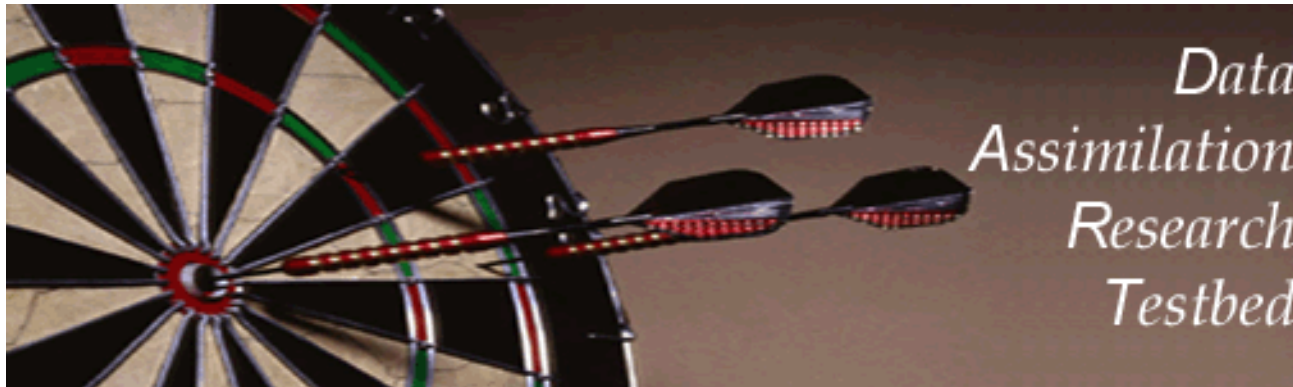


# Data Assimilation Research Testbed Tutorial



## Section 19: DART-Compliant Models and Making Models Compliant

Version 2.0: September, 2006

## 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 12 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.

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.

Also finite volume version available

## 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 12 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  $\delta t$  for model?
5. Stub for *subroutine adv\_1step* (just say  $\delta t$  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  $\delta t$ ?

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 end\_model*: cleans up when all done.