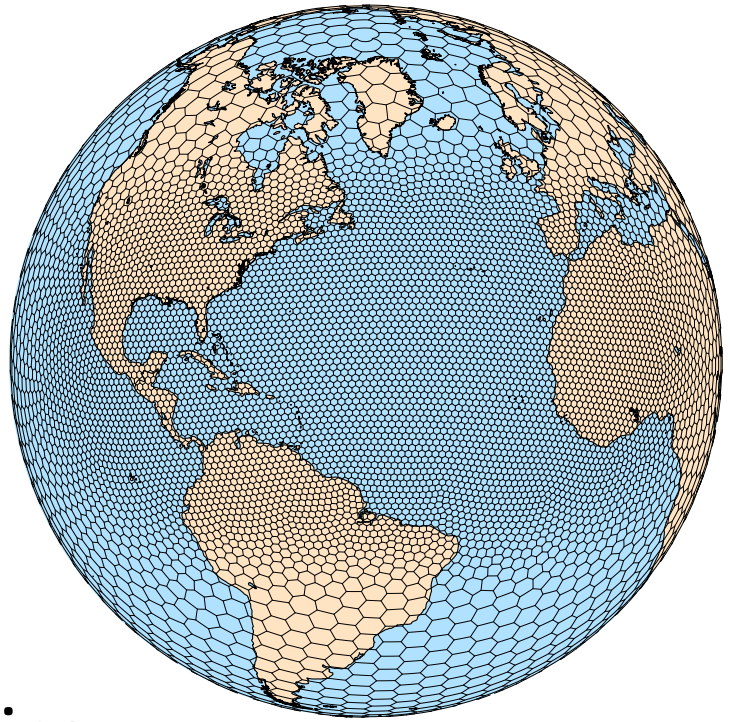
MPAS

Model for Prediction Across Scales



- Overview
- Meshes
- Atmospheric solver, physics
- *Compiling and running MPAS*
- Summary
- Practical session

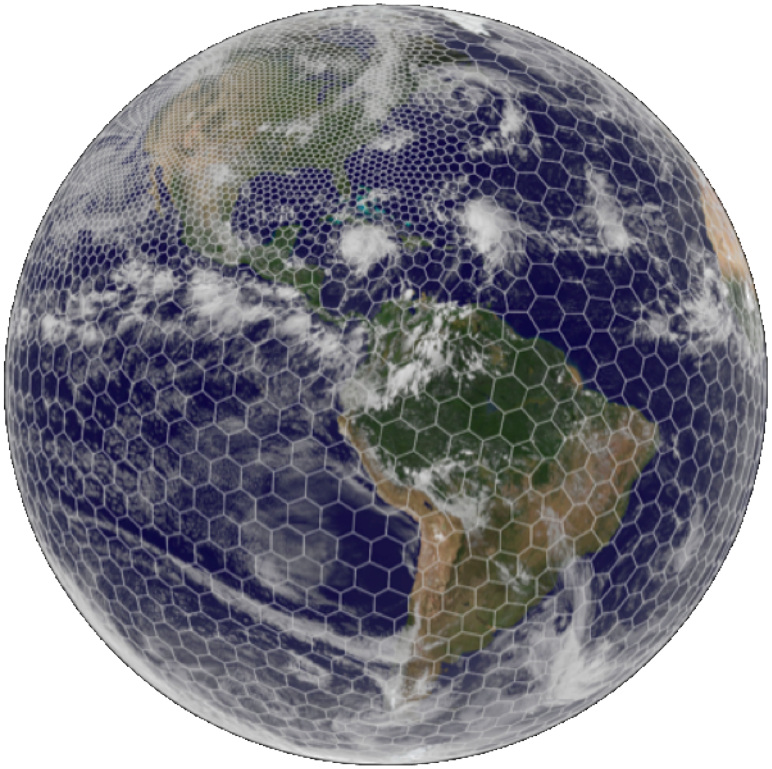
Los Alamos NATIONAL LABORATORY EST. 1943

U.S. DEPARTMENT OF ENERGY | Office of Science

NSF

NCAR NATIONAL CENTER FOR ATMOSPHERIC RESEARCH

OUTLINE



1. **Preliminary requirements**
2. Obtaining MPAS source code
3. Compiling MPAS-Atmosphere
4. Creating initial conditions
5. Running the model
6. Post-processing

http://mpas-dev.github.io/

Preliminary requirements

In order to compile MPAS and its required libraries, working C and Fortran compilers are required

- The Fortran compiler should be recent enough to support the ISO_C_BINDING module from the Fortran 2003 standard and procedure pointer components of derived types
- Most versions of common compilers from the last couple of years should be fine

Building MPAS (v4.0 or later) requires the following libraries:

- Any implementation of MPI-2, e.g., MPICH, MVAPICH, OpenMPI
  - Ensure that `mpif90` and `mpicc` commands are in your path
- Parallel-NetCDF (http://trac.mcs.anl.gov/projects/parallel-netcdf)
  - Set `PNETCDF` environment variable to base installation directory
- PIO (https://github.com/PARALLELIO/ParallelIO)
  - Set `PIO` environment variable to base installation directory

Preliminary requirements

Assuming Fortran and C compilers are available, and a working MPI installation is also available, installing Parallel-NetCDF and PIO should take less than 10 minutes:

Parallel-NetCDF 1.6.0

```
$ setenv CC gcc
$ setenv FC gfortran
$ setenv F77 gfortran
$ setenv MPICC mpicc
$ setenv MPIF90 mpif90
$ setenv MPIF77 mpif90
$ cd parallel-netcdf-1.6.0
$ ./configure \
  --prefix=/home/duda/pnetcdf \
  --disable-cxx
$ make
$ make install
```

PIO 1.7.1

(Assuming environment variables from Parallel-NetCDF installation)

```
$ setenv MPIFC mpif90
$ setenv PNETCDF_PATH \
                  /home/duda/pnetcdf
$ cd pio1_7_1/pio
$ ./configure \
  --prefix=/home/duda/pio \
  --disable-netcdf
$ make
$ make install
```

Notes on obtaining and installing PIO

The PIO library is undergoing rapid development, and many different versions of the library are available; *which versions are supported and recommended?*

1) For ease of installation, try PIO 1.7.1

- Can be installed using only standard 'configure' and 'make' tools
- Supports NetCDF-3 and Parallel-NetCDF I/O
- Download: https://github.com/PARALLELIO/ParallelIO/releases/tag/pio1_7_1

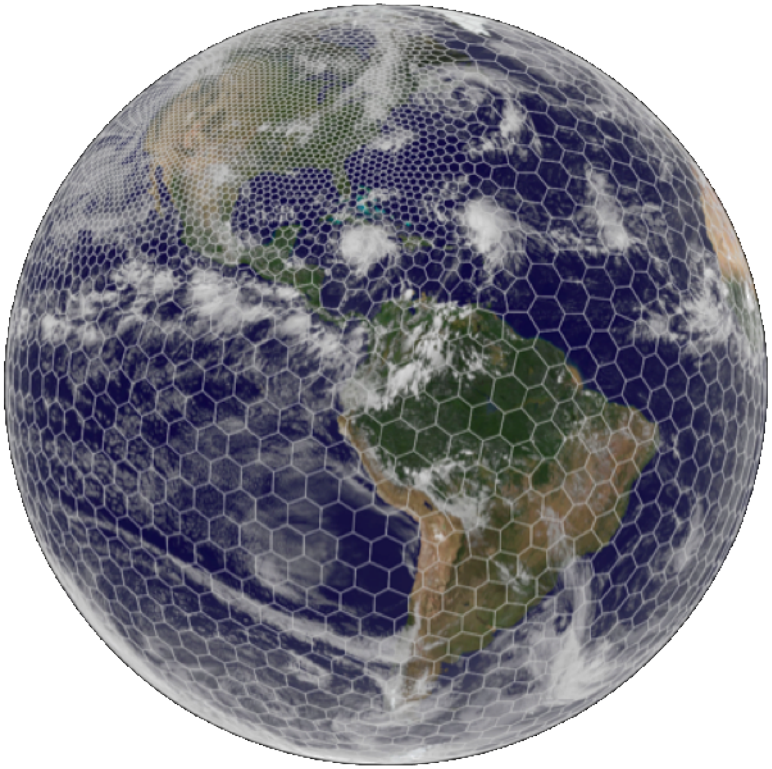
2) If NetCDF-4 I/O is needed or desirable, try PIO 1.9.19

- Requires recent versions of *'cmake'*, plus standard 'make'
- Supports NetCDF-3, NetCDF-4 (in parallel via PHDF5), and Parallel-NetCDF I/O
- Download: https://github.com/PARALLELIO/ParallelIO/releases/tag/pio1_9_19

PIO versions 2.x are not supported in MPAS v4.0 and earlier!

