Running MPAS Part 2: Variable-resolution global meshes, I/O streams, restart runs, and other options

Michael G. Duda
NCAR/MMM

This material is based upon work supported by the NSF National Center for Atmospheric Research, a major facility sponsored by the U.S. National Science Foundation and managed by the University Corporation for Atmospheric Research. Any opinions, findings and conclusions or recommendations expressed in this material do not necessarily reflect the views of NSF.

MPAS
Model for Prediction Across Scales

Review

In the "Running MPAS Part 1" talk, we saw:

- How to interpolate time-invariant terrestrial fields to make a “static” file for real-data simulations
- How to interpolate meteorological and land-surface fields to produce real-data initial conditions
- How to produce SST and sea-ice update files
- How to run a simulation
- How to set up idealized test cases

Review

In the "Running MPAS Part 1" talk, we saw:

- How to interpolate time-invariant terrestrial fields to make a “static” file for real-data simulations
- How to interpolate meteorological and land-surface fields to produce real-data initial conditions
- How to produce SST and sea-ice update files
- How to set up idealized test cases

What we will cover in this talk:

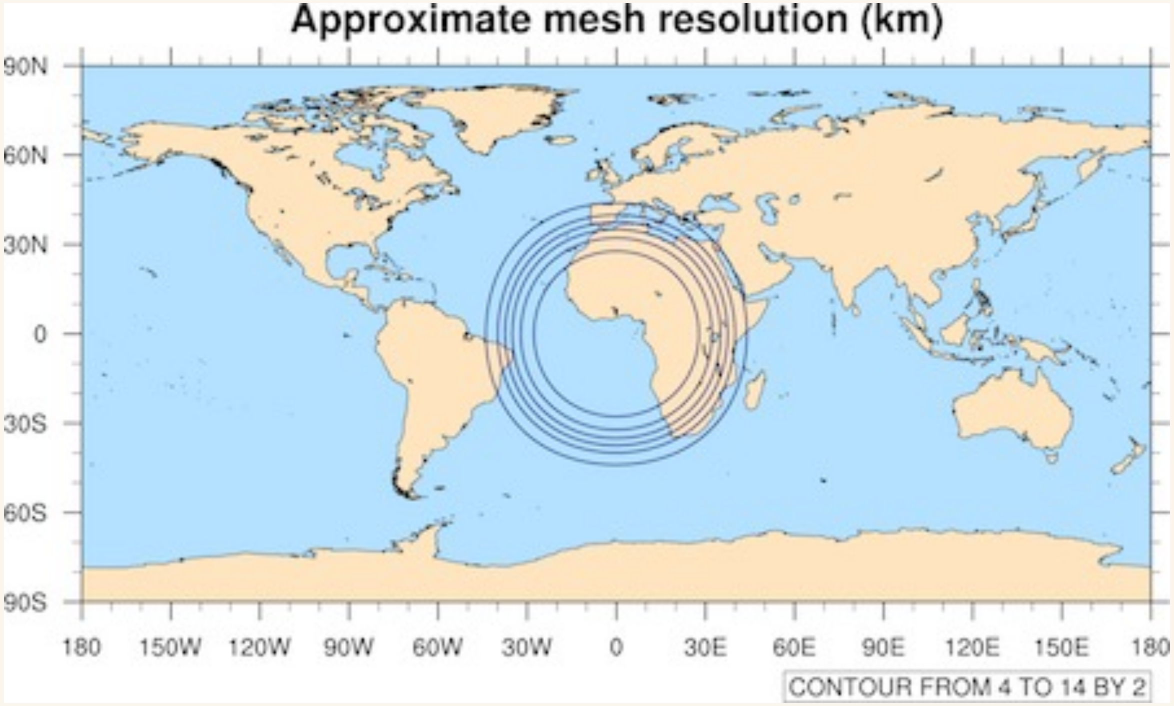
- How to work with variable-resolution meshes
- Details of the MPAS *streams* files
- How to restart a simulation from a previously saved checkpoint
- And a few other model options…

Outline

1. How to work with variable-resolution meshes
2. Details of the MPAS *streams* files
3. How to restart a simulation from a previously saved checkpoint
4. And a few other model options…

Variable-resolution meshes

You might expect that generating a variable-resolution mesh is a simple matter...



