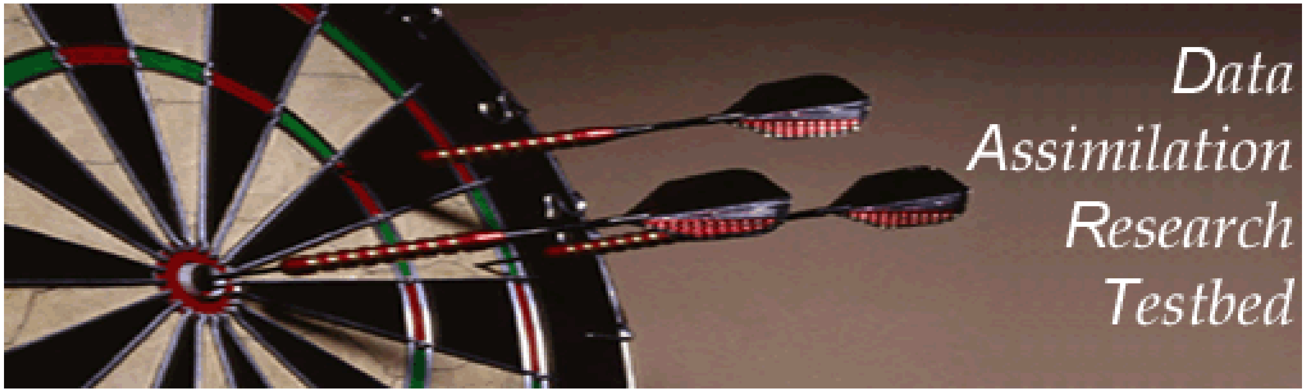
Data Assimilation Research Testbed Tutorial



Section 19: DART-Compliant Models and Making Models Compliant

Version 1.0: June, 2005

DART compliant models:

DART uses identical assimilation code for menagerie of models.

Same namelists you've been using in low-order models still apply.

To work with DART, models must supply a subset of 13 interfaces.

Normally done by creating a model_mod that 'wraps' the model.

More on interfaces below.

Large models compliant with DART.

1. Two layer Primitive Equations global model.

Incorporated by Jeff Whitaker and Tom Hamill at NOAA/CDC.

Cheapest look at more complex dynamics and spherical domain.

Relatively small and fast.

2. GFDL grid point atmospheric GCM dynamical core.

Can be configured with fewer than 20,000 variables OR
As large as you want

Good tool for transitioning from small to huge models.

3. MIT general circulation model: Annulus configuration.

Useful for looking at different geometries.

Will be usable in conjunction with laboratory facility.

Could also be configured as traditional ocean or atmosphere GCM.

4. Community Atmosphere Model (CAM).

NCAR's global climate GCM.

A spectral model

- State is actually represented as spherical harmonic amplitudes.
- DART sees state on a grid.

Medium to huge configurations: (250,000 to many million).

Can act as a Numerical Weather Prediction Model.

5. Weather Research and Forecast Model (MMM version)

NCAR/NCEP regional gridpoint model.
Designed for short term predictions of smaller scale weather.

Can be configured with nested grid.

Medium to huge configurations.

Needs horizontal boundary forcing.

6. ROSE Middle atmosphere dynamics and chemistry model.

Medium to huge.

Code only available by permission.

Lots of nearly passive tracers.

Potential for many interesting indirect observing types.

Creating a DART compliant model.

Total of 13 interfaces for full compliance.

Can have partial compliance with subset of these.

See html documentation for existing models and...

See *models/template/model_mod.f90* for stripped interfaces.

Most minimal interface includes:

1. *function get_model_size*: how big is the model?
2. *function get_state_meta_data*: returns location (and kind) of each state variable element (DART sees one long vector for state).
3. *subroutine static_init_model*: does any initialization required by model, for instance allocating storage, reading namelist...

An initial ensemble of state vectors; can be generated offline.

With this implementation, can assimilate identity obs. at a single time.

Increasing functionality:

4. *function get_model_time_step*: what is δt for model?

5. Stub for *subroutine adv_1step* (just say δt is 0).

Can now test repeated assimilations of identity observations.

Further increasing functionality (option A):

6. Allowing non-identity observation operators:

Implement *subroutine model_interpolate*:

Given a location (and kind), return interpolated state value.

Can test repeated assimilations of non-identity observations.

Further increasing functionality (option B):

7. Some way to advance the model in time.

This can be done by implementing *subroutine adv_1step*

Given state vector, what is state vector after δt?

OR

By implementing a shell script that advances the model.

Reads a state vector from a file, writes updated vector.

Can do arbitrary OSSEs.

Can do OSEs for models that have real observations.

Additional interfaces for increased functionality:

8. *subroutine init_conditions*: returns a state to start from.
9. *subroutine init_time*: returns an initial time to start from.
10. *subroutine pert_model_states*: Generate an ensemble member by perturbing a control state.
11. *subroutines nc_write_model_atts* & *nc_write_model_vars* netCDF output for your model state vector.
12. *subroutine model_get_close_states*: required for efficient assimilation in bigger models. Returns a list of state variables within a certain distance of a given location.
13. *subroutine end_model*: cleans up when all done.