MPAS-Atmosphere Model User's Guide

Version 4.0

May 20, 2015

Foreword

This user's guide describes the Model for Prediction Across Scales – Atmosphere (MPAS-A) Version 4.0. MPAS-A is the non-hydrostatic atmosphere model built within the MPAS framework. Users guides for other MPAS components, such as MPAS-Ocean, are separate from this guide.

The component models and framework that comprise MPAS are being developed collaboratively between Los Alamos National Laboratory (LANL) and the National Center for Atmospheric Research (NCAR). Common functionality required by different MPAS component models, such as parallel input/output, time management, block decomposition, etc., is provided by the MPAS framework, while development of specific component models, referred to in MPAS as *cores*, is handled by the individual development groups. Currently, LANL is responsible for the ocean core, MPAS-O, and NCAR is responsible for the atmospheric core, MPAS-A.

MPAS is very much a collaborative development of both the shared architectures and the component models. There are a number of contributors to the developments leading to the MPAS-A solver, and many of these developments are shared with the ocean core. The C-grid Voronoi discretization is based on critical developments from John Thuburn, Todd Ringler, Bill Skamarock, and Joe Klemp. The mesh generation that enables the MPAS-A development received major contributions from Todd Ringler, Doug Jacobson, Max Gunzburger, and Lili Ju. Significant developments in the transport scheme were accomplished by Bill Skamarock and Almut Gassmann. The atmospheric physics has been taken from the Advanced Research WRF model; these physics have benefitted from the work of hundreds of scientists. On the framework side we leverage a number of outside packages that have received extensive development from a wide community, including NetCDF (UCAR/Unidata) and PIO (as used in the Community Earth Systems Model, CESM).

The software developed for MPAS is open source, and it has been copyrighted under a BSD license. The simple copyright statement can be found at the beginning of MPAS source files and the complete copyright statement can be found in this user's guide or in the 'LICENSE' file accompanying the source code.

We conclude by noting that this user's guide is a work in progress. We welcome suggestions for improvements to this guide, including additions, corrections, clarifications, etc. Updates to MPAS-A, including the most recent code, user's guide, and test cases, may be found at `http://mpas-dev.github.com`.

Contributors to this guide:
Michael Duda, Laura Fowler, Bill Skamarock, Conrad Roesch, Doug Jacobsen, and Todd Ringler.

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

Contents

Chapter 1

MPAS-Atmosphere Overview

The Model for Prediction Across Scales – Atmosphere (MPAS-A) is a non-hydrostatic atmosphere model that is part of a family of Earth-system component models collectively known as MPAS. All MPAS models have in common their use of centroidal Voronoi tessellations for their horizontal meshes, which has motivated the development of a common software framework that provides a high-level driver program and infrastructure for providing parallel execution, input and output, and other software infrastructure.

1.1 Features

Important features of MPAS-A include:

- Fully-compressible, non-hydrostatic dynamics
- Split-explicit Runge-Kutta time integration
- Exact conservation of dry-air mass and scalar mass
- Positive-definite and monotonic transport options
- Generalized terrain-following height coordinate
- Support for unstructured variable-resolution (horizontal) mesh integrations for the sphere and Cartesian planes.

At present, MPAS-A includes parameterizations of physical processes taken from the Weather Research and Forecasting (WRF) Model [1]. Specifically, MPAS-A has support for:

- Radiation: CAM and RRTMG long-wave and short-wave radiation schemes
- Land-surface: NOAH land-surface model
- Surface-layer: Monin-Obukhov
- Boundary-layer: YSU PBL scheme
- Convection: Kain-Fritsch and Tiedtke convection parameterizations
- Cloud microphysics: WSM6 and Kessler schemes

[1] `http://www.wrf-model.org/`.

1.2 Model components

MPAS-A is comprised of two main components: the model, which includes atmospheric dynamics and physics; and an initialization component for generating initial conditions for the atmospheric and land-surface state as well as update files for sea-surface temperature and sea ice. Both components (model and initialization) are built as *cores* within the MPAS software framework and make use of the same driver program and software infrastructure. However, each component is compiled as a separate executable.

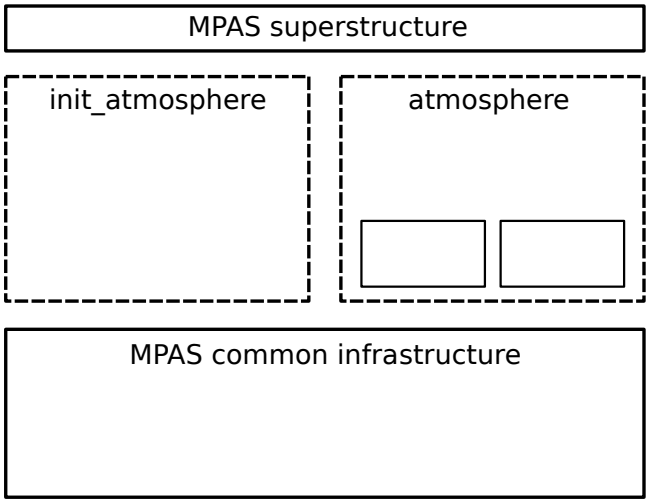


Figure 1.1: The initialization and model components of MPAS-A are built as separate *cores* within the MPAS framework.

A succinct description of building and running MPAS-A is given in Chapter 2, the Quick Start Guide. Detailed instructions for building these components are given in Chapter 3, and the basic steps to create initial conditions and run the MPAS-A model are outlined in Chapter 7.