Left: Contours of horizontal grid distance for a variable-resolution, 15 km – 3 km MPAS mesh

... but some meshes have taken *months* to generate using our current software on a desktop system

- *So, we'd like to re-use meshes whenever possible!*

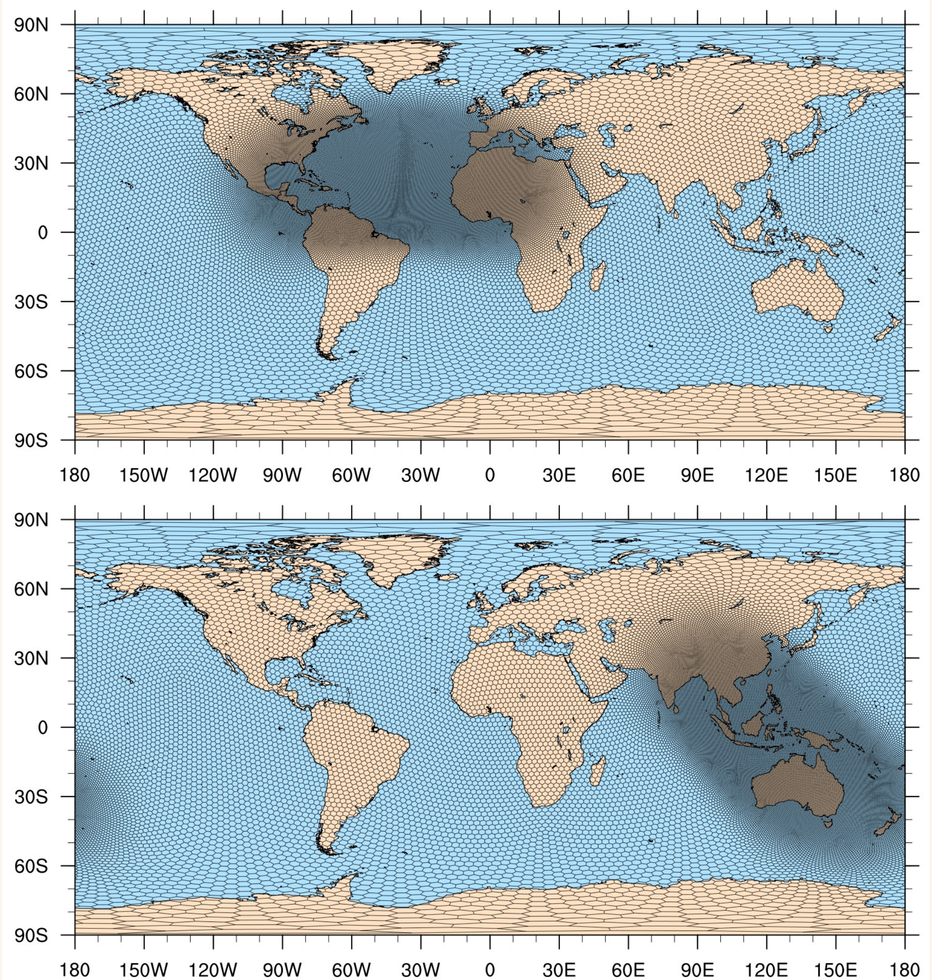
Variable-resolution meshes

The key idea for re-using variable-resolution meshes is to rotate the refined region

This may be accomplished easily (and quickly!) using the "grid_rotate" tool

- Implements two solid-body rotations for spherical meshes:

1. Move center of refined region from one location to another
2. Rotate the relocated refinement about its center to change orientation

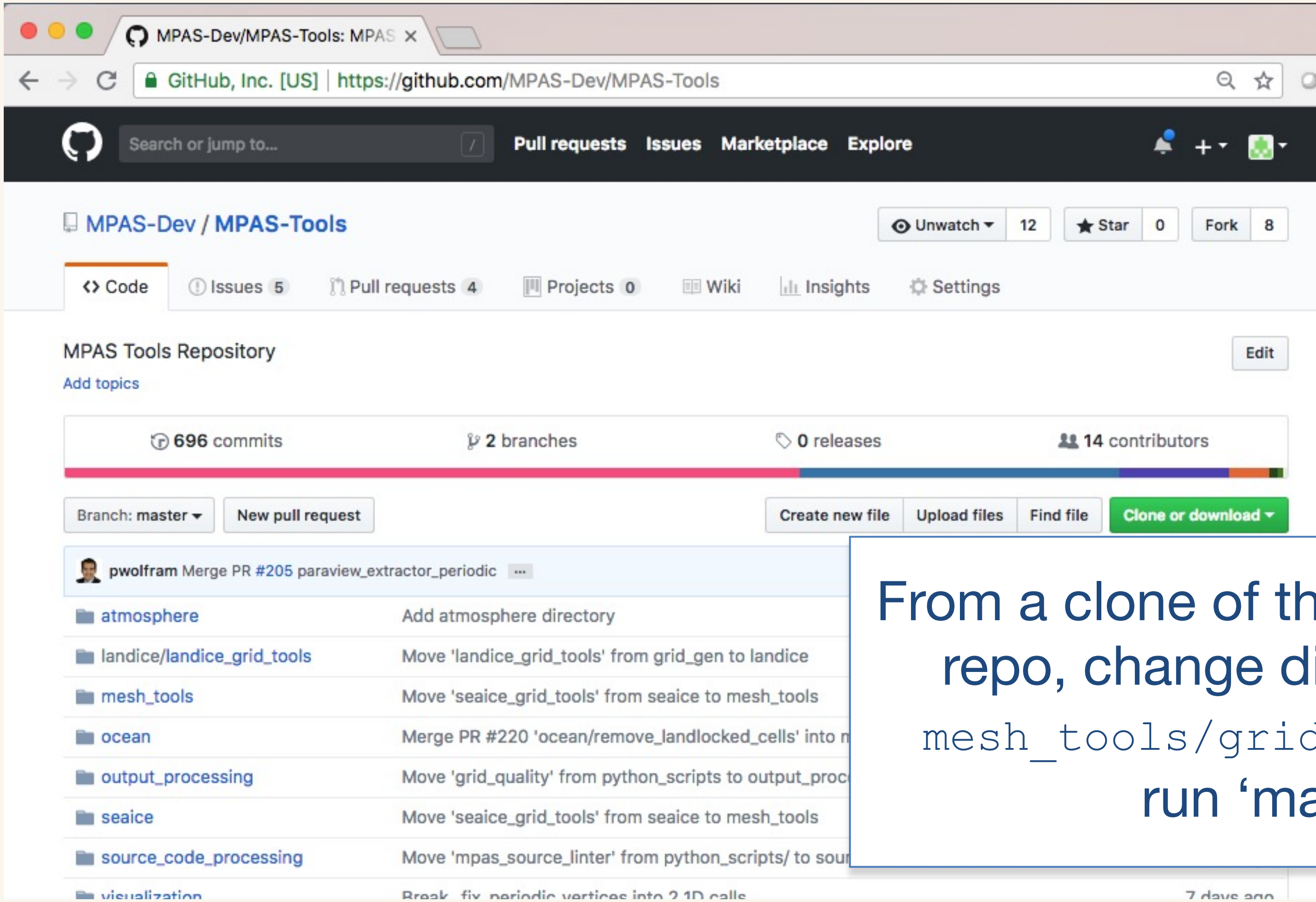


Above: A refinement region originally centered at 25N, 40W has been shifted to 7S, 125E and rotated by -45 degrees.

MPAS
Model for Prediction Across Scales

Variable-resolution meshes

The `grid_rotate` tool is available in a GitHub repository at https://github.com/MPAS-Dev/MPAS-Tools.git



From a clone of the MPAS-Tools repo, change directories to `mesh_tools/grid_rotate` and run 'make'

Variable-resolution meshes

The `grid_rotate` tool uses a Fortran namelist file to control rotation of the mesh:

```
&input
   config_original_latitude_degrees = 0.0
   config_original_longitude_degrees = 0.0

   config_new_latitude_degrees = -19.5
   config_new_longitude_degrees = -62.0
   config_birdseye_rotation_counter_clockwise_degrees = 90
/
```

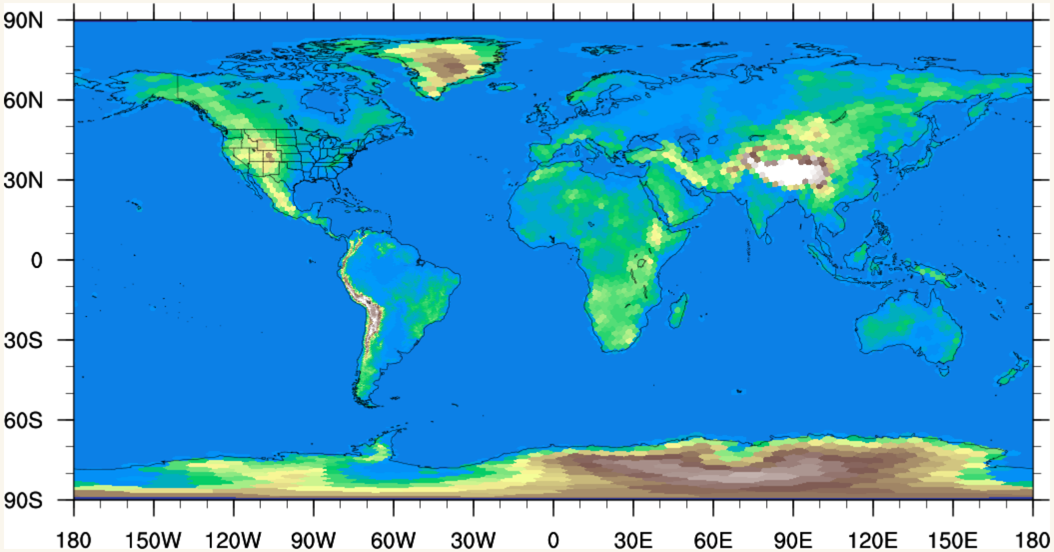
Typical usage might look like:

```
grid_rotate x5.30210.grid.nc SouthAmerica.grid.nc
```

Variable-resolution meshes

After rotating a variable-resolution mesh, one can produce a "static" file for real-data simulations, or, e.g., baroclinic wave idealized ICs, as usual

Right: Terrain field for a variable-resolution, 240 km – 48 km MPAS mesh with refinement over South America



When running MPAS-A, be sure to set:

- **config_dt** appropriately for the finest-resolution part of the mesh

Outline

1. How to work with variable-resolution meshes
2. Details of the MPAS *streams* files
3. How to restart a simulation from a previously saved checkpoint
4. And a few other model options…

MPAS
Model for Prediction Across Scales

MPAS I/O "Streams"

Recall that we used the `streams.atmosphere` file to set the names of the input and output files for the MPAS-Atmosphere model:

```
<immutable_stream name="input"
                  type="input"
                  filename_template="x1.10242.init.nc"
                  input_interval="initial_only" />

<immutable_stream name="restart"
                  type="input;output"
                  filename_template="restart.$Y-$M-$D_$h.$m.$s.nc"
                  input_interval="initial_only"
                  output_interval="1_00:00:00" />

<stream name="output"
        type="output"
        filename_template="history.$Y-$M-$D_$h.$m.$s.nc"
        output_interval="6:00:00" >

        <file name="stream_list.atmosphere.output"/>
</stream>
```

MPAS I/O “Streams”

An XML file seems overly complicated for setting the names of input and output files…

- You may begin to suspect that the *streams* files are capable of a little more than this

Chapter 5

Configuring Model Input and Output

The reading and writing of model fields in MPAS is handled by user-configurable *streams*. A st
a fixed set of model fields, together with dimensions and attributes, that are all written o
to or from the same file or set of files. Each MPAS model core may define its own set of
that it typically uses for reading initial conditions, for writing and reading restart fields,
additional model history fields. Besides these default streams, users may define new stream
certain diagnostic fields at a higher temporal frequency than the usual model history fields.
Streams are defined in XML configuration files that are created at build time for each m
name of this XML file is simply ‘streams.’ suffixed with the name of the core. For example,
the *atmosphere* core are defined in a file named ‘streams.atmosphere’, and the streams for the
core are defined in a file named ‘streams.init_atmosphere’. An XML stream file may further
text files that contain lists of the model fields that are read or written in each of the stream
XML stream file.
Changes to the XML stream configuration file will take effect the next time an MPAS co
is no need to re-compile after making modifications to the XML files. As described in the ne
therefore possible, e.g., to change the interval at which a stream is written, the template fo

Chapter 5 of the MPAS-A Users’ Guide describes the complete functionality provided by *streams* files

MPAS
Model for Prediction Across Scales

Anatomy of an XML streams file

An example streams configuration file:

```
<streams>
<immutable_stream name="input"
                  type="input"
                  filename_template="x1.4002.init.nc"
                  input_interval="initial_only" />

<immutable_stream name="restart"
                  type="input;output"
                  filename_template="restart.$Y-$M-$D_$h.nc"
                  input_interval="initial_only"
                  output_interval="1_00:00:00" />

<stream name="output"
        type="output"
        filename_template="history.$Y-$M-$D_$h.$m.$s.nc"
        output_interval="6:00:00" >

        <var name=”mslp"/>
        <var name=”height_500hPa"/>
        <var name=”rainc"/>
        <var name=”rainnc"/>
</stream>
</streams>
```

Anatomy of an XML streams file

An example streams configuration file:

```
<streams>
<immutable_stream name="input"
                  type="input"
                  filename_template="x1.4002.init.nc"
                  input_interval="initial_only" />

<immutable_stream name="restart"
                  type="input;output"
                  filename_template="restart.$Y-$M-$D_$h.nc"
                  input_interval="initial_only"
                  output_interval="1_00:00:00" />

<stream name="output"
        type="output"
        filename_template="history.$Y-$M-$D_$h.$m.$s.nc"
        output_interval="6:00:00" >

        <var name=”mslp"/>
        <var name=”height_500hPa"/>
        <var name=”rainc"/>
        <var name=”rainnc"/>
</stream>
</streams>
```

The variables read/written by immutable streams may not be changed at runtime

This is the set of variables written by this stream

Anatomy of an XML streams file

An example streams configuration file:

```
<streams>
<immutable_stream name="input"
                  type="input"
                  filename_template="x1.4002.init.nc"
                  input_interval="initial_only" />

<immutable_stream name="restart"
                  type="input;output"
                  filename_template="restart.$Y-$M-$D_$h.nc"
                  input_interval="initial_only"
                  output_interval="1_00:00:00" />

<stream name="output"
        type="output"
        filename_template="history.$Y-$M-$D_$h.$m.$s.nc"
        output_interval="6:00:00" >

        <var name="mslp"/>
        <var name="height_500hPa"/>
        <var name="rainc"/>
        <var name="rainnc"/>
</stream>
</streams>
```

This stream is named "input"

This stream is named "restart"

This stream is named "output"

Anatomy of an XML streams file

An example streams configuration file:

```
<streams>
<immutable_stream name="input"
                  type="input"
                  filename_template="x1.4002.init.nc"
                  input_interval="initial_only" />

<immutable_stream name="restart"
                  type="input;output"
                  filename_template="restart.$Y-$M-$D_$h.nc"
                  input_interval="initial_only"
                  output_interval="1_00:00:00" />

<stream name="output"
        type="output"
        filename_template="history.$Y-$M-$D_$h.$m.$s.nc"
        output_interval="6:00:00" >

        <var name="mslp"/>
        <var name="height_500hPa"/>
        <var name="rainc"/>
        <var name="rainnc"/>
</stream>
</streams>
```

This stream is only read by MPAS

This stream is both read and written

This stream is only written

Anatomy of an XML streams file

An example streams configuration file:

```
<streams>
<immutable_stream name="input"
                  type="input"
                  filename_template="x1.4002.init.nc"
                  input_interval="initial_only" />

<immutable_stream name="restart"
                  type="input;output"
                  filename_template="restart.$Y-$M-$D_$h.nc"
                  input_interval="initial_only"
                  output_interval="1_00:00:00" />

<stream name="output"
        type="output"
        filename_template="history.$Y-$M-$D_$h.$m.$s.nc"
        output_interval="6:00:00" >

        <var name="mslp"/>
        <var name="height_500hPa"/>
        <var name="rainc"/>
        <var name="rainnc"/>
</stream>
</streams>
```

This stream reads from a file named "x1.4002.init.nc"

This stream reads and writes from files with names of this form

This stream writes to files with names of this form

MPAS
Model for Prediction Across Scales

Anatomy of an XML streams file

An example streams configuration file:

```
<streams>
<immutable_stream name="input"
                  type="input"
                  filename_template="x1.4002.init.nc"
                  input_interval="initial_only" />

<immutable_stream name="restart"
                  type="input;output"
                  filename_template="restart.$Y-$M-$D_$h.nc"
                  input_interval="initial_only"
                  output_interval="1_00:00:00" />

<stream name="output"
        type="output"
        filename_template="history.$Y-$M-$D_$h.$m.$s.nc"
        output_interval="6:00:00" >

        <var name=”mslp"/>
        <var name=”height_500hPa"/>
        <var name=”rainc"/>
        <var name=”rainnc"/>
</stream>
</streams>
```

This stream is read only at the start of execution

This stream is read only at the start of execution

This stream is written every 1 day

This stream is written every 6 hours

Anatomy of an XML streams file

At runtime, it's easy to define a new output stream!

```
<stream name="sfcwinds"
        type="output"
        filename_template="winds.$Y$M$D$h$m.nc"
        filename_interval="24:00:00"
        output_interval="0:30:00" >

        <var name="u10"/>
        <var name="v10"/>
</stream>
```

Anatomy of an XML streams file

At runtime, it's easy to define a new output stream!

Define a new stream with <stream> … </stream> *tags and give the stream a unique name*

```
<stream name="sfcwinds"
        type="output"
        filename_template="winds.$Y$M$D$h$m.nc"
        filename_interval="24:00:00"
        output_interval="0:30:00" >

        <var name="u10"/>
        <var name="v10"/>
</stream>
```

Anatomy of an XML streams file

At runtime, it's easy to define a new output stream!

Set the type of the stream to "output"

```
<stream name="sfcwinds"
        type="output"
        filename_template="winds.$Y$M$D$h$m.nc"
        filename_interval="24:00:00"
        output_interval="0:30:00" >

        <var name="u10"/>
        <var name="v10"/>
</stream>
```

MPAS
Model for Prediction Across Scales

Anatomy of an XML streams file

At runtime, it's easy to define a new output stream!

Provide a filename template to be use for the output files. Possible variables include:

`$Y`	= *year*	`$h`	= *hour*
`$M`	= *month*	`$m`	= *minute*
`$D`	= *day of month*	`$s`	= *second*
`$d`	= *day of year*		

```
<stream name="sfcwinds"
        type="output"
        filename_template="winds.$Y$M$D$h$m.nc"
        filename_interval="24:00:00"
        output_interval="0:30:00" >

        <var name="u10"/>
        <var name="v10"/>
</stream>
```

Anatomy of an XML streams file

At runtime, it's easy to define a new output stream!

Specify how often the stream will be written. Time formats can be "ss", "mm:ss", "hh:mm:ss", or "ddd_hh:mm:ss". A value of "none" means the stream is effectively deactivated (it is never written).

```
<stream name="sfcwinds"
        type="output"
        filename_template="winds.$Y$M$D$h$m.nc"
        filename_interval="24:00:00"
        output_interval="0:30:00" >

        <var name="u10"/>
        <var name="v10"/>
</stream>
```

Anatomy of an XML streams file

At runtime, it's easy to define a new output stream!

Optionally, specify how often one output file should be closed and a new one opened. The default is to place all output records into separate files (i.e., the filename interval is the same as the output interval).

```
<stream name=”sfcwinds"
        type="output"
        filename_template=”winds.$Y$M$D$h$m.nc”
        filename_interval=”24:00:00”
        output_interval=”0:30:00" >

        <var name=”u10"/>
        <var name=”v10"/>
</stream>
```

Anatomy of an XML streams file

At runtime, it's easy to define a new output stream!

List one or more variables to be written to the stream as <var/> XML tags.

```
<stream name=”sfcwinds"
        type="output"
        filename_template=”winds.$Y$M$D$h$m.nc”
        filename_interval=”24:00:00”
        output_interval=”0:30:00" >

        <var name=”u10"/>
        <var name=”v10"/>
</stream>
```

Anatomy of an XML streams file

Input and output streams may contain any field defined in the MPAS `Registry.xml` file

Appendix D of the MPAS-A Users' Guide lists every field available for input/output

Appendix D

Description of Model Fields

Every field that may be read or written in a NetCDF *stream* (as described in Chapter 5) by the MPAS-Atmosphere model is described in this chapter. The dimensionality of each field is given in Fortran storage order (i.e., the fastest-varying dimension is inner-most).

a_tri (real) (nVertLevels, nCells, Time)

Units	*unitless*
Description	*implicit tridiagonal solve coefficients*
Accessed in code	as 'a_tri' from the 'diag' pool

absnxt (real) (nVertLevels, cam_dim1, nCells, Time)

Units	-
Description	*Total nearest layer absorptivity*
Accessed in code	as 'absnxt' from the 'diag_physics' pool

Outline

1. How to work with variable-resolution meshes
2. Details of the MPAS *streams* files
3. How to restart a simulation from a previously saved checkpoint
4. And a few other model options…

Restarting a simulation

Saving checkpoints (restart state) periodically during a simulation is as easy as setting an output interval for the "restart" stream:

```
<immutable_stream name="restart"
                  type="input;output"
                  filename_template="restart.$Y-$M-$D_$h.$m.$s.nc"
                  input_interval="initial_only"
                  output_interval="1_00:00:00" />
```

Note that the "restart" stream is both an "input" and an "output" stream:

- Read if we are performing a restart simulation
- Written periodically during a simulation

Restarting a simulation

Restarting a simulation from any existing restart file requires two namelist changes:

```
&nhyd_model
    config_start_time = "2014-09-11_00:00:00"
/
```

The time from which we wish to restart the simulation

```
&restart
    config_do_restart = true
/
```

Outline

1. How to work with variable-resolution meshes
2. Details of the MPAS *streams* files
3. How to restart a simulation from a previously saved checkpoint
4. And a few other model options…

Other options: detailed printout

More information on the locations of the min/max horizontal and vertical velocities in the simulation can be requested:

```
&printout
    config_print_detailed_minmax_vel = true
/
```

```
Begin timestep 2014-09-10_00:16:00
...

  global min w: -1.03829 k=14, -27.5557 lat, -68.5647 lon
  global max w: 0.757052 k=19, -34.0048 lat, -52.0361 lon
  global min u: -117.846 k=41, -69.4637 lat, 135.753 lon
  global max u: 118.322 k=41, -69.6092 lat, 129.425 lon
  global max wsp: 118.366 k=41, -69.6092 lat, 129.425 lon
  Timing for integration step: 3.15425 s
```

Other options: physics suites

In MPAS v8.1, there are two *suites* of physics:

```
&physics
    config_physics_suite = 'mesoscale_reference'
/
```

```
&physics
    config_physics_suite = 'convection_permitting'
/
```

We'll say more about physics options in the physics lecture

Note: before running the 'convection_permitting' suite for the first time, you'll need to generate look-up tables for the Thompson microphysics with the `build_tables` *utility.*

Other options: soundings

It's also possible to write out soundings from the model grid cells that contain specified (lat,lon) locations

1) Create a text file named `sounding_locations.txt` with a list of sounding locations and names (latitude longitude name)

```
 40.0   -105.25   Boulder
 28.7     77.2    NewDelhi
-77.85   166.67   McMurdo
```

2) In the `namelist.atmosphere` file, select the interval at which soundings will be written from the model

```
&soundings
    config_sounding_interval = '1:00:00'
/
```

3) Sounding text files will be written as `<name>.YYYYMMDDhhmmss.snd`