- The 2.x versions introduce changes to APIs that break compatibility with current MPAS release
- MPAS may be compatible with PIO 2.x (but no longer 1.x ?) in MPAS v5.0

OUTLINE



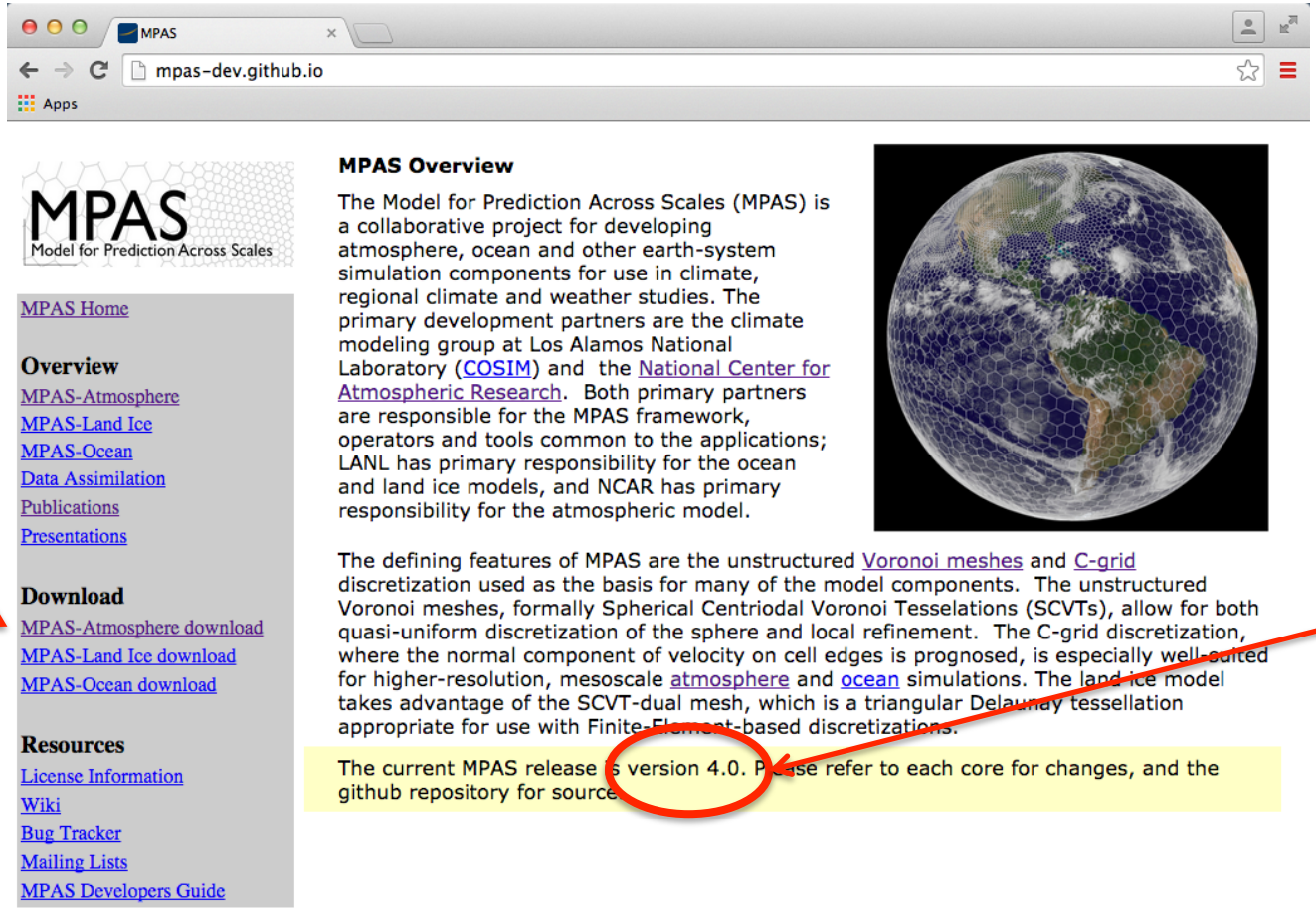
1. Preliminary requirements
2. **Obtaining MPAS source code**
3. Compiling MPAS-Atmosphere
4. Creating initial conditions
5. Running the model
6. Post-processing

http://mpas-dev.github.io/

Obtaining MPAS source code

There are two possible methods for obtaining MPAS source code:

1) Download a .tar.gz file of the latest release
2) Checkout the code directly from the release repository (preferred option!)