Chapter 2

MPAS-Atmosphere Quick Start Guide

This chapter provides MPAS-Atmosphere users with a quick start description of how to build and run the model. It is meant merely as a brief overview of the process, while the more detailed descriptions of each step are provided in later chapters.

In general, the build process follows the following steps.

1. Build or locate an implementation of MPI (MPICH, OpenMPI, MVAPICH2, etc; Section 3.1).
2. Build the serial NetCDF library (Section 3.2.1).
3. Build the Parallel-NetCDF library (Section 3.2.2).
4. Build the Parallel I/O library (Section 3.2.3).
5. **Optionally,** build the METIS package (Section 4.1).
6. Obtain the MPAS source code.
7. Build the *init_atmosphere* and *atmosphere* cores (Section 3.3).

After completing these steps, executable files named `init_atmosphere_model` and `atmosphere_model` should have been created in the top-level MPAS directory. Once both executables have been created, a complete model run can be completed with the following steps:

1. Create a run directory.
2. Link the `init_atmosphere_model` and `atmosphere_model` executables to the run directory, as well as physics lookup tables (`*TBL`, `*DBL`, `*DATA`).
3. Copy the `namelist.*`, `streams.*`, and `stream_list.*` files to the run directory.
4. Edit the namelist files and the stream files appropriately (Chapter 7).
5. **Optionally,** prepare meshes for the simulation (Chapter 4).
6. Run `init_atmosphere_model` to create initial conditions, then run `atmosphere_model` to perform model integration.
7. Visualize output, and perform analyses.

Chapter 3

Building MPAS

3.1 Prequisites

To build MPAS, compatible C and Fortran compilers are required. Additionally, the MPAS software relies on the PIO parallel I/O library to read and write model fields, and the PIO library requires the standard NetCDF library as well as the Parallel-NetCDF library from Argonne National Labs. All libraries must be compiled with the same compilers that will be used to build MPAS. Section 3.2 summarizes the basic procedure of installing the required I/O libraries for MPAS.

In order for the MPAS makefiles to find the PIO, Parallel-NetCDF, and NetCDF include files and libraries, the environment variables `PIO`, `PNETCDF`, and `NETCDF` should be set to the root installation directories of the PIO, Parallel-NetCDF, and NetCDF installations, respectively.

An MPI installation such as MPICH or OpenMPI is also required, and there is no option to build a serial version of the MPAS executables. There is currently no support for shared-memory parallelism with OpenMP within the MPAS framework.

3.2 Compiling I/O Libraries

IMPORTANT NOTE: *The instructions provided in this section for installing libraries have been successfully used by MPAS developers, but due to differences in library versions, compilers, and system configurations, it is recommended that users consult documentation provided by individual library vendors should problems arise during installation. The MPAS developers take no responsibility for third-party libraries.*

Although most recent versions of the I/O libraries should work, the most tested versions of these libraries are: NetCDF 4.1.3, Parallel-NetCDF 1.3.1, and PIO 1.7.1. The NetCDF and Parallel-NetCDF libraries must be installed before building PIO library.

3.2.1 NetCDF

Version 4.1.3 of the NetCDF library may be downloaded from `http://www.unidata.ucar.edu/downloads/netcdf/netcdf-4_1_3/index.jsp`. Assuming the gfortran and gcc compilers will be used, the following shell commands (in csh/tcsh) are generally sufficient to install NetCDF. Of course, other compilers may be used by making suitable substitutions in the commands, below.

```
> setenv FC gfortran
> setenv F77 gfortran
```

```
> setenv F90 gfortran
> setenv CC gcc
> ./configure --prefix=XXXXX --disable-dap --disable-netcdf-4 --disable-cxx
--disable-shared --enable-fortran
> make all check
> make install
```

Here, `XXXXX` should be replaced with the directory that will serve as the root installation directory for NetCDF. *Before proceeding to compile PIO the* `NETCDF_PATH` *environment variable should be set to the NetCDF root installation directory.*

Certain compilers require addition flags in the CPPFLAGS environment variable. Please refer to the NetCDF installation instructions for these flags.

3.2.2 Parallel-NetCDF

Version 1.3.1 of the Parallel-NetCDF library may be downloaded from `https://trac.mcs.anl.gov/projects/parallel-netcdf/wiki/Download`. Again, assuming the gfortran and gcc compilers will be used, the following shell commands are generally sufficient to install Parallel-NetCDF; environment variables set during the installation of the NetCDF library are assumed to be already set.

```
> setenv MPIF90 mpif90
> setenv MPIF77 mpif90
> setenv MPICC mpicc
> ./configure --prefix=XXXXX
> make
> make install
```

Here, `XXXXX` should be replaced with the directory that will serve as the root installation directory for Parallel-NetCDF. *Before proceeding to compile PIO the* `PNETCDF_PATH` *environment variable should be set to the Parallel-NetCDF root installation directory.*

3.2.3 PIO

Due to the rapid development pace of the PIO library, it is recommended to obtain and install PIO following the instructions at `http://www.cesm.ucar.edu/models/pio/` for building PIO.

After PIO is built and installed the `PIO` enviroment variable should be set to the directory where PIO was installed. Recent versions of PIO support a `--prefix` option, which specifies the installation directory, while older versions do not, in which case the `PIO` environment variable should be set to the directory where PIO was compiled.

3.3 Compiling MPAS

IMPORTANT NOTE: *Before compiling MPAS, the* `NETCDF`*,* `PNETCDF`*, and* `PIO` *environment variables must be set to the library installation directories as described in the previous section.*

The MPAS code uses only the 'make' utility for compilation. Rather than employing a separate configuration step before building the code, all information about compilers, compiler flags, etc.,

is contained in the top-level `Makefile`; each supported combination of compilers (i.e., a configuration) is included in the `Makefile` as a separate make target, and the user selects among these configurations by running `make` with the name of a build target specified on the command-line, e.g.,

```
> make gfortran
```

to build the code using the GNU Fortran and C compilers. Some of the available targets are listed in the table below, and additional targets can be added by simply editing the `Makefile` in the top-level directory.

