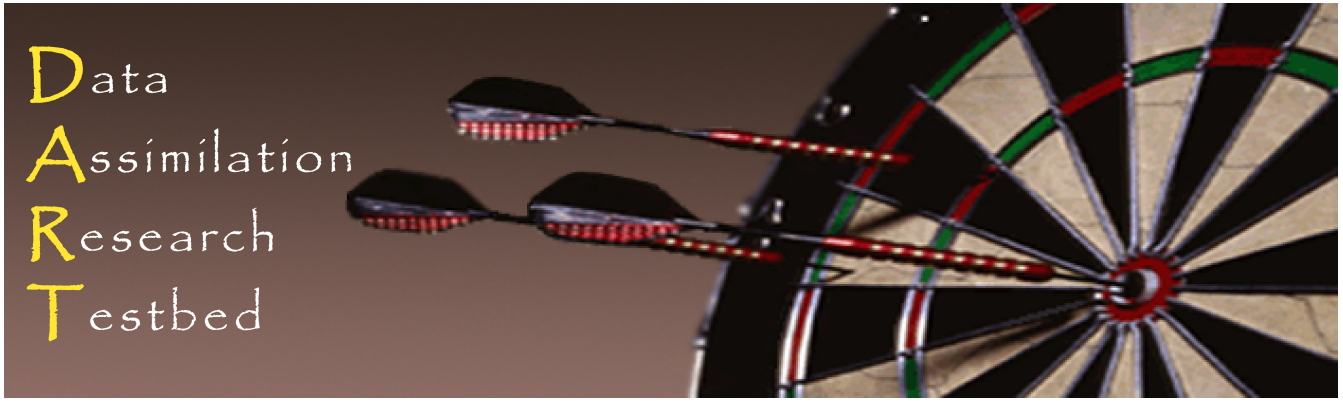
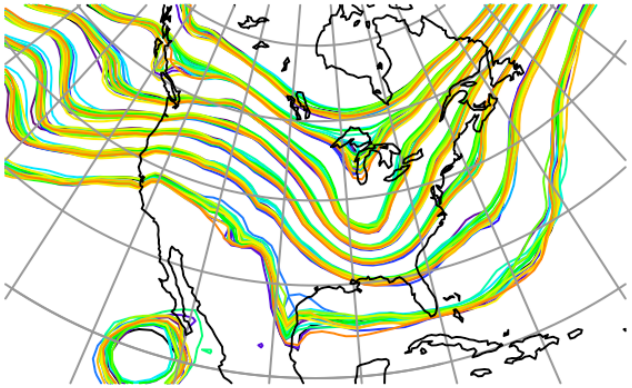


DART Tutorial Section 19: Making DART-Compliant Models



The National Center for Atmospheric Research is sponsored by the National Science Foundation. Any opinions, findings and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

NCAR | National Center for
UCAR | Atmospheric Research

DART Compliant Models

This section has not yet been updated for the Manhattan release.
An update version should be available soon.

In the meantime, the online documentation for adding a model to DART is available at:

https://dart.ucar.edu/pages/Models.html#adding_a_model

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 18 interfaces.

Normally done by creating a model_mod that either 'wraps' the model (so the model itself is embedded in filter), or a model_mod that interfaces to the model (where it runs outside of filter).

More on interfaces follows.

Large Models Compliant with DART

Many models available, from simple to highly complex.

Low order models, often for DA research

Geophysical models: Atmosphere, Ocean, Land, Solar, Space Weather, Upper Atmosphere, Hydrology, CICE (coming soon)

Non-geophysical applications: Economics, Target-tracking

DART Compliant Global Atmosphere Models

CAM	Community Atmosphere Model (all 3 dynamical cores)	NCAR
CAM/CHEM	CAM with Chemistry	NCAR
WACCM	Whole Atmosphere Community Climate Model	NCAR
AM2	Atmosphere Model 2	NOAA/GFDL
ECHAM4,6	European Centre Hamburg Model	Hamburg
Planet WRF	Global version of WRF	JPL
MPAS	Model for Prediction Across Scales	NCAR/DOE
LMDZ5	Laboratoire de Météorologie Dynamique	Indian Inst. of Technology, Delhi

DART Compliant Regional Atmosphere Models

WRF/ARW	Weather Research and Forecast Model	NCAR
WRF/CHEM	WRF with Chemistry	NCAR
NCOMMAS	Collaborative Model for Multiscale Atmospheric Simulation	NOAA/NSSL
COAMPS	Coupled Ocean/Atmosphere Mesoscale Prediction System	US Navy
CMAQ *	Community Multi-scale Air Quality (* interface not available)	EPA
COSMO	Consortium for Small-Scale Modeling	DWD

DART Compliant Ocean Models

POP	Parallel Ocean Program	DOE/NCAR
MIT OGCM	Ocean General Circulation Model	MIT
ROMS	Regional Ocean Modeling System	Rutgers
MPAS	Model for Prediction Across Scales (Under development)	DOE/LANL

DART Compliant Upper Atm / Space Weather

ROSE		NCAR
TIEGCM	Thermosphere Ionosphere Electrodynamic GCM	NCAR/HAO
GITM	Global Ionosphere Thermosphere Model	Michigan
Solar Dynamo	Dynamo/Sunspot Model	NCAR/HAO

DART Compliant Land / Hydrology Models

CLM	Community Land Model	NCAR
NOAH	Relatively simple land model	Community
CABLE	Community Atmosphere Biosphere Land Exchange	CAWCR (Australia)
WRF Hydro	Hydrologic Model	NCAAR

Creating a DART Compliant Model

Total of 18 interfaces for full compliance.

Can have partial compliance with subset of these.

See:

http://www.image.ucar.edu/DAReS/DART/DART2_Documentation.php#adding_a_model

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

The 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.

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.

Increasing Functionality (cont)

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 experiments for models that have real observations.

Additional interfaces

8. *subroutine init_conditions:* returns a state to start from. May not be appropriate for large models and can error out in this case.

9. *subroutine init_time:* returns an initial time to start from. Again, may not be appropriate for large models and can error out.

10. *subroutine pert_model_state:* Generate an ensemble member by perturbing a control state. Optional. (filter will perturb if model_mod has no special needs.)

11. *subroutines nc_write_model_atts* & *nc_write_model_vars:* format netCDF diagnostic output for your model. Can be a single 1D vector for initial implementation.

12. *subroutine end_model:* cleans up when done. E.g. deallocate space allocated in the static_init_model subroutine.

Additional interfaces (cont)

13-15. *get_close_maxdist_init, get_close_obs_init, get_close_obs:* Routines that are normally provided by *location_mod.* Can start by including a *use* statement from *location_mod* and a *public* declaration as seen in the *template/model_mod.* These allow more control on efficiency, vertical transformations, etc. for close searches if needed.

16. *ens_mean_for_model:* not usually required for low-order models. Some large models need the model ensemble mean to do consistent computations independent of any single ensemble member.
For example, computing consistent distances between locations that depend on pressure. Can be a stub that simply returns.

Approaches, Tools

Copy the DART models/template directory to a new location.

Add routines in the order suggested here so you can test as you go.

Decide how DART and your model will interact:

1. Subroutine-callable model inside filter
2. Serial, easily-advanced model driven from script called by filter
3. Parallel, complex-scripting model advanced separately from filter

Use *models/template/model_mod_check.f90* for testing specific routines as you implement them.

Start with single observation, no model advance, before getting more complicated.

Assimilating Observations in Your Model

To assimilate observation types in your model, you need code in your model_mod.f90 :: model_interpolate() subroutine for all generic kinds needed by the forward operators.

1. Could be a simple interpolation if that observation kind is also present in the model state.

2. Could require a vertical transformation based on kinds present in the model state. e.g. temperature, moisture, pressure.

3. Could require forward operator code in an obs_def_xxx_mod.f90 if the observation kind cannot be directly interpolated by the model. In this case you need to be able to interpolate any kinds required by the forward operator code in the obs_def file.

DART Tutorial Index to Sections