DART Documentation

Release 2.0.0

DART Team

Feb 06, 2019

Contents:

CHAPTER

ONE

DART LANAI DIFFERENCES FROM KODIAK RELEASE NOTES

1.1 Overview

This document includes an overview of the changes in the DART system since the Kodiak release. For further details on any of these items look at the HTML documentation for that specific part of the system.

There is a longer companion document for this release, the Lanai Release Notes, which include installation instructions, a walk-through of running one of the low-order models, the diagnostics, and a description of non-backward compatible changes. See the Notes for Current Users section for additional information on changes in this release.

1.2 Changes to Core DART routines

This section describes changes in the basic DART library routines since the Kodiak release.

- Added a completely new random number generator based on the Mersenne Twister algorithm from the GNU scientific library. It seems to have better behavior if reseeded frequently, which is a possible usage pattern if perfect_model_obs is run for only single steps and the model is advanced in an external script. As part of this code update all random number code was moved into the random_seq_mod and random_nr_mod is deprecated.
- Perfect_model_obs calls a seed routine in the time manager now that generates a consistent seed based on the current time of the state. This makes subsequent runs give consistent results and yet separate runs don't get identical error values.
- Added random number generator seeds in several routines to try to get consistent results no matter how many MPI tasks the code was run with. This includes:
  - cam model_mod.f90, pert_model_state()
  - assim_tools_mod.f90, filter_assim(), filter kinds 2, 3, and 5
  - wrf model_mod.f90, pert_model_state()
  - adaptive_inflate_mod.f90, adaptive_inflate_init(), non-deterministic inf
- There is a new &filter_nml namelist item: enable_special_outlier_code. If .true. the DART quality control code will call a separate subroutine at the end of filter.f90 to evaluate the outlier threshold. The user can add code to that routine to change the threshold based on observation type or values as they wish. If .false. the default filter outlier threshold code will be called and the user routine ignored.
- If your *model_mod.f90* provides a customized *get_close_obs()* routine that makes use of the types/kinds arguments for either the base location or the close location list, there is an important change in this release. The fifth argument to the *get_close_obs()* call is now a list of generic kinds corresponding to the location list. The fourth argument to the *get_dist()* routine is now also a generic kind and not a specific type. In previous versions of the

system the list of close locations was sometimes a list of specific types and other times a list of generic kinds. The system now always passes generic kinds for the close locations list for consistency. The base location and specific type remains the same as before. If you have a *get_close_obs()* routine in your *model_mod.f90* file and have questions about usage, contact the DART development team.

- Filter will call the end_model() subroutine in the model_mod for the first time. It should have been called all along, but was not.
- Added a time sort routine in the time_manager_mod.
- Avoid a pair of all-to-all transposes when setting the inflation mean and sd from the namelist. The new code finds the task which has the two copies and sets them directly without a transpose. The log messages were also moved to the end of the routine - if you read in the mean/sd values from a restart file the log messages that printed out the min/max values needed to be after the read from the file.
- Reordered the send/receive loops in the all-to-all transposes to scale better on yellowstone.
- Remove a state-vector size array from the stack in read_ensemble_restart(). The array is now allocated only if needed and then deallocated. The ensemble write routine was changed before the Kodiak release but the same code in read was apparently not changed simply as an oversight.
- If the ensemble mean is selected to be written out in dart restart file format, the date might not have been updated correctly. The code was fixed to ensure the ensemble mean date in the file was correct.
- filter writes the ensemble size into the log file.
- Reorganized the code in the section of obs_model_mod that prints out the time windows, with and without verbose details. Should be clearer if the next observation is in or out of the current assimilation window, and if the model needs to advance or not.
- Added a fill_inflation_restart utility which can write a file with a fixed mean and sd, so the first step of a long assimilation run can use the same 'start_from_restart_file' as subsequent steps.
- Added new location module options:
  - Channel coordinate system
  - [0-1] periodic 3D coordinate system
  - X,Y,Z 3D Cartesian coordinate system
  - 2D annulus coordinate system

1.2.1 |

1.3 New Models or Changes to Existing Models

Several new models have been incorporated into DART. This section details both changes to existing models and descriptions of new models that have been added since the Kodiak release.

- Support for components under the CESM framework:
  - Added support for the Community Land Model (CLM).
  - Added support to run the Community Atmospheric Model (CAM) under the CESM framework.
  - Added support for the CESM 1.1.1 release for CAM, POP, CLM: includes experiment setup scripts and assimilation scripts.

  - CAM, POP, and/or CLM can be assimilated either individually or in combination while running under the CESM framework. If assimilating into multiple components, they are assimilated sequentially with observations only affecting a single component directly. Other components are indirectly affected through interactions with the coupler.
  - Setup scripts are provided to configure a CESM experiment using the multi-instance feature of CESM to support ensembles for assimilation.
  - POP state vector contains potential temperature; observations from the World Ocean Database are in-situ or sensible temperature. The model_mod now corrects for this.
    * The state vector has all along contained potential temperature and not in-situ (sensible) temperature. The observations from the World Ocean Database are of sensible temperature. Changed the specific kind in the model_mod to be `QTY_POTENTIAL_TEMPERATURE` and added new code to convert from potential to in-situ temperature. Differences for even the deeper obs (4-5km) is still small (~ 0.2 degree). (in-situ or sensible temperature is what you measure with a regular thermometer.)
  - Support for the SE core (HOMME) of CAM has been developed but **is not** part of the current release. Contact the DART group if you have an interest in running this configuration of CAM.
- Changes to the WRF model_mod:
  - Allow advanced microphysics schemes (needed interpolation for 7 new kinds)
  - Interpolation in the vertical is done in log(p) instead of linear pressure space. log(p) is the default, but a compile-time variable can restore the linear interpolation.
  - Added support in the namelist to avoid writing updated fields back into the wrf netcdf files. The fields are still updated during the assimilation but the updated data is not written back to the wrfinput file during the dart_to_wrf step.
  - Fixed an obscure bug in the vertical convert routine of the wrf model_mod that would occasionally fail to convert an obs. This would make tiny differences in the output as the number of mpi tasks change. No quantitative differences in the results but they were not bitwise compatible before and they are again now.
- Added support for the MPAS_ATM and MPAS_OCN models.
  - Added interpolation routines for the voroni-tesselation grid (roughly hexagonal)
  - Includes vertical conversion routines for vertical localization.
  - Added code to the mpas_atm model to interpolate specific humidity and pressure, so we can assimilate GPS obs now.
- Added support for the 'SQG' uniform PV two-surface QC+1 spectral model.
- Added support for a flux-transport solar dynamo model.
- Added support for the GITM upper atmosphere model.
- Added support for the NOAH land model.
- Added support for the NAAPS model.
- Added model_mod interface code for the NOGAPS model to the SVN repository.
- Simple advection model:
  - Fix where the random number seed is set in the models/simple_advection model_mod - it needed to be sooner than it was being called.

1.4 New or changed Forward Operators

This section describes changes to the Foward Operators and new Generic Kinds or Specific Types that have been added since the Kodiak release.

- Many new kinds added to the DEFAULT_obs_kind_mod.f90:
  - QTY_CANOPY_WATER
  - QTY_CARBON
  - QTY_CLW_PATH
  - QTY_DIFFERENTIAL_REFLECTIVITY
  - QTY_DUST
  - QTY_EDGE_NORMAL_SPEED
  - QTY_FLASH_RATE_2D
  - QTY_GRAUPEL_VOLUME
  - QTY_GROUND_HEAT_FLUX QTY_HAIL_MIXING_RATIO
  - QTY_HAIL_NUMBER_CONCENTR
  - QTY_HAIL_VOLUME QTY_ICE QTY_INTEGRATED_AOD
  - QTY_INTEGRATED_DUST
  - QTY_INTEGRATED_SEASALT QTY_INTEGRATED_SMOKE
  - QTY_INTEGRATED_SULFATE
  - QTY_LATENT_HEAT_FLUX
  - QTY_LEAF_AREA_INDEX
  - QTY_LEAF_CARBON
  - QTY_LEAF_NITROGEN QTY_LIQUID_WATER
  - QTY_MICROWAVE_BRIGHT_TEMP
  - QTY_NET_CARBON_FLUX
  - QTY_NET_CARBON_PRODUCTION
  - QTY_NEUTRON_INTENSITY
  - QTY_NITROGEN QTY_RADIATION
  - QTY_ROOT_CARBON
  - QTY_ROOT_NITROGEN
  - QTY_SEASALT
  - QTY_SENSIBLE_HEAT_FLUX
  - QTY_SMOKE
  - QTY_SNOWCOVER_FRAC
  - QTY_SNOW_THICKNESS
  - QTY_SNOW_WATER
  - QTY_SO2

  - QTY_SOIL_CARBON
  - QTY_SOIL_NITROGEN
  - QTY_SPECIFIC_DIFFERENTIAL_PHASE
  - QTY_STEM_CARBON
  - QTY_STEM_NITROGEN
  - QTY_SULFATE
  - QTY_VORTEX_WMAX
  - QTY_WATER_TABLE_DEPTH
  - QTY_WIND_TURBINE_POWER
  - plus slots 151-250 reserved for Chemistry (specifically WRF-Chem) kinds
- Added a forward operator for total precipitable water. It loops over model levels so it can be used as an example of how to handle this without having to hardcode the number of levels into the operator.
- Added a forward operator (and obs_seq file converter) for COSMOS ground moisture observations.
- Added a forward operator (and obs_seq file converter) for MIDAS observations of Total Electron Count.
- Added a 'set_1d_integral()' routine to the obs_def_1d_state_mod.f90 forward operator for the low order models. This subroutine isn't used by filter but it would be needed if someone wanted to write a standalone program to generate obs of this type. We use this file as an example of how to write an obs type that has metadata, but we need to give an example of how to set the metadata if you aren't using create_obs_sequence interactively (e.g. your data is in netcdf and you have a separate converter program.)

1.5 Observation Converters

This section describes support for new observation types or sources that have been added since the Kodiak release.

- Added an obs_sequence converter for wind profiler data from MADIS.
- Added an obs_sequence converter for Ameriflux land observations(latent heat flux, sensible heat flux, net ecosystem production).
- Added an obs_sequence converter for MODIS snow coverage measurements.
- Added an obs_sequence converter for COSMOS ground moisture observations.
- Added an obs_sequence converter for MIDAS observations of Total Electron Count.
- Updated scripts for the GPS converter; added options to convert data from multiple satellites.
- More scripting support in the MADIS obs converters; more error checks added to the rawin converter.
- Added processing for wind profiler observation to the wrf_dart_obs_preprocess program.
- Fix BUG in airs converter - the humidity obs are accumulated across the layers and so the best location for them is the layer midpoint and not on the edges (levels) as the temperature obs are. Also fixed off-by-one error where the converter would make one more obs above the requested top level.
- Made gts_to_dart converter create separate obs types for surface dewpoint vs obs aloft because they have different vertical coordinates.
- Converted mss commands to hpss commands for a couple observation converter shell scripts (inc AIRS).

- New matlab code to generate evenly spaced observations on the surface of a sphere (e.g. the globe).
- Added obs_loop.f90 example file in obs_sequence directory; example template for how to construct special purpose obs_sequence tools.
- Change the default in the script for the prepbufr converter so it will swap bytes, since all machines except ibms will need this now.
- The 'wrf_dart_obs_preprocess' program now refuses to superob observations that include the pole, since the simple averaging of latitude and longitude that works everyplace else won't work there. Also treats observations near the prime meridian more correctly.

1.6 New or updated DART Diagnostics

This section describes new or updated diagnostic routines that have been added since the Kodiak release.

- Handle empty epochs in the obs_seq_to_netcdf converter.
- Added a matlab utility to show the output of a 'hop' test (running a model for a continuous period vs. stopping and restarting a run).
- Improved the routine that computes axes tick values in plots with multiple values plotted on the same plot.
- The obs_common_subset program can select common observations from up to 4 observation sequence files at a time.
- Add code in obs_seq_verify to ensure that the ensemble members are in the same order in all netcdf files.
- Added support for the unstructured grids of mpas to our matlab diagnostics.
- Fix to writing of ReportTime in obs_seq_coverage.
- Fixed logic in obs_seq_verify when determining the forecast lat.
- Fixed loops inside obs_seq_coverage which were using the wrong limits on the loops. Fixed writing of 'ntimes' in output netcdf variable.
- The obs_common_subset tool supports comparing more than 2 obs_seq.final files at a time, and will loop over sets of files.
- Rewrote the algorithm in the obs_selection tool so it had better scaling with large numbers of obs.
- Several improvements to the 'obs_diag' program:
  - Added preliminary support for a list of 'trusted obs' in the obs_diag program.
  - Can disable the rank histogram generation with a namelist item.
  - Can define height_edges or heights in the namelist, but not both.
  - The 'rat_cri' namelist item (critical ratio) has been deprecated.
- Extend obs_seq_verify so it can be used for forecasts from a single member. minor changes to obs_selection, obs_seq_coverage and obs_seq_verify to support a single member.
- Added Matlab script to read/print timestamps from binary dart restart/ic files.
- Default for obs_seq_to_netcdf in all the namelists is now 'one big time bin' so you don't have to know the exact timespan of an obs_seq.final file before converting to netCDF.

1.7 Tutorial, Scripting, Setup, Builds

This section describes updates and changes to the tutorial materials, scripting, setup, and build information since the Kodiak release.

- The mkmf-generated Makefiles now take care of calling 'fixsystem' if needed so the mpi utilities code compiles without further user intervention.
- Make the default input.nml for the Lorenz 96 and Lorenz 63 model gives good assimilation results. Rename the original input.nml to input.workshop.nml. The workshop_setup script renames it back before doing anything else so this won't break the workshop instructions. Simplify all the workshop_setup.csh scripts to do the minimal work needed by the DART tutorial.
- Updates to the models/template directory with the start of a full 3d geophysical model template. Still under construction.
- Move the pdf files in the tutorial directory up a level. Removed framemaker source files because we no longer have access to a working version of the Framemaker software. Moved routines that generate figures and diagrams to a non-distributed directory of the subversion repository.
- Enable netCDF large file support in the work/input.nml for models which are likely to have large state vectors.
- Minor updates to the doc.css file, make pages look identical in the safari and firefox browsers.
- Added a utility that sorts and reformats namelists, culls all comments to the bottom of the file. Useful for doing diffs and finding duplicated namelists in a file.
- Cleaned up mkmf files - removed files for obsolete platforms and compilers, updated suggested default flags for intel.
- Update the mkmf template for gfortran to allow fortran source lines longer than 132 characters.

1.8 Terms of Use

Contact: DART core group Revision: $Revision$ Source: URL Change Date: $Date$ Change history: try "svn log" or "svn diff"

CHAPTER

TWO

FORTRAN

2.1 Closest Member Tool

program `closest_member_tool`

Program to overwrite the time on each ensemble in a restart file.

Use types_mod (i8() , max_num_doms() , max_files() , r8() , obstypelength()), state_structure_mod (get_num_domains()), ensemble_manager_mod (init_ensemble_manager() , get_my_num_vars() , end_ensemble_manager() , ensemble_type() , get_my_vars() , compute_copy_mean()), obs_kind_mod (get_num_quantities() , get_name_for_quantity() , get_index_for_quantity()), utilities_mod (set_multiple_filename_lists() , nmlfileunit() , close_file() , open_file() , check_namelist_read() , e_msg() , e_err() , error_handler() , do_nml_file() , register_module() , find_namelist_in_file() , do_nml_term()), state_vector_io_mod (read_state()), time_manager_mod (set_time_missing() , print_time() , time_type()), sort_mod (index_sort()), mpi_utilities_mod (my_task_id() , send_sum_to() , finalize_mpi_utilities() , task_count() , initialize_mpi_utilities()), location_mod (location_type()), assim_model_mod (get_model_size() , static_init_assim_model() , get_state_meta_data()), io_filenames_mod (read_copy() , io_filenames_init() , set_io_copy_flag() , get_stage_metadata() , file_info_type() , set_member_file_metadata() , set_file_metadata() , get_restart_filename() , file_info_dump() , stage_metadata_type())

CHAPTER

THREE

CONVERSION PROCESS

– Working from Revision 12951 (12945 on rma_trunk and trunk)

3.1 1. Move from Subversion server to Github.

3.1.1 A. In docker container use git svn clone to create a local Git clone of the desired Subversion branch/trunk.

Example:

$ docker run -it niemacka/shpinx-git

$ git svn clone https://svn-dares-dart.cgd.ucar.edu/DART/trunk/ –no-metadata –no-minimize-url

- Use –revision ##### to pull from a given revision number forward
- For classic/trunk use revision 9994

3.1.2 B. Create an empty GitHub repository under the desired account at github.com.

3.1.3 C. Push local git clone to GitHub repo.

Example:

$ git push –all https://github.com/account-name/repo-name

- Correct GitHub repo url will be at the top of empty repository.
- This will always push to the Master branch of the repo.
- Will require GitHub username and password

3.1.4 D. It is possible that there will be an error that causes the push to fail because GitHub does not allow files bigger than 100 MB. If that is the case these files will have to be removed in order to push to GitHub (They can be added later to the latest commit/revision but their commit/revision history cannot be kept). Use the git filter-branch command to remove necessary files.

Example:

$ git filter-branch –force –index-filter ‘git rm –cached –ignore-unmatch filename‘ –prune-empty –tag-name-filter cat – –all

- For each file over 100 MB then...

$ git for-each-ref –format=‘delete %(refname)‘ refs/original | git update-ref –stdin

$ git reflog expire –expire=now –all

$ git gc –prune=now

3.1.5 E. Now push to GitHub again and there should be no issues. (Use the —force option with git push).

Example:

$ git push –force –all https://github.com/account-name/repo-name

3.1.6 F. If the Subversion branch was not the current working branch create a new branch in Github cloned from the Master branch.

Example:

– Move ‘trunk‘ to GitHub using the above steps. Then if trunk is not the current working branch click branch dropdown in top left of GitHub repository and type a new branch name (for example ‘classic‘) that describes the branch. Then press enter. The new branch should now appear in the dropdown.

- At this point the master branch can be deleted if you so choose, or you can overwrite it by pushing another branch from subversion to GitHub using the steps above.
- To overwrite the master branch use the –force option with git push

3.2 2. Convert existing documentation to sphinx-doc style documentation. (This could be done either before or after the SVN branches have been pushed to GitHub.)

3.2.1 A. From the local machine create a clone of the new Github repository.

Example:

$ git clone https://github.com/account-name/repo-name

3.2.2 B. Move all desired html files (or other convertible filetypes) that relate to documentation to a single directory.

3.2.3 C. To convert the html file to markdown (.md) which is used by sphinx use pandoc.

Example:

$ docker run -v `pwd` :/source jagregory/pandoc -f html -t markdown myfile.html -o myfile.md

- If there are a large number of files enter the file names you want to convert into a txt file with one name per line. Then use a python script to convert them.
- Once the desired files are converted in pandoc you can delete the redundant html files.

3.2.4 D. Next push the changes to GitHub.

Example:

$ git add —all

$ git commit -m “converted html to md with pandoc”

- git commits require a message

$ git push -u origin master

- Once the changes have been pushed to GitHub the local repository can be deleted if you so choose.

3.2.5 E. Open an interactive docker container and clone the new Github repo.

Example:

$ docker run -it niemacka/sphinx

$ git clone https://github.com/account-name/repo-name

3.2.6 F. Choose a location for the sphinx-doc setup. Note that file paths are important for sphinx and particularly for the auto-documenting features. Sphinx likes to use paths relative to the location of the conf.py and index.rst files so it may be simplest to set these up in the root directory, although there is a way to setup an absolute path in the config (which I haven’t gotten to work with GitHub).

3.2.7 G. Once in the desired directory use the sphinx quickstart command to start an interactive process that will generate all the required files for sphinx-doc.

Example:

$ sphinx-quickstart

– answer the questions presented and enable any feature that you intend to use

- Many elements can be left as default with no issues.
- It is important to enable the python auto-doc extension
- Enable github pages integration
- It is important to create the Unix makefile

3.2.8 H. There should now be a conf.py file that contains the configuration info that was set up during the quickstart. To allow sphinx to read markdown files add 'recommonmark' to the sphinx extensions section of the conf.py file. This should be near the top of the file.

Eample:

extensions = [

'sphinx.ext.autodoc',

'sphinx.ext.githubpages',

'recommonmark',

]

3.2.9 I. There is also an index.rst file (or whatever name it was given during the quick start). This file contains the Table of Contents Tree (toctree) which is where you can list the markdown files that you wish to include in your documentation using their relative path to the index.rst file and excluding the .md extension.

Example:

- First install vim in the container

$ apt-get update

$ apt-get install vim -y

$ vi index.rst

– Should see something like this...

.. toctree:: :maxdepth: 2

– Just list the files you want to include...

.. toctree:: .:maxdepth: 2

usage/installation

usage/quickstart

...

3.2.10 J. Once the toctree has been updated. Use the commands from the makefile to create an html version and a pdf version of the documents.

Example:

$ make html

$ make latexpdf

- If no other changes have been made to the documents both of these commands could generate a lot of warnings (which should be related to incorrect references to other internal documentation and images) They will build if you are running in an interactive container

but these errors will stop circleci so it is important to check all references and images in a document and confirm there are no errors before integration with circleci.

- You can verify that they built by checking the _build directory for a latex and html subdirectory.
- Some issue may be resolved by running make html or make latexpdf multiple times.

3.2.11 K. Push the changes to GitHub like in step d.

- When you try to commit from within a docker container you will get a message asking you to set an email and name to assign to the commit by adding to the git config.

Example:

$ git config –global user.email “you@example.com“

$ git config –global user.name “Your Name“

3.3 3. Set up the sphinx-fortran extension. (This can all be done on GitHub or in a local clone of the repository but the process is the same.)

3.3.1 A. Open the conf.py configuration file and add the sphinx fortran extensions to the list of sphinx extensions this should be near the top of the document.

Example:

– It should look something like this... extensions = [

‘sphinx.ext.autodoc‘,

‘sphinx.ext.todo‘,

‘sphinx.ext.githubpages‘,]

– Then add sphinxfortran.fortran_domain and sphinxfortran.fortran_autodoc... extensions = [

‘sphinx.ext.autodoc‘,

‘sphinx.ext.todo‘,

‘sphinx.ext.githubpages‘,

‘sphinxfortran.fortran_domain‘,

‘sphinxfortran.fortran_autodoc‘,

]

3.3.2 B. Below the extensions add a source for the fortran files to be parsed. This can be a list of files or a relative path using wildcards.

Example:

fortran_src = [‘relativepath1/specific-file.f90’, ‘relativepath2/specific-file2.f90’]

or

fortran_src = [‘relativepath/*.f90’]

3.3.3 C. All setup should now be complete and the sphinx-fortran extension commands can now be used to document fortran programs, modules, etc. . .

Example:

– In a md document that is included in the toctree add a line like the examples below to auto-document that element.

.. f:autoprogram:: program-name

.. f:autofunction:: [modname]/funcname

.. f:autosubroutine:: [modname/]/subrname

– To document all programs, functions, and subroutines in a source file you can use. . .

..f:autosrcfile:: pathname

- Important to note auto-documenting will include descriptive comments about the element but only if they start on the line immediately below where the element is being declared.

- program example
- !> example description must start here or it will not be included in the
- !> documentation.

program example

!> This doesn’t work

3.3.4 D. Now when the html or pdf sphinx files are generated the auto-documentation should be included.

- You may need to use the make html and make latexpdf commands multiple times.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

C