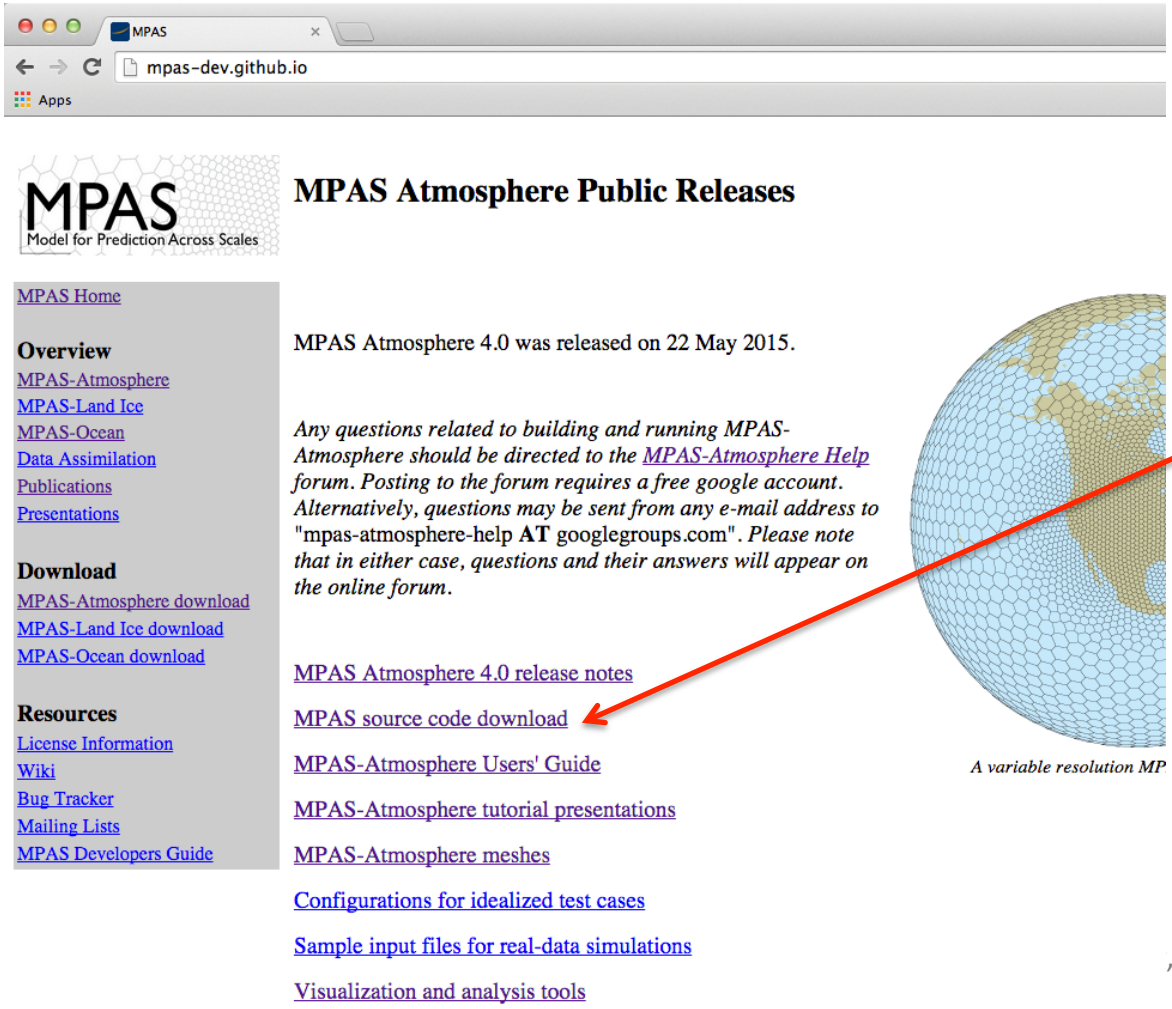
MPAS-A download options

Current release version

Obtaining MPAS source code

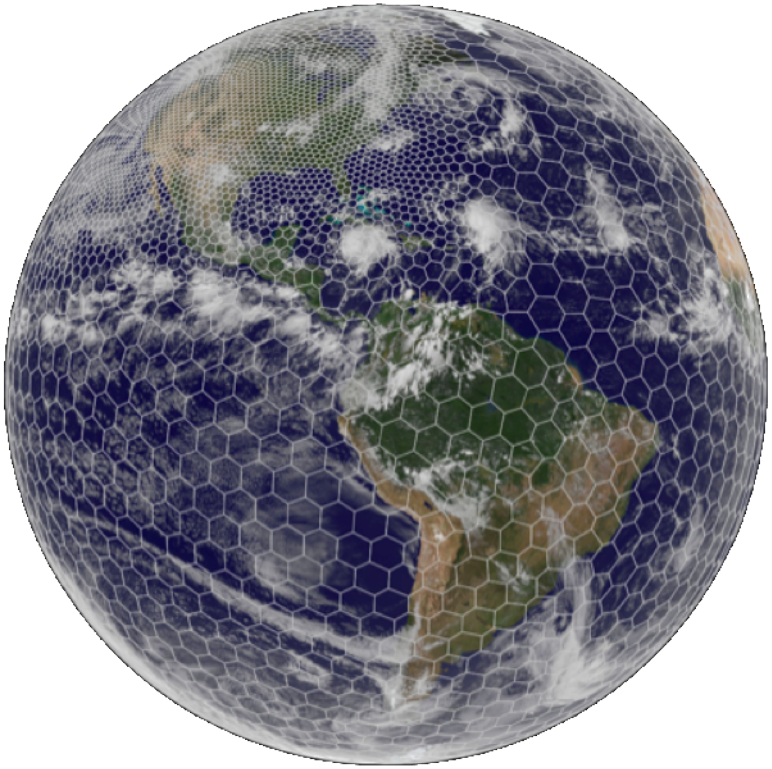
There are two possible methods for obtaining MPAS source code:

1) Download a .tar.gz file of the latest release
2) Checkout the code directly from the release repository (preferred option!)



Before downloading the code, you'll have the option to register

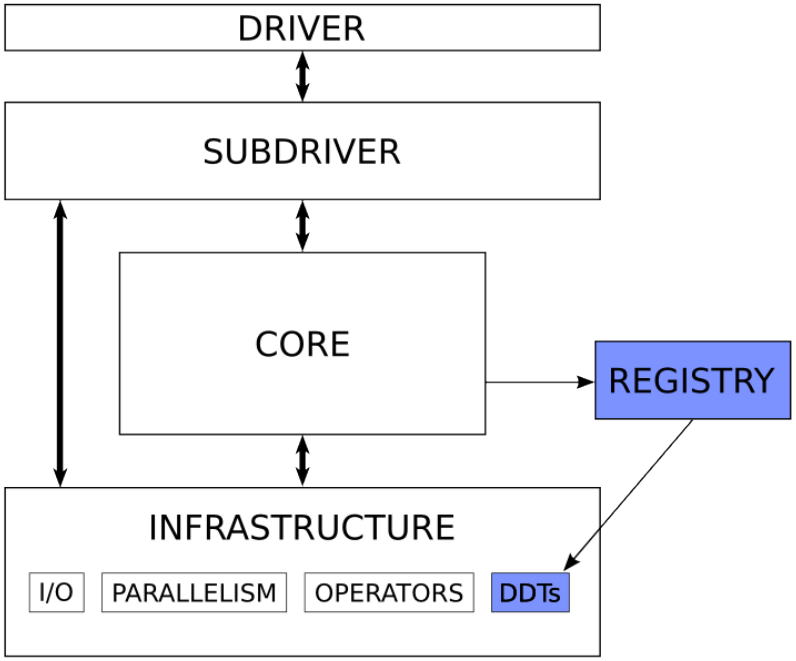
OUTLINE



1. Preliminary requirements
2. Obtaining MPAS source code
3. **Compiling MPAS-Atmosphere**
4. Creating initial conditions
5. Running the model
6. Post-processing

http://mpas-dev.github.io/

Model Organization



Checking out the MPAS code provides all MPAS models, not just MPAS-Atmosphere

- All models share a common set of infrastructure modules
- Each MPAS model is implemented as a "core" that lives in its own directory
- User must select which "core" to compile
- Each "core" is associated with a source code subdirectory under src/ and has a Registry file (similar to WRF)

Running MPAS-Atmosphere involves two "cores":

- The **init_atmosphere** core is responsible for
  - Interpolating static fields to the mesh (similar to geogrid.exe)
  - Generating a vertical grid (similar to real.exe)
  - Horizontally and vertically interpolating meteorological data to the 3-d grid (similar to metgrid.exe and real.exe)
  - *Where to we get meteorological data? From ungrib.exe!*
- The **atmosphere** core is the model itself, the equivalent of wrf.exe

Compiling MPAS

There is no “configuration” step for MPAS, unlike, e.g., for the WRF model

- All build flags are either set in the top-level Makefile or on the command-line

General MPAS build command:

```
$ make <target> CORE=<core> <options>
```

<target> can be either

clean

or

xlf
gfortran
ifort
pgi
bluegene
... plus a few others...

For MPAS-Atmosphere, <core> may be

atmosphere
init_atmosphere

<options> can be zero or more of

DEBUG=true
GEN_F90=true
AUTOCLEAN=true

Compiling MPAS

Typical build of both the init_atmosphere and atmosphere cores involves:

$ make gfortran CORE=init_atmosphere *(build init_atmosphere_model)*

$ make clean CORE=atmosphere *(clean any files used by both init_atmosphere and atmosphere)*

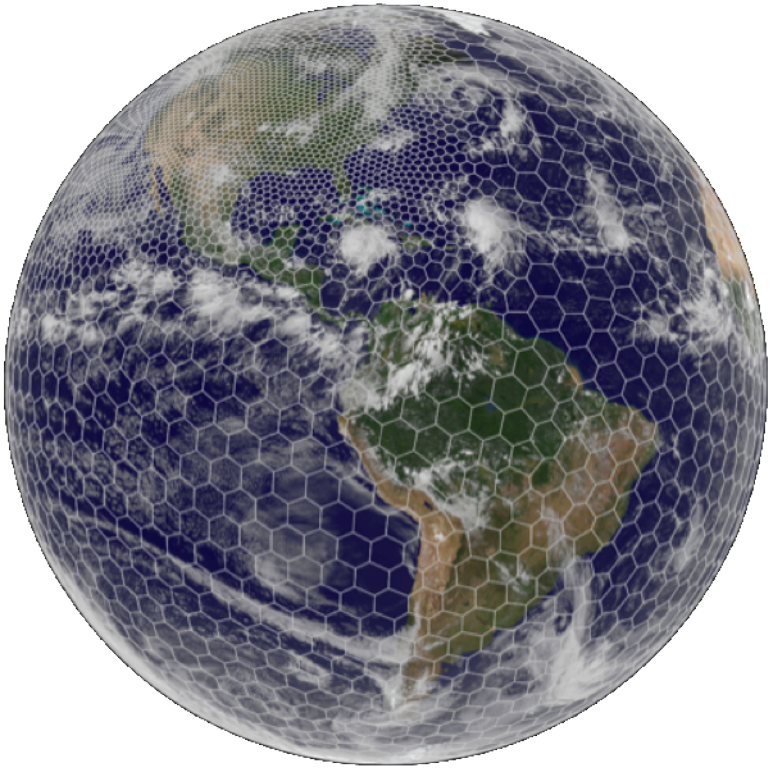
$ make gfortran CORE=atmosphere *(build atmosphere_model)*

By default, MPAS cores are built with double-precision reals

MPAS-*Atmosphere* can be built in single precision – see Section 3.4 of the Users' Guide for details

- execution time ~35% less compared with double-precision
- output files approximately half as large
- **Beginning with MPAS v3.0, it is possible to run the model in double precision while writing history files in single precision!**

OUTLINE

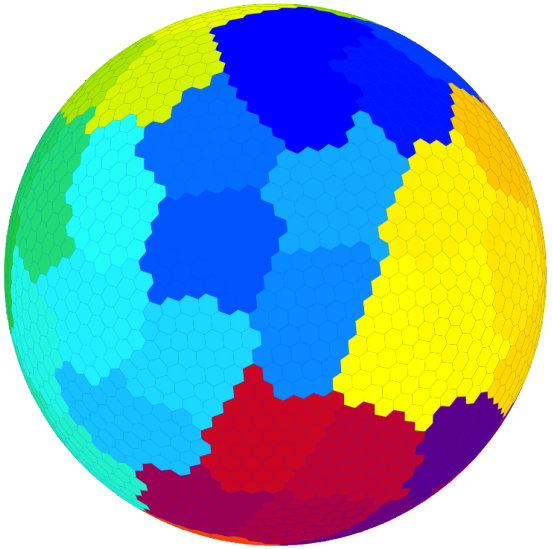


1. Preliminary requirements
2. Obtaining MPAS source code
3. Compiling MPAS-Atmosphere
4. **Creating initial conditions**
   But first, a few digressions...
5. Running the model
6. Post-processing

http://mpas-dev.github.io/

Digression: Mesh partition files

Recall that MPAS meshes must be partitioned using *Metis* in order for MPAS to be run in parallel



However, the meshes available from the MPAS-Atmosphere download page are provided with several pre-computed partition files

- **In many cases, it may not be necessary for you to run Metis yourself; just use a pre-computed partitioning**

For example, the x1.40962 mesh (about 120-km resolution) is provided with the following files:

`x1.40962.grid.nc` – the mesh itself
`x1.40962.graph.info` – the mesh connectivity graph
`x1.40962.graph.info.part.2` – pre-computed partitioning for 2 MPI tasks
`x1.40962.graph.info.part.8` – pre-computed partitioning for 8 MPI tasks
`x1.40962.graph.info.part.16` – pre-computed partitioning for 16 MPI tasks
…

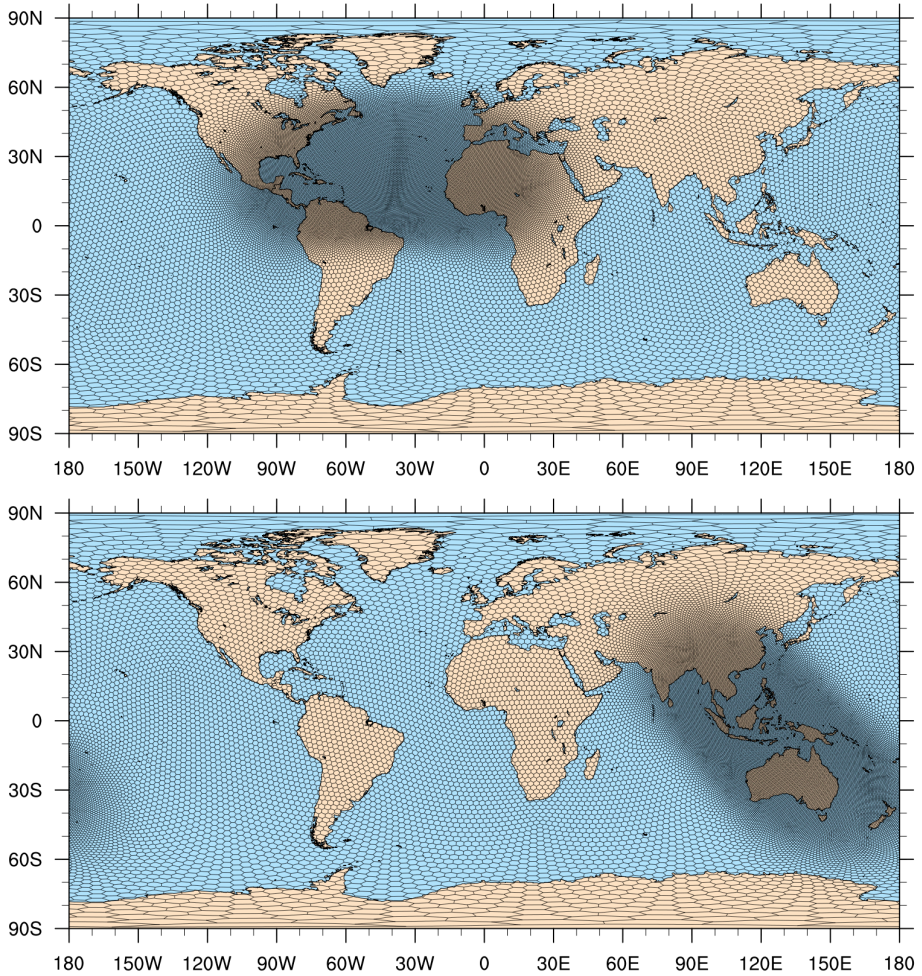
Digression: Preparing variable-resolution meshes

Before interpolating any fields to the mesh, the location of refinement can be moved:

Using the *grid_rotate* utility available through the MPAS-Atmosphere mesh download page, specify:

- Reference point location (lat,lon) in the original grid – usually the center of the refinement region
- Reference point location (lat,lon) in the rotated mesh
- A counter-clockwise rotation of the mesh about the vector from the center of the sphere through the reference point

NB: The original graph decomposition files can still be used, since cell connectivity does not change!



Above: A refinement region originally centered at 25N, 40W has been shifted to 7S, 125E and rotated by -45 degrees.

Digression: Controlling MPAS input and output

Before MPAS v3.0, the names of input and output files were set in the namelist file

- Old namelist files will not work with MPAS v3.0 and later

Beginning with MPAS v3.0, all MPAS input and output is controlled in an XML file named streams.<core>

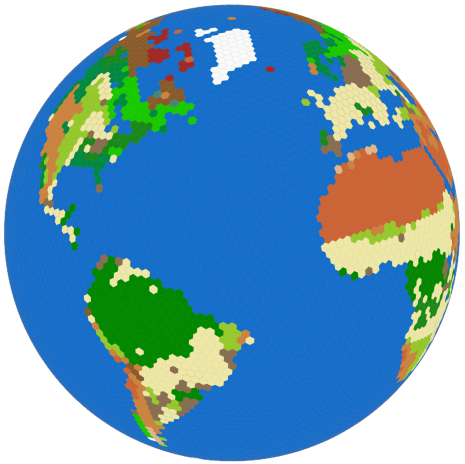
- For the *init_atmosphere* core, the file is named `streams.init_atmosphere`
- For the *atmosphere* core, the file is named `streams.atmosphere`

Example:

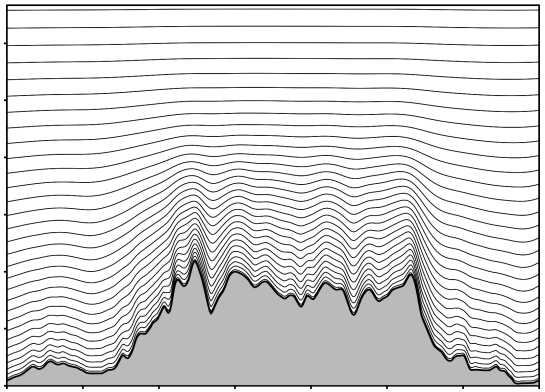
```
<immutable_stream name="input"
                  type="input"
                  filename_template="x1.40962.grid.nc"
                  input_interval="initial_only"/>
```

Refer to Chapter 5 of the MPAS-Atmosphere Users' Guide for a complete description of options for controlling input and output.

Overview of steps to run MPAS-Atmosphere

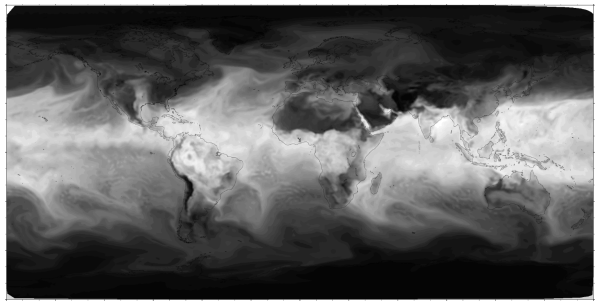


1. Interpolate geographical fields to the mesh: terrain, landuse, soil type, etc.

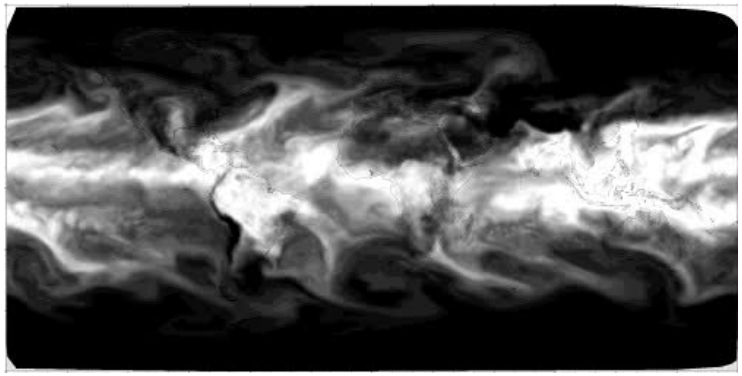


2. Generate the vertical grid

3. Prepare GRIB data with the WPS *ungrib.exe*



4. Interpolate initial conditions to the 3d mesh



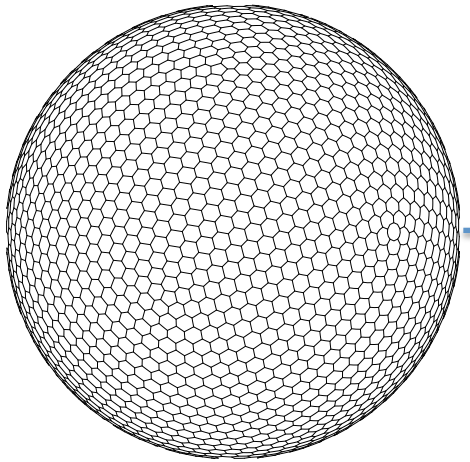
5. Run the MPAS-Atmosphere model itself

init_atmosphere_model ungrib.exe atmosphere_model

Creating real-data initial conditions: geographic fields

For real-data initial conditions, the first step is to interpolate static, geographic fields to the mesh

- Geographic datasets come from the WRF Pre-processing System
- Interpolation typically takes about 60 minutes
- *Must be run with just one MPI task!*



“input” stream filename_template:

x1.40962.grid.nc

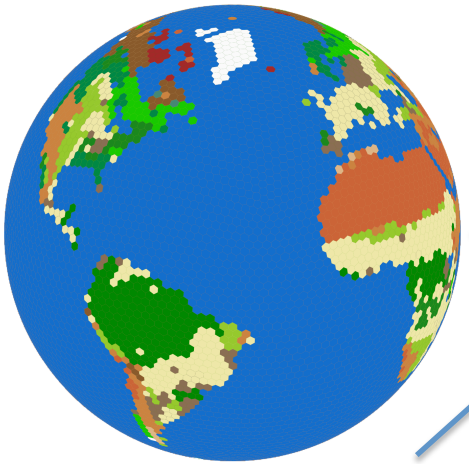
See MPAS-Atmosphere Users’ Guide, section 7.2.1 for important namelist settings...

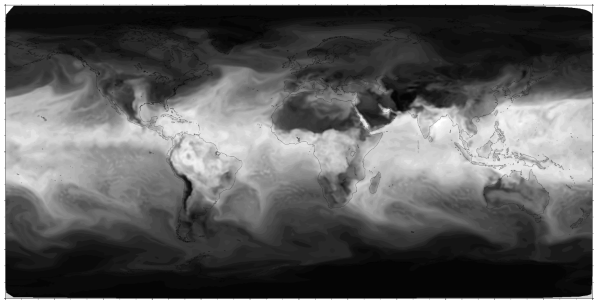
“output” stream filename_template:

x1.40962.static.nc

Creating real-data initial conditions: meteorological fields

After interpolating geographic fields to the horizontal mesh, we can generate a vertical grid and interpolate meteorological initial conditions

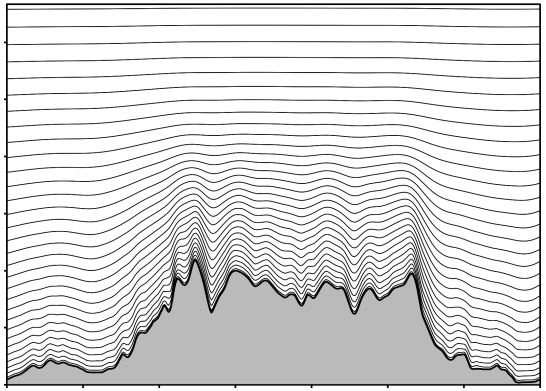


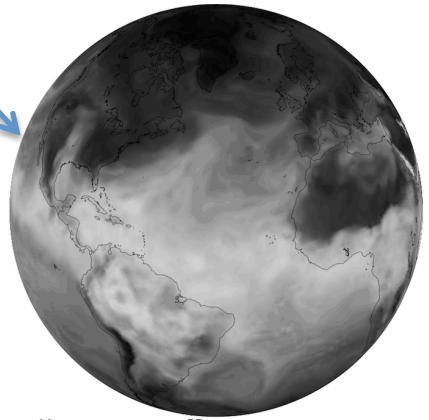


“input” stream
filename_template:

x1.40962.static.nc

init_atmosphere_model





“output” stream
filename_template:

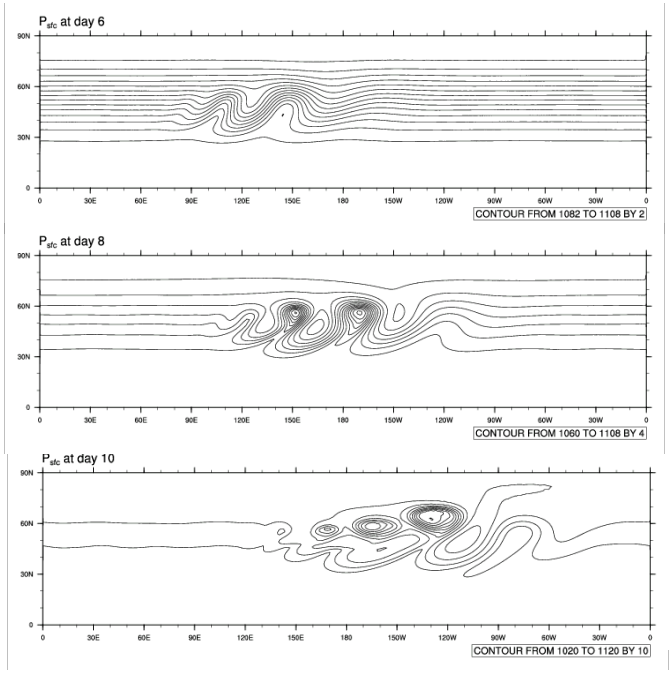
x1.40962.init.nc

See MPAS-Atmosphere Users’ Guide, section 7.2.3 for important namelist settings...

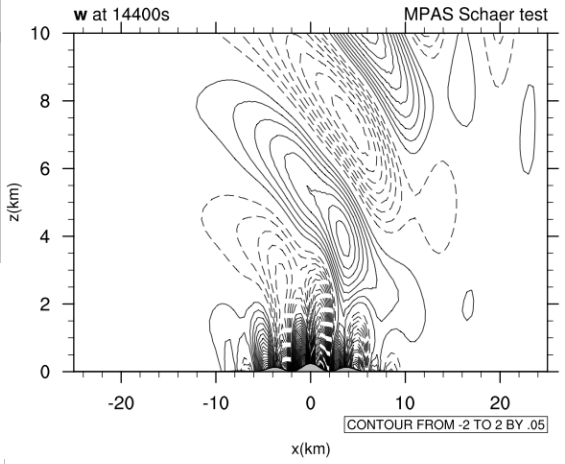
Creating idealized initial conditions

Besides real-data cases, there are several idealized cases that MPAS-Atmosphere supports “out-of-the-box”

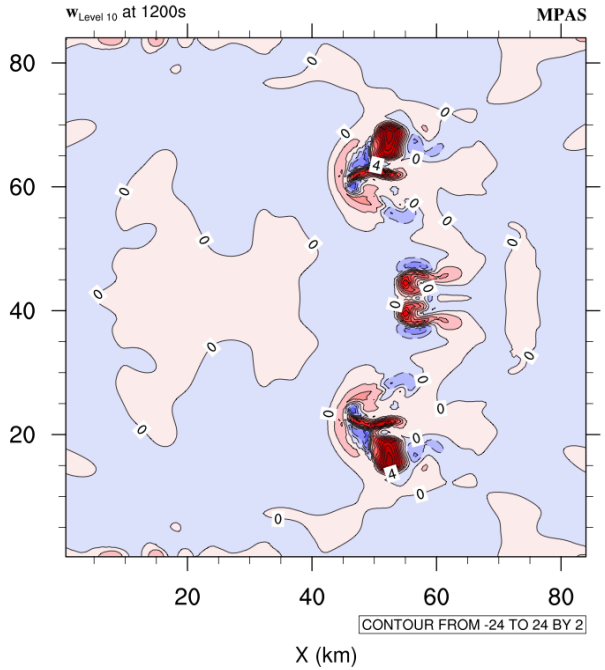
- Meshes and namelist files for these cases can be found through the MPAS-Atmosphere download page



Jablonowski and Williamson baroclinic wave

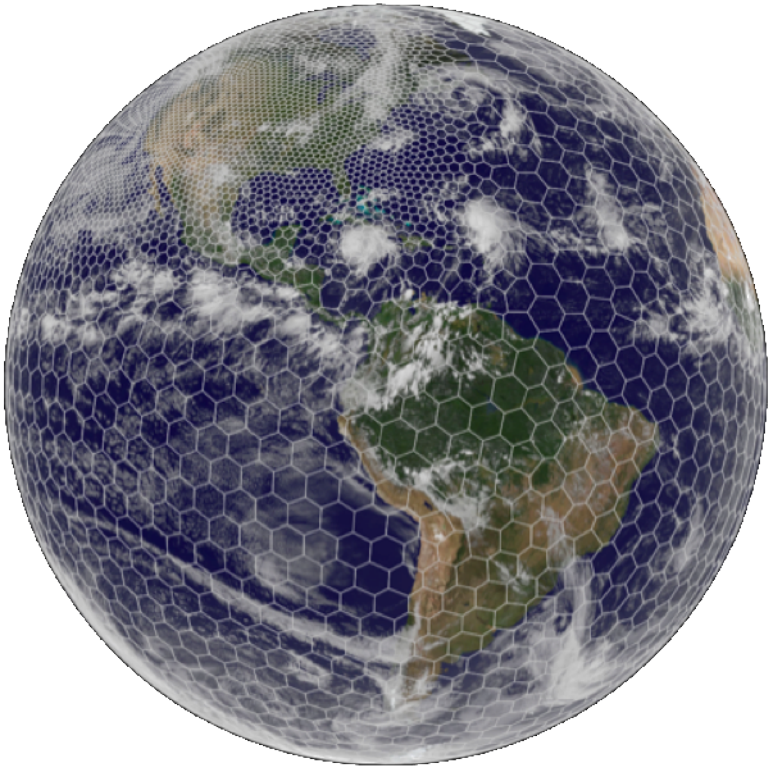


Schaer mountain wave



Supercell thunderstorm

OUTLINE



1. Preliminary requirements
2. Obtaining MPAS source code
3. Compiling MPAS-Atmosphere
4. Creating initial conditions
5. **Running the model**
6. Post-processing

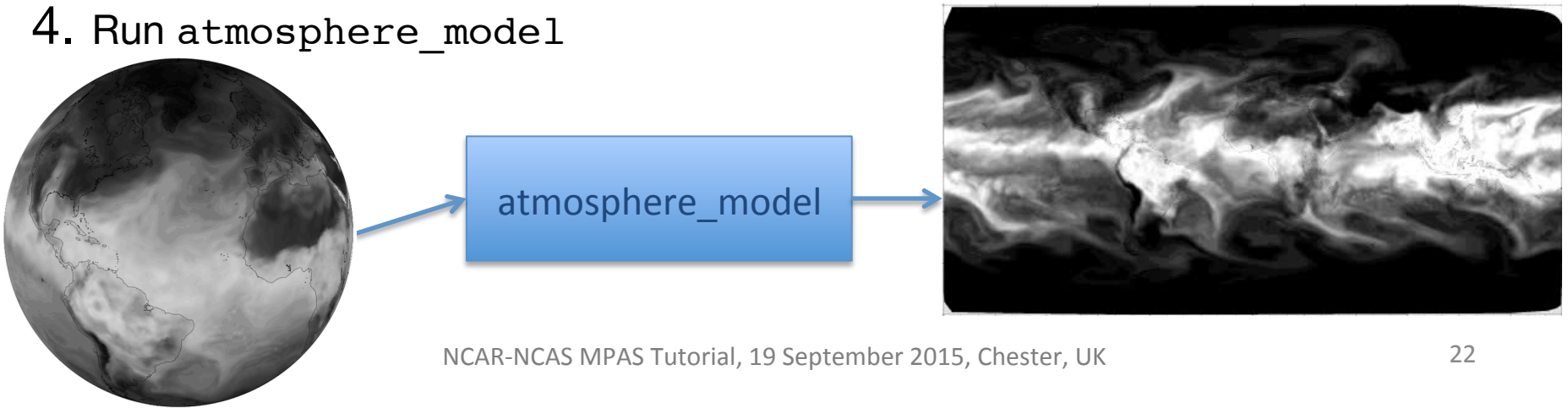
http://mpas-dev.github.io/

Running the MPAS-Atmosphere model

The same `atmosphere_model` executable can be used for either real-data or idealized simulations

Given initial conditions (e.g., `x1.40962.init.nc`), all that is needed to run the model is to:

1. Edit the `namelist.atmosphere` file to set model timestep, mixing and damping parameters, physics parameterizations, etc.
2. Edit the `streams.atmosphere` file to specify the name of the input initial conditions file and the frequency of model history files
3. Ensure that the proper mesh partition file (e.g., `x1.40962.graph.info.part.64`) is present
4. Run `atmosphere_model`



Running the MPAS-Atmosphere model (2)

Before running the model itself (atmosphere_model), verify that the following namelist options have been properly set:

- **config_dt –** The model timestep, in seconds; with MPAS v4.0, try starting with a timestep of between 5 and 6 times the minimum model grid spacing in kilometers; also ensure that model output interval is evenly divided by the timestep
- **config_len_disp –** The length-scale for explicit horizontal mixing; set this to the minimum grid distance (in meters) in the mesh

Besides these crucial namelist options, ensure that the names of input and output files are correctly set in the streams.atmosphere file!

Running the MPAS-Atmosphere model

As the model runs, information about the progress of the model is written to the file `log.0000.err`

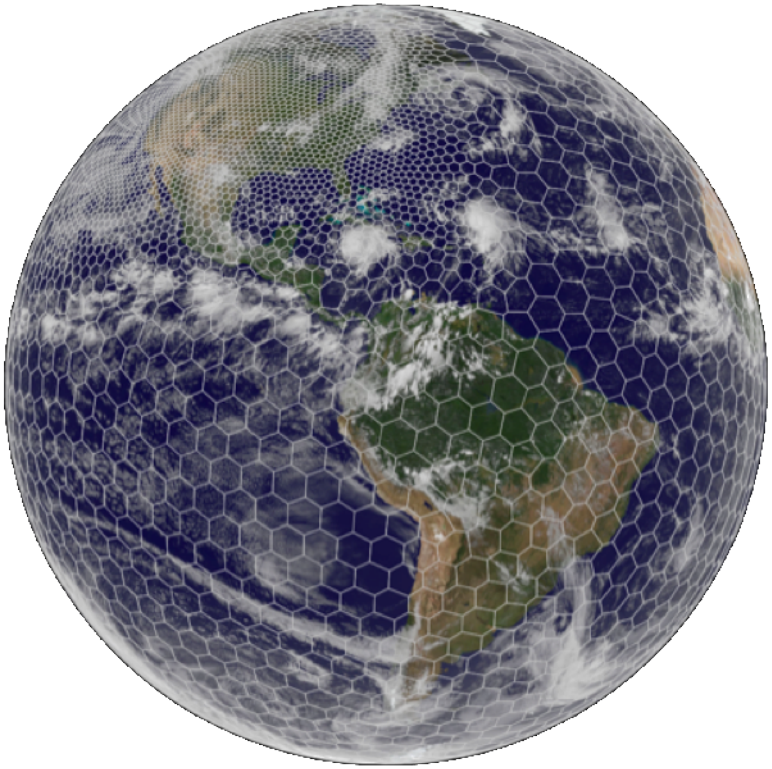
One can tail this file to check on model progress, e.g.,

```
$ tail -f log.0000.err
```

```
Begin timestep 2014-09-10_00:30:00
 --- time to run the LW radiation scheme L_RADLW = T
 --- time to run the SW radiation scheme L_RADSW = T
 --- time to run the convection scheme L_CONV    = T
 --- time to apply limit to accumulated rainc and rainnc L_ACRAIN   = F
 --- time to apply limit to accumulated radiation diags. L_ACRADT   = F
 --- time to calculate additional physics_diagnostics               = F

 global min, max w  -0.459342797817586        1.55674402225575
 global min, max u   -118.625775963647        118.161845710799
```

Above: Example output for a timestep in the log file from a typical model run.

OUTLINE



1. Preliminary requirements
2. Obtaining MPAS source code
3. Compiling MPAS-Atmosphere
4. Creating initial conditions
5. Running the model
6. **Post-processing**

http://mpas-dev.github.io/

Example NCL scripts

MPAS

mpas-dev.github.io

Apps

MPAS
Model for Prediction Across Scales

MPAS Atmosphere Public Releases

MPAS Home

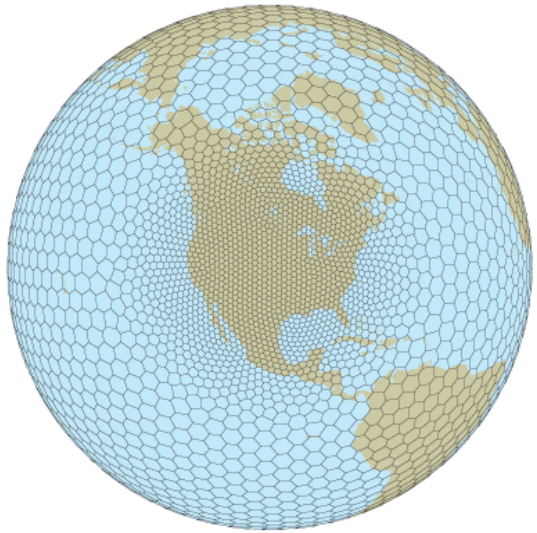
Overview
MPAS-Atmosphere
MPAS-Land Ice
MPAS-Ocean
Data Assimilation
Publications
Presentations

Download
MPAS-Atmosphere download
MPAS-Land Ice download
MPAS-Ocean download

Resources
License Information
Wiki
Bug Tracker
Mailing Lists
MPAS Developers Guide

MPAS Atmosphere 4.0 was released on 22 May 2015.

Any questions related to building and running MPAS-Atmosphere should be directed to the MPAS-Atmosphere Help forum. Posting to the forum requires a free google account. Alternatively, questions may be sent from any e-mail address to "mpas-atmosphere-help **AT** googlegroups.com". *Please note that in either case, questions and their answers will appear on the online forum.*

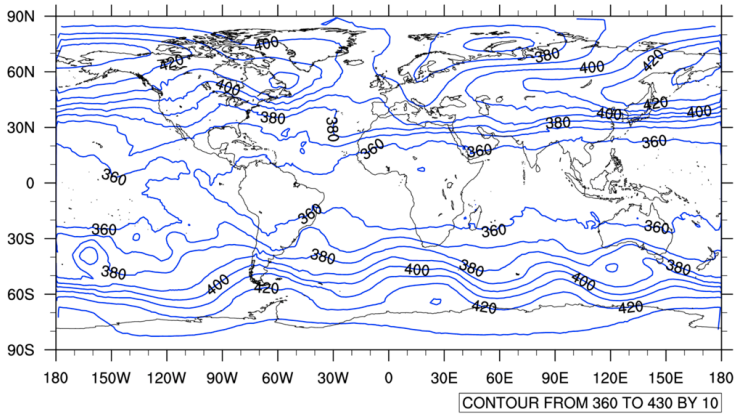


A variable resolution MPAS Voronoi mesh

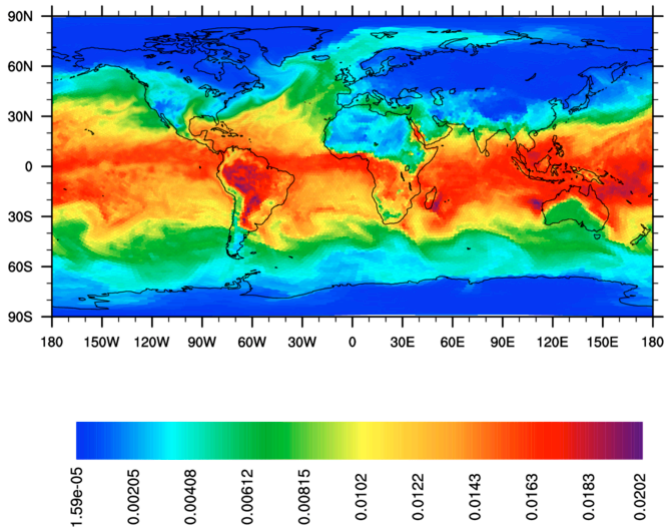
MPAS Atmosphere 4.0 release notes
MPAS source code download
MPAS-Atmosphere Users' Guide
MPAS-Atmosphere tutorial presentations
MPAS-Atmosphere meshes
Configurations for idealized test cases
Sample input files for real-data simulations
Visualization and analysis tools

The MPAS-Atmosphere download page contains a collection of example NCL scripts

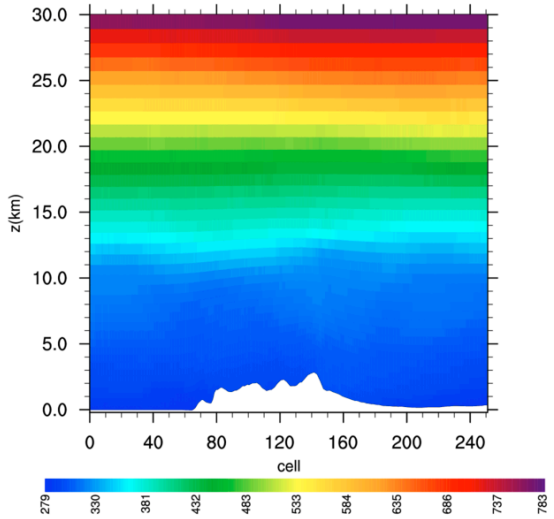
Example NCL scripts



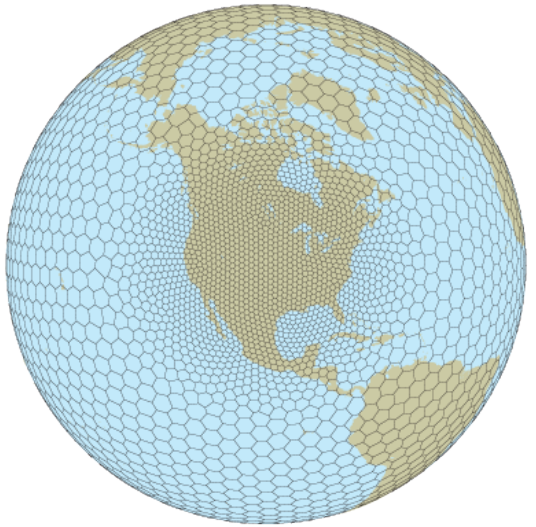
Contours – simple or color-filled



Individual grid cells as a color-filled polygons



Vertical cross-sections with specified endpoints



Primal mesh against a map background

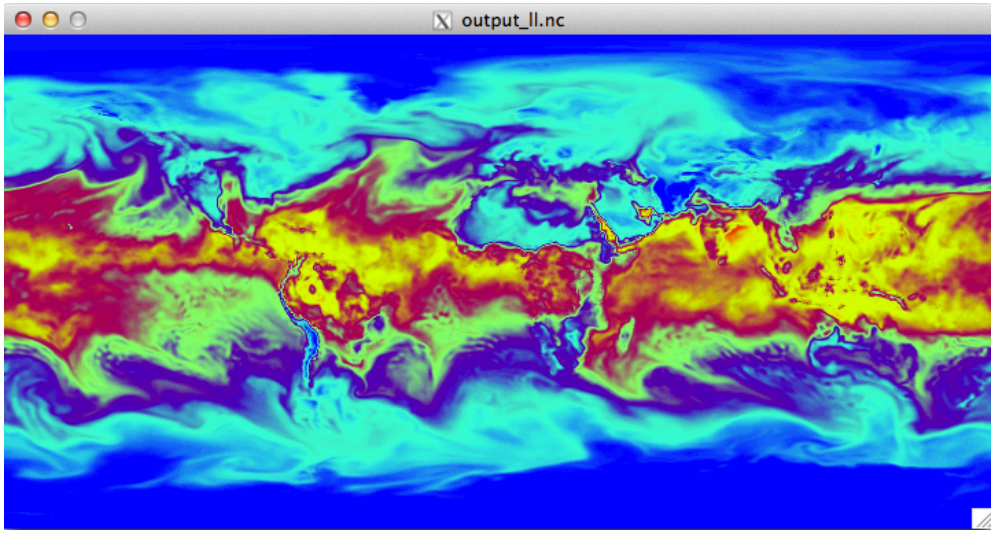
Interpolating to a lat-lon mesh

From the same page as other example NCL scripts, one can download a script, *mpas_to_latlon.ncl*, that interpolates cell-based fields to a regular latitude-longitude grid

- Allows for quick visualization of MPAS fields using, e.g., ncview
- Resolution of the lat-lon grid may be specified in the NCL script
- The MPAS field to be interpolated must also be specified
- Makes use of ESMF regridding routines, ***requires NCL 6.1.0 or later!***

Script requires two steps:

1) Generate remapping weights
    - These weights saved to a file
2) Remap fields using the pre-computed weights



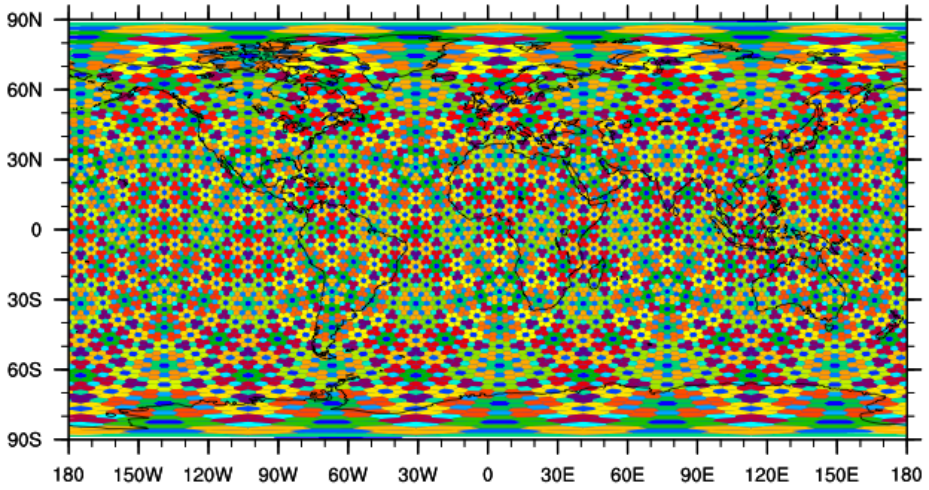
How to efficiently find the model cell closest to a given location?

In WRF, finding the nearest grid cell to a given (lat,lon) location is a constant-time operation:

1. Using the map projection equations for the WRF grid projection, compute the real-valued (x,y) coordinates of the (lat,lon) location
2. Round the real-valued coordinates to the nearest integer

However, in MPAS, *there is no projection*, and the horizontal cells may be indexed in any order.

- We could just compute the distance from (lat,lon) to every cell center in the mesh and choose the nearest cell, or we could do something more efficient...



Above: Cells in the x1.10242 mesh colored according to their global index

How to efficiently find the model cell closest to a given location?

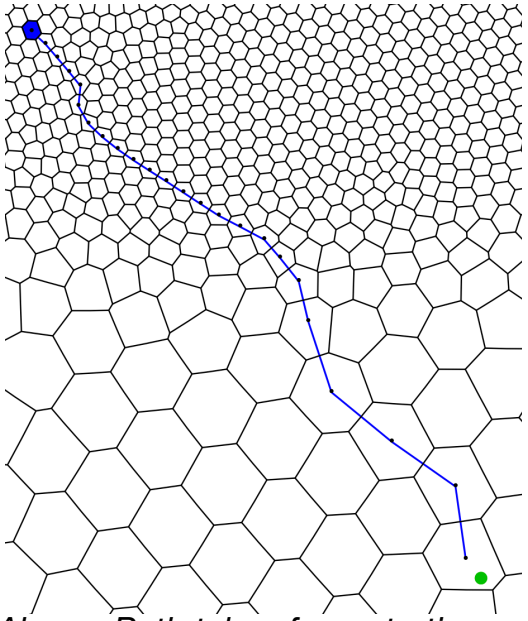
One solution would be to use search trees – perhaps a *kd*-tree – to store the cells in a mesh

- *O*(n log n) setup cost; each search takes O(log n) time, for a mesh with *n* cells

Alternatively, we can make use of the grid connectivity arrays nEdgesOnCell and cellsOnCell to navigate a path of monotonically decreasing distance to the (lat,lon) location

- No setup cost, at most $O(n^{1/2})$ cost* per search
- For repeated searches of "nearby" locations, almost constant cost!

```
C_nearest = any starting cell
C_test = NULL
do while (C_nearest ≠ C_test)
    C_test = C_nearest
    d = distance from C_test to (lat,lon)
    for i = 1 to nEdgesOnCell(C_test)
        k = cellsOnCell(i, C_test)
        d' = distance from k to (lat,lon)
        if ( d' < d )
            d = d'; C_nearest = k
```



Above: Path taken from starting cell (blue) to target location (green circle).

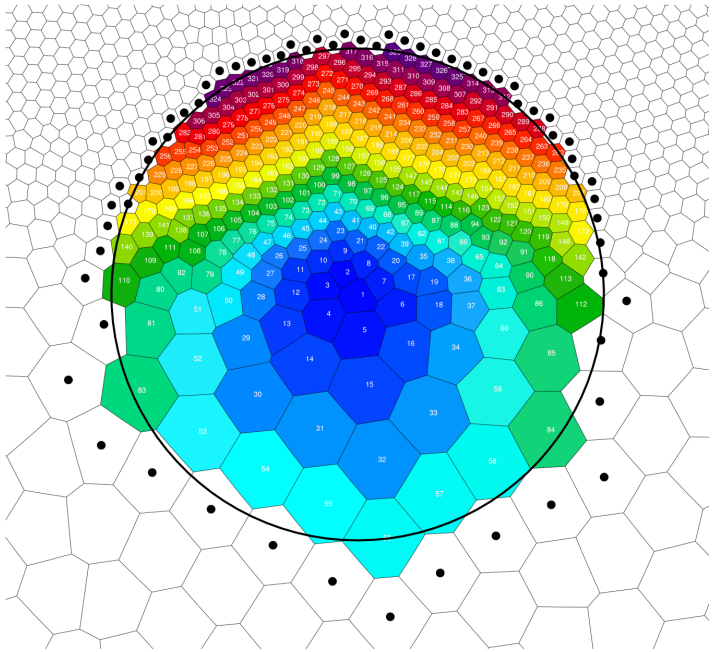
*At least, intuitively...

How to efficiently scan all cells within a specified radius of a location?

Similar to the problem of nearest grid cell, to scan all cells within a specified radius of a given (lat,lon) location, we could check all cells in the mesh...

... or we could make use of the connectivity arrays.

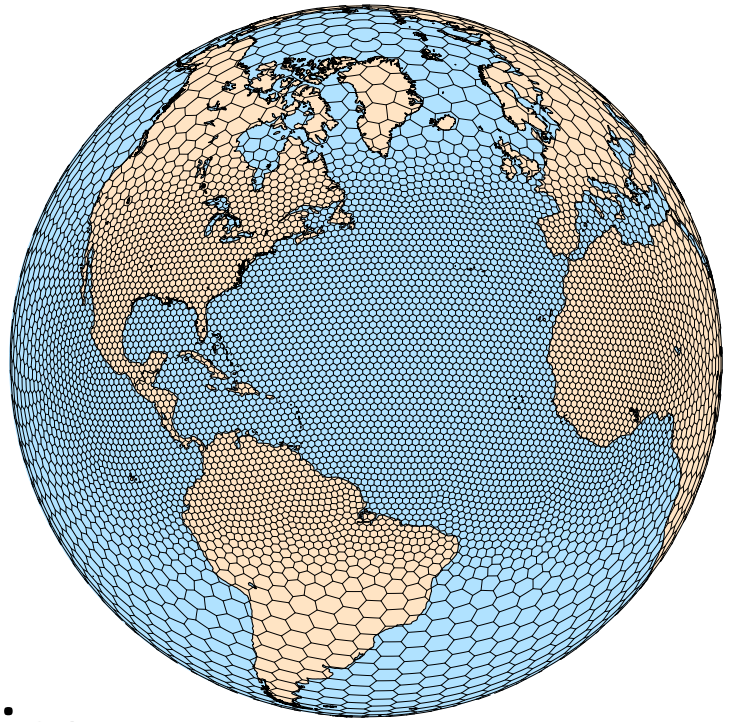
```
C = origin of the search
mark C as visited
insert C into the queue
do while (queue not empty)
      C = next cell from the queue

      C within search radius, so process C

      for i = 1 to nEdgesOnCell(C)
            k = cellsOnCell(i,C)
            if ( k not visited )
                  mark k as visited
                  if (k within search radius)
                        insert k into the queue
```



Above: Cells shaded according to the order in which *they were visited by a 750-km radius search; dots indicate cells that were considered but found to be at a radius >750 km.*

MPAS

Model for Prediction Across Scales



- Overview
- Meshes
- Atmospheric solver, physics
- Compiling and running MPAS
- *Summary*
- Practical session

Los Alamos NATIONAL LABORATORY EST. 1943

U.S. DEPARTMENT OF ENERGY | Office of Science

NSF

NCAR NATIONAL CENTER FOR ATMOSPHERIC RESEARCH