Target	Fortran compiler	C compiler	MPI wrappers
`xlf`	xlf90	xlc	mpxlf90 / mpcc
`pgi`	pgf90	pgcc	mpif90 / mpicc
`ifort`	ifort	gcc	mpif90 / mpicc
`gfortran`	gfortran	gcc	mpif90 / mpicc
`bluegene`	bgxlf95_r	bgxlc_r	mpxlf95_r / mpxlc_r

The MPAS framework supports multiple *cores* — currently a shallow water model, an ocean model, a non-hydrostatic atmosphere model, and a non-hydrostatic atmosphere initialization core — so the build process must be told which core to build. This is done by either setting the environment variable `CORE` to the name of the model core to build, or by specifying the core to be built explicitly on the command-line when running `make`. For the atmosphere core, for example, one may run either

```
> setenv CORE atmosphere
> make gfortran
```

or

```
> make gfortran CORE=atmosphere
```

If the `CORE` environment variable is set and a core is specified on the command-line, the command-line value takes precedence; if no core is specified, either on the command line or via the `CORE` environment variable, the build process will stop with an error message stating such. Assuming compilation is successful, the model executable, named `${CORE}_model` (e.g., `atmosphere_model`), should be created in the top-level MPAS directory.

In order to get a list of available cores, one can simply run the top-level `Makefile` without setting the `CORE` environment variable or passing the core via the command-line. An example of the output from this can be seen below.

```
> make
( make error )

Usage: make target CORE=[core] [options]

Example targets:
    ifort
```

```
    gfortran
    xlf
    pgi

Availabe Cores:
    atmosphere
    init_atmosphere
    landice
    ocean
    sw

Available Options:
    DEBUG=true    - builds debug version. Default is optimized version.
    USE_PAPI=true - builds version using PAPI for timers. Default is off.
    TAU=true      - builds version using TAU hooks for profiling. Default is off.
    AUTOCLEAN=true    - forces a clean of infrastructure prior to build new core.
    GEN_F90=true  - Generates intermediate .f90 files through CPP, and builds with them.

Ensure that NETCDF, PNETCDF, PIO, and PAPI (if USE_PAPI=true) are environment variables
that point to the absolute paths for the libraries.

************ ERROR ************
No CORE specified. Quitting.
************ ERROR ************
```

3.4 Selecting a single-precision build

Beginning with version 2.0, MPAS-Atmosphere can be compiled and run in single-precision, offering faster model execution and smaller input and output files. To compile a single-precision MPAS-Atmosphere executable, it is necessary to first add `-DSINGLE_PRECISION` to the definition of `MODEL_FORMULATION` at the top of the main `Makefile`. Additionally, any compiler flags, such as `-fdefault-real-8`, `-r8`, or `-real-size 64` must be removed from the compiler target that will be used. For example, if compiling with the gfortran compiler, one would change the line

```
"FFLAGS_OPT = -O3 -m64 -ffree-line-length-none -fdefault-real-8 -fdefault-double-8
                                                    -fconvert=big-endian -ffree-form" \
```

in the top-level `Makefile` to

```
"FFLAGS_OPT = -O3 -m64 -ffree-line-length-none -fconvert=big-endian -ffree-form" \
```

before compiling the *init_atmsophere_model* and *atmosphere_model* executables as usual.

Note that running MPAS-Atmosphere in single-precision requires the user to begin with single-precision SCVT grid files, and all pre-processing steps must be run using a single-precision version of init_atmosphere_model with these grid files. In order to obtain suitable grid files, any existing double-precision SCVT grid file that was downloaded from the MPAS-Atmosphere meshes download page may be run through the `double_to_float_grid` converter program to produce a single-precision grid file. Using the double-precision grid files with single-precision executables will not work.

3.5 Cleaning

To remove all files that were created when the model was built, including the model executable itself, `make` may be run for the 'clean' target:

```
> make clean
```

As with compiling, the core to be cleaned is specified by the `CORE` environment variable, or by specifying a core explicitly on the command-line with `CORE=`.

Chapter 4

Preparing Meshes

This chapter describes the steps used to prepare SCVT meshes for use in MPAS-A. For quasi-uniform meshes, very little preparation is actually needed, and generally, one only needs to prepare mesh decomposition files — files that describe the decomposition of the SCVT mesh across processors — when running MPAS-A using multiple MPI tasks. The procedure for creating these mesh decomposition files is described in the first section.

For variable-resolution SCVT meshes, the area of mesh refinement may be rotated to any part of the sphere using a program, grid_rotate, described in the second section. This utility program may be obtained from the MPAS-A download page.

4.1 Graph partitioning with METIS

Before MPAS can be run in parallel, a mesh decomposition file with an appropriate number of partitions (equal to the number of MPI tasks that will be used) is required. A limited number of mesh decomposition files, named `graph.info.part.*`, are provided with each mesh, as is the mesh connectivity file, named `graph.info`. If the number of MPI tasks to be used when running MPAS matches one of the pre-computed decomposition files, then there is no need to run METIS.

In order to create new mesh decomposition files for some particular number of MPI tasks, only the `graph.info` file is required. The currently supported method for partitioning a graph.info file uses the METIS software (http://glaros.dtc.umn.edu/gkhome/views/metis). The serial graph partitioning program, METIS (rather than ParMETIS or hMETIS) should be sufficient for quickly partitioning any mesh usable by MPAS.

After installing METIS, a `graph.info` file may be partitioned into N partitions by running

```
> gpmetis graph.info N
```

where N is the required number of partitions. The resulting file, `graph.info.part.`N, can then be copied into the MPAS run directory before running the model with N MPI tasks.

4.2 Relocating refinement regions on the sphere

The purpose of the grid_rotate program is simply to rotate an MPAS mesh file, moving a refinement region from one geographic location to another, so that the mesh can be re-used for different applications. This utility was developed out of the need to save computational resources, since

generating an SCVT — particularly one with a large number of generating points or a high degree of refinement — can take considerable time.

To build the grid_rotate program, the Makefile should first be edited to set the Fortran compiler to be used; if the NetCDF installation pointed to by the `NETCDF` environment variable was build with a separate Fortran interface library, it will also be necessary to add `-lnetcdff` just before `-lnetcdf` in the Makefile. After editing the Makefile, running 'make' should result in a grid_rotate executable file.

Besides the MPAS grid file to be rotated, grid_rotate requires a namelist file, `namelist.input`, which specifies the rotation to be applied to the mesh. The namelist variables are summarized in the table below

config_original_latitude_degrees	original latitude of any point on the sphere
config_original_longitude_degrees	original longitude of any point on the sphere
config_new_latitude_degrees	latitude to which the original point should be shifted
config_new_longitude_degrees	longitude to which the original point should be shifted
config_birdseye_rotation_counter_clockwise_degrees	rotation about a vector from the sphere center through the original point

Essentially, one chooses any point on the sphere, decides where that point should be shifted to, and specifies any change to the orientation (i.e., rotation) of the mesh about that point.

Having set the rotation parameters in the `namelist.input` file, the grid_rotate program should be run with two command-line options specifying the original grid file name and the name of the rotated grid file to be produced, e.g.,

```
> grid_rotate grid.nc grid_SE_Asia_refinement.nc
```

The original grid file will not be altered, and a new, rotated grid file will be created. The NCL script `mesh.ncl` may be used to plot either of the original or rotated grid files after suitable setting the name of the grid file in the script.

Note: The grid_rotate program initializes the new, rotated grid file to a copy of the original grid file. If the original grid file has only read permission (i.e., no write permission), then so will the copy, and consequently, the grid_rotate program will fail when attempting to update the fields in the copy.

Chapter 5

Configuring Model Input and Output

The reading and writing of model fields in MPAS is handled by user-configurable *streams*. A stream represents a fixed set of model fields, together with dimensions and attributes, that are all written or read together to or from the same file or set of files. Each MPAS model core may define its own set of default streams that it typically uses for reading initial conditions, for writing and reading restart fields, and for writing additional model history fields. Besides these default streams, users may define new streams to, e.g., write 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 model core. The name of this XML file is simply 'streams.' suffixed with the name of the core. For example, the streams for the *atmosphere* core are defined in a file named 'streams.atmosphere', and the streams for the *init_atmosphere* core are defined in a file named 'streams.init_atmosphere'. An XML stream file may further reference other text files that contain lists of the model fields that are read or written in each of the streams defined in the XML stream file.

Changes to the XML stream configuration file will take effect the next time an MPAS core is run; there is no need to re-compile after making modifications to the XML files. As described in the next section, it is therefore possible, e.g., to change the interval at which a stream is written, the template for the filenames associated with a stream, or the set of fields that are written to a stream, without the need to re-compile any code.

Two classes of streams exist in MPAS: *immutable* streams and *mutable* streams. Immutable streams are those for which the set of fields that belong to the stream may not be modified at model run-time; however, it is possible to modify the interval at which the stream is read or written, the filename template describing the files containing the stream on disk, and several other parameters of the stream. In contrast, all aspects of mutable streams, including the set of fields that belong to the stream, may be modified at run-time. The motivation for the creation of two stream classes is the idea that an MPAS core may not function correctly if certain fields are not read in upon model start-up or written to restart files, and it is therefore not reasonable for users to modify this set of required fields at run-time. An MPAS core developer may choose to implement such streams as immutable streams. Since fields may not be added to an immutable stream at run-time, new immutable streams may not be defined at run-time, and the only type of new stream that may be defined at run-time is the mutable stream type.

5.1 XML stream configuration files

The XML stream configuration file for an MPAS core always has a parent XML *element* named `streams`, within which individual streams are defined:

```
<streams>

    ... one or more stream definitions ...

</streams>
```

Immutable streams are defined with the `immutable_stream` element, and mutable streams are defined with the `stream` element:

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


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

    ... model fields belonging to this stream ...

</stream>
```

As shown in the example stream definitions, above, both classes of stream have the following required attributes:

- `name` — A unique name used to refer to the stream.
- `type` — The type of stream, either `"input"`, `"output"`, `"input;output"`, or `"none"`. A stream may be both an input and an output stream (i.e., `"input;output"`) if, for example, it is read once at model start-up to provide initial conditions and thereafter written periodically to provide model checkpoints. A stream may be defined as neither input nor output (i.e., `"none"`) for the purposes of defining a set of fields for inclusion other streams. Note that, for immutable streams, the type attribute may not be changed at run-time.
- `filename_template` — The template for files that exist or will be created by the stream. The filename template may include any of the following variables, which are expanded based on the simulated time at which files are first created.
  - `$Y` — Year
  - `$M` — Month

  - `$D` — Day of the month
  - `$d` — Day of the year
  - `$h` — Hour
  - `$m` — Minute
  - `$s` — Second

  A filename template may include either a relative or an absolute path, in which case MPAS will attempt to create any directories in the path that do not exist, subject to filesystem permissions.

- `input_interval` — For streams that have type `"input"` or `"input;output"`, the interval, beginning at the model initial time, at which the stream will be read. Possible values include a time interval specification in the format `"YYYY-MM-DD_hh:mm:ss"`; the value `"initial_only"`, which specifies that the stream is read only once at the model initial time; or the value `"none"`, which specifies that the stream is not read during a model run.

- `output_interval` — For streams that have type `"output"` or `"input;output"`, the interval, beginning at the model initial time, at which the stream will be written. Possible values include a time interval specification in the format `"YYYY-MM-DD_hh:mm:ss"`; the value `"initial_only"`, which specifies that the stream is written only once at the model initial time; or the value `"none"`, which specifies that the stream is not written during a model run.

Finally, the set of fields that belong to a mutable stream may be specified with any combination of the following elements. Note that, for immutable streams, no fields are specified at run-time in the XML configuration file.

- `var` — Associates the specified variable with the stream. The variable may be any of those defined in an MPAS core's Registry.xml file, but may not include individual constituent arrays from a var_array.

- `var_array` — Associates all constituent variables in a var_array, defined in an MPAS core's Registry.xml file, with the stream.

- `var_struct` — Associates all variables in a var_struct, defined in an MPAS core's Registry.xml file, with the stream.

- `stream` — Associates all explicitly associated fields in the specified stream with the stream; streams are not recursively included.

- `file` — Associates all variables listed in the specified text file, with one field per line, with the stream.

5.2 Optional stream attributes

Besides the required attributes described in the preceding section, several additional, optional attributes may be added to the definition of a stream.

- `filename_interval` — The interval between the timestamps used in the construction of the names of files associated with a stream. Possible values include a time interval specification in the format `"YYYY-MM-DD_hh:mm:ss"`; the value `"none"`, indicating that only one file containing all times is associated with the stream; the value `"input_interval"` that, for input type streams, indicates that each time to be read from the stream will come from a unique file; or the value `"output_interval"` that, for output type streams, indicates that each time to be written to the stream will go to a unique file whose name is based on the timestamp of the data being written. The default value is `"input_interval"` for input type streams and `"output_interval"` for output type streams. For streams of type `"input;output"`, the default filename interval is `"input_interval"` if the input interval is an interval (i.e., not `"initial_only"`), or `"output_interval"` otherwise. Refer to Section 5.3.1 for an example of the use of the filename_interval attribute.

- `reference_time` — A time that is an integral number of filename intervals from the timestamp of any file associated with the stream. The default value is the start time of the model simulation. Refer to Section 5.3.3 for an example of the use of the reference_time attribute.

- `clobber_mode` — Specifies how a stream should handle attempts to write to a file that already exists. Possible values for the mode include:

  - `"overwrite"` — The stream is allowed to overwrite records in existing files and to append new records to existing files; records not explicitly written to are left untouched.
  - `"truncate"` or `"replace_files"` — The stream is allowed to overwrite existing files, which are first truncated to remove any existing records; this is equivalent to replacing any existing files with newly created files of the same name.
  - `"append"` — The stream is only allowed to append new records to existing files; existing records may not be overwritten.
  - `"never_modify"` — The stream is not allowed to modify existing files in any way.

  The default clobber mode for streams is `"never_modify"`. Refer to Section 5.3.2 for an example of the use of the clobber_mode attribute.

- `precision` — The precision with which real-valued fields will be written or read in a stream. Possible values include `"single"` for 4-byte real values, `"double"` for 8-byte real values, or `"native"`, which specifies that real-valued fields will be written or read in whatever precision the MPAS core was compiled. The default value is `"native"`. Refer to Section 5.3.1 for an example of the use of the precision attribute.

- `packages` — A list of packages attached to the stream. A stream will be active (i.e., read or written) only if at least one of the packages attached to it is active, or if no packages at all are attached. Package names are provided as a semi-colon-separated list. Note that packages may only be defined in an MPAS core's Registry.xml file at build time. By default, no packages are attached to a stream.

- `io_type` — The underlying library and file format that will be used to read or write a stream. Possible values include:
  - `"pnetcdf"` — Read/write the stream with classic large-file NetCDF files (CDF-2) using the ANL Parallel-NetCDF library.
  - `"pnetcdf,cdf5"` — Read/write the stream with large-variable files (CDF-5) using the ANL Parallel-NetCDF library.
  - `"netcdf"` — Read/write the stream with classic large-file NetCDF files (CDF-2) using the Unidata serial NetCDF library.
  - `"netcdf4"` — Read/write the stream with HDF-5 files using the Unidata parallel NetCDF-4 library.

  Note that the PIO library must have been built with support for the selected `io_type`. By default, all input and output streams are read and written using the `"pnetcdf"` option.

5.3 Stream definition examples

This section provides several example streams that make use of the optional stream attributes described in Section 5.2. All examples are of output streams, since it is more likely that a user will need to write additional fields than to read additional fields, which a model would need to be aware of; however, the concepts that are illustrated here translate directly to input streams as well.

5.3.1 Example: a single-precision output stream with one month of data per file

In this example, the optional attribute specification `filename_interval="01-00_00:00:00"` is added to force a new output file to be created for the stream every month. Note that the general format for time interval specifications is `YYYY-MM-DD_hh:mm:ss`, where any leading terms can be omitted; in this case, the year part of the interval is omitted. To reduce the file size, the specification `precision="single"` is also added to force real-valued fields to be written as 4-byte floating-point values, rather than the default of 8 bytes.

```
<stream name="diagnostics"
        type="output"
        filename_template="diagnostics.$Y-$M.nc"
        filename_interval="01-00_00:00:00"
        precision="single"
        output_interval="6:00:00" >

     <var name="u10"/>
     <var name="v10"/>
     <var name="t2"/>
     <var name="q2"/>

</stream>
```

The only fields that will be written to this stream are the hypothetical 10-m diagnosed wind components, the 2-m temperature, and the 2-m specific humidity variables. Also, note that the filename template only includes the year and month from the model valid time; this can be problematic when the simulation starts in the middle of a month, and a solution for this problem is illustrated in the example of Section 5.3.3.

5.3.2 Example: appending records to existing output files

By default, streams will never modify existing files whose filenames match the name of a file that would otherwise be written during the course of a simulation. However, when restarting a simulation that is expected to add more records to existing output files, it can be useful to instruct the MPAS I/O system to append these records, thereby modifying existing files. This may be accomplished with the `clobber_mode` attribute.

```
<stream name="diagnostics"
        type="output"
        filename_template="diagnostics.$Y-$M.nc"
        filename_interval="01-00_00:00:00"
        precision="single"
        clobber_mode="append"
        output_interval="6:00:00" >

     <var name="u10"/>
     <var name="v10"/>
     <var name="t2"/>
     <var name="q2"/>

</stream>
```

In general, if MPAS were to attempt to write a record at a time that already existed in an output file, a clobber_mode of 'append' would not permit the write to take place, since this would modify existing data; in 'append' mode, only new records may be added. However, due to a peculiarity in the implementation of the 'append' clobber mode, it may be possible for an output file to contain duplicate times. This can happen when the first record that is appended to an existing file has a timestamp not matching any in the file, after which, any record that is written — regardless of whether its timestamp matches one already in the file — will be appended to the end of the file. This situation may arise, for example, when restarting a model simulation with a shorter `output_interval` than was used in the original model simulation with an MPAS core that does not write the first output time for restart runs.

5.3.3 Example: referencing filename intervals to a time other than the start time

The example stream of the previous sections creates a new file each month during the simulation, and the filenames contain only the year and month of the timestamp when the file was created. If a simulation begins at 00 UTC on the first day of a month, then each file in the diagnostic stream will contain only output times that fall within the month in the filename. However, if a

simulation were to begin in the middle of a month — for example, the month of June, 2014 — the first diagnostics output file would have a filename of 'diagnostics.2014-06.nc', but rather than containing only output fields valid in June, it would contain all fields written between the middle of June and the middle of July, at which point one month of simulation would have elapsed, and a new output file, 'diagnostics.2014-07.nc', would be created.

In order to ensure that the file 'diagnostics.2014-06.nc' contained only data from June 2014, the `reference_time` attribute may be added such that the day, hour, minute, and second in the date and time represent the first day of the month at 00 UTC. In this example, the year and month of the reference time are not important, since the purpose of the reference time here is to describe to MPAS that the monthly filename interval begins (i.e., is referenced to) the first day of the month.

```
<stream name="diagnostics"
        type="output"
        filename_template="diagnostics.$Y-$M.nc"
        filename_interval="01-00_00:00:00"
        reference_time="2014-01-01_00:00:00"
        precision="single"
        clobber_mode="append"
        output_interval="6:00:00" >

     <var name="u10"/>
     <var name="v10"/>
     <var name="t2"/>
     <var name="q2"/>

</stream>
```

In general, the components of a timestamp, `YYYY-MM-DD_hh:mm:ss`, that are less significant than (i.e., to the right of) those contained in a filename template are important in a reference_time. For example, with a `filename_template` that contained only the year, the month component of the `reference_time` would become important to identify the month of the year on which the yearly basis for filenames would begin.

Chapter 6

Physics Suites

Beginning with version 4.0, MPAS-Atmosphere introduces a new way of selecting the physics schemes to be used in a simulation. Rather than selecting individual parameterization schemes for different processes (e.g., convection, microphysics, etc.), the preferred method is for the user to select a *suite* of parameterization schemes that have been tested together. The selection of a physics suite is made via the new namelist option `config_physics_suite` in the `&physics` namelist record. Each of the available suites are described in the sections that follow.

Although the preferred method for selecting the schemes in a simulation is via the choice of a suite, the need to enable or disable individual schemes, or to substitute alternative schemes for the suite default, is recognized. Accordingly, it is possible to override the choice of any individual parameterization scheme through the namelist options described in Appendix B.7. This is useful, e.g., to disable all parameterization except for microphysics when running some idealized simulations, or to turn off the cumulus scheme when running at cloud-resolving resolutions.

6.1 Suite: mesoscale_reference

The default physics suite in MPAS-Atmosphere is the 'mesoscale_reference' suite, which contains the schemes listed in Table 6.1. This suite has been tested for mesoscale resolutions ($>$ 10 km cell spacing), and is not appropriate for convective-scale simulations because the Tiedtke scheme will remove convective instability before resolved-scale motions (convective cells) can respond to it.

Table 6.1: The set of parameterization schemes used by the 'mesoscale_reference' physics suite.

Parameterization	Scheme
Convection	Tiedtke
Microphysics	WSM6
Land surface	Noah
Boundary layer	YSU
Surface layer	Monin-Obukhov
Radiation, LW	RRTMG
Radiation, SW	RRTMG
Gravity wave drag	none

6.2 Suite: none

As of Version 4.0, the only other recognized physics suite in MPAS-Atmosphere is the 'none' suite, which sets all physics parameterizations to 'off'. This suite is primarily intended for use with idealized simulations. For example, the idealized supercell test case makes use of the 'none' suite, but with the microphysics scheme explicitly overridden:

```
config_physics_suite = 'none'
config_microp_scheme = 'kessler'
```

Chapter 7

Running the MPAS Non-hydrostatic Atmosphere Model

Given an SCVT mesh, this chapter describes the two main steps to running the MPAS-Atmosphere model: creating initial conditions and running the model itself. This chapter makes use of two MPAS cores, `init_atmosphere` and `atmosphere`, which are, respectively, used for initializing and running the non-hydrostatic atmospheric model. Sections 7.1 and 7.2 of this chapter describe the creation of idealized and real-data non-hydrostatic initial condition files using the `init_atmosphere` core. Section 7.3 describes the basic procedure of running the model itself.

Each section of this chapter follows a familiar pattern of compiling and executing MPAS model components, albeit using different cores depending on its intended use. The compilation will create either an initialization or a model executable, which are named, respectively, `init_atmosphere_model` and `atmosphere_model`. In general, an executable is run with `mpiexec` or `mpirun`, for example:

```
> mpiexec -n 8 atmosphere_model
```

where `8` is the number of MPI tasks to be used. In any case where `n` > 1, there must exist a corresponding graph decomposition file, e.g., `graph.info.part.8`. For more on graph decomposition, see Section 4.1.

7.1 Creating idealized ICs

There are several idealized test cases supported within the `init_atmosphere` model initialization core:

1 — Jablonowski and Williamson baroclinic wave, no initial perturbation [1]
2 — Jablonowski and Williamson baroclinic wave, with initial perturbation
3 — Jablonowski and Williamson baroclinic wave, with normal-mode perturbation
4 — squall line
5 — super-cell
6 — mountain wave

[1] Jablonowski, C. and D.L. Williamson, 2006, A baroclinic instability test case for atmospheric model dynamical cores, *QJRMS*, 132, 2943-2975. doi:10.1256/qj.06.12.

Creating idealized initial conditions is fairly straightforward, as no external data are required and the starting date/time is irrelevant to building the `init.nc` file that will be used to run the model.

The following steps summarize the creation of `init.nc`:

- Include a `grid.nc` file, which contains the SCVT mesh, in the working directory
- If running with more than one MPI task, include a `graph.info.part.*` file in the working directory (Section 4.1)
- Compile MPAS with the `init_atmosphere` core specified (Section 3.3)
- Edit the `namelist.init_atmosphere` configuration file (described below)
- Edit the `streams.init_atmosphere` I/O configuration file (described below)
- Run `init_atmosphere_model` to create the initial condition file, `init.nc`

When the `init_atmosphere_model` executable is built, a default namelist, `namelist.init_atmosphere`, will have been created. A number of the namelist parameters found in `namelist.init_atmosphere` are irrelevant to creating idealized conditions and can be removed or ignored. The following table outlines the namelist parameters that are required; comments are given on the right for certain key parameters, and formal explanations for all namelist parameters can be found in Appendix A.

&nhyd_model		
config_init_case = 2		a number between 1 and 6 corresponding to the cases listed at the beginning of this section
config_start_time = '0000-01-01_00:00:00'		the starting time for the simulation
config_theta_adv_order = 3		advection order for theta
/		
&dimensions		
config_nvertlevels = 26		the number of vertical levels to be used in the model
/		
&decomposition		
config_block_decomp_file_prefix	=	if running in parallel, needs to match the grid decomposition file prefix
'graph.info.part.'		
/		

After editing the `namelist.init_atmosphere` namelist file, the name of the input SCVT grid file, as well as the name of the initial condition file to be created, must be set in the XML I/O configuration file, `streams.init_atmosphere`. For a detailed description of the format of the XML I/O configuration file, refer to Chapter 5. Specifically, the `filename_template` attribute must be set to the name of the SCVT grid file in the `"input"` stream definition, and the `filename_template` attribute must be set to name of the initial condition file to be created in the `"output"` stream definition.

7.2 Creating real-data ICs

Creating real-data initial conditions is similar to that of the idealized case described in the previous section, but is more involved as it requires interpolation of static geographic data (e.g., topography, land cover, soil category, etc.), surface fields such as soil temperature and SST, and the atmospheric initial conditions valid at a specific date and time. The static datasets are the same as those used by the WRF model, and the surface fields and atmospheric initial conditions can be obtained from, e.g., NCEP's GFS data using the WRF pre-processing system (WPS).

Creating real-data initial conditions requires a single compilation of the `init_atmosphere` core, but the actual generation of the IC files will take place using three separate runs of `init_atmosphere`, where each of these runs is described individually in the following sub-sections. While it is possible to condense the three real-data initialization steps into fewer executions, running each step separately will both improve clarity and, as will become apparent, save a significant amount of time when generating subsequent initial conditions, that is, when making initial conditions using the same mesh but different starting times.

The first of the three steps, described in Section 7.2.1, is the interpolation of static fields onto the mesh to create a `static.nc` file. This step cannot be run in parallel and takes considerably longer than the steps that follow, however, the fields being static, this step need only be run once for a particular mesh, regardless of the number of initial condition files that are ultimately created from the `static.nc` output file. Described in Section 7.2.2 is an optional step that creates a file `surface.nc` containing surface data at regular intervals. This file is used in the case where the model run makes periodic external updates to surface fields (currently only sea-ice and SST). Finally, Section 7.2.3 describes the processing of the atmospheric initial conditions beginning with the `static.nc` file created in Section 7.2.1. Naturally, each of the initialization runs described in the three following sections will make use of a `namelist.init_atmosphere` namelist file, and as was the case for idealized initial conditions, the default `namelist.init_atmosphere` file in the MPAS directory may be used as a starting point. Not every variable in this namelist is needed for any particular step, and therefore each section will elaborate only on the namelist variables that are immediately relevant.

7.2.1 Static fields

The generation of a file `static.nc` requires a set of static geographic data. A suitable dataset can be obtained from the WRF model's download page
`http://www.mmm.ucar.edu/wrf/users/download/get_source.html`. These static data files should be downloaded to a directory, which will be specified the `namelist.init_atmosphere` file (described below) prior to running this interpolation step. The result of this run will be the creation of a NetCDF file (`static.nc`), which is used in the two steps, described in Sections and , following this to create surface update files and dynamic initial conditions. Note that `static.nc` can be generated once and then used repeatedly to generate surface update and initial condition files for different start times.

The following steps summarize the creation of `static.nc`:

- Download geographic data from the WRF download page (described above)
- Compile MPAS with the `init_atmosphere` core specified (Section 3.3)
- Include a `grid.nc` file in the working directory
- Edit the `namelist.init_atmosphere` configuration file (described below)

- Edit the `streams.init_atmosphere` I/O configuration file (described below)
- Run `init_atmosphere_model` *with only one MPI task specified* to create `static.nc`

Note that it is critical for this step that the initialization core is run serially; afterward, however, the steps described in 7.2.2 and 7.2.3 may be run with more than one MPI task.

&nhyd_model config_init_case = 7 /	must be 7, the real-data initialization case
&dimensions config_nvertlevels = 1 config_nsoillevels = 1 /	the following two variables must be present now, though their values do not become significant until §3.2.3
&data_sources config_geog_data_path = '/WPS_GEOG/' /	absolute path to static files obtained from the WRF download page
&preproc_stages config_static_interp = .true. config_vertical_grid = .false. config_met_interp = .false. config_input_sst = .false. /	only the static_interp stage should be enabled

After editing the `namelist.init_atmosphere` namelist file, the name of the input SCVT grid file, as well as the name of the static file to be created, must be set in the XML I/O configuration file, `streams.init_atmosphere`. For a detailed description of the format of the XML I/O configuration file, refer to Chapter 5. Specifically, the `filename_template` attribute must be set to the name of the SCVT grid file in the `"input"` stream definition, and the `filename_template` attribute must be set to name of the static file to be created in the `"output"` stream definition.

7.2.2 Surface field updates

This step is optional — it is required only if surface fields are to be periodically updated during the model run. The surface data could originate from any number of sources, though the most straightforward way to obtain a dataset in the appropriate format is to process GRIB data (e.g., GFS GRIB data) with the *ungrib* program of the WRF model's pre-processing system (WPS). Detailed instructions for building and running the WPS, and the process of generating intermediate data files from GFS data, can be found in Chapter 3 of the WRF User Guide: `http://www.mmm.ucar.edu/wrf/users/docs/user_guide/users_guide_chap3.htm`.

The following steps summarize the creation of `surface.nc`:

- Include surface data intermediate files in the working directory

- Include a `static.nc` file in the working directory (Section 7.2.1)
- If running in parallel, include a `graph.info.part.*` in the working directory (Section 4.1)
- Edit the `namelist.init_atmosphere` configuration file (see below)
- Edit the `streams.init_atmosphere` I/O configuration file (described below)
- Run `init_atmosphere_model` to create `surface.nc`

&nhyd_model		
config_init_case = 8		must be 8, the surface field initialization case
config_start_time = '2010-10-23_00:00:00'		time to begin processing surface data
config_stop_time = '2010-10-30_00:00:00'		time to end processing surface data
/		
&data_sources		
config_sfc_prefix = 'SST'		the prefix of the intermediate data files containing SST and sea-ice
config_fg_interval = 86400		interval between intermediate files to use for SST and sea-ice
/		
&preproc_stages		only the config_input_sst step should be enabled
config_static_interp = .false.		
config_vertical_grid = .false.		
config_met_interp = .false.		
config_input_sst = .true.		
config_frac_seaice = .true.		
/		
&decomposition		
config_block_decomp_file_prefix	=	if running in parallel, needs to match the grid decomposition file prefix
'graph.info.part.'		
/		

After editing the `namelist.init_atmosphere` namelist file, the name of the static file, as well as the name of the surface update file to be created, must be set in the XML I/O configuration file, `streams.init_atmosphere`. Specifically, the `filename_template` attribute must be set to the name of the static file in the `"input"` stream definition, and the `filename_template` attribute must be set to name of the surface update file to be created in the `"surface"` stream definition. *Also, for the "surface" stream, ensure that the "output_interval" attribute is set to the interval at which the surface intermediate files are provided.*

7.2.3 Vertical grid generation and initial field interpolation

The final step for creating a real-data initial conditions file (`init.nc`) is to generate a vertical grid, the parameters of which will be specified in the `namelist.init_atmosphere` file, and to obtain

an initial conditions dataset and interpolate it onto the model grid. As stated previously, while initial conditions could ultimately be obtained from many different data sources, here we assume the use of intermediate data files obtained from GFS data using the WPS ungrib program. Detailed instructions for building and running the WPS, and how to generate intermediate data files from GFS data, can be found in Chapter 3 of the WRF user guide:
`http://www.mmm.ucar.edu/wrf/users/docs/user_guide/users_guide_chap3.htm`.

The following steps summarize the creation of `init.nc`:

- Include a WPS intermediate data file in the working directory
- Include the `static.nc` file in the working directory (Section 7.2.1)
- If running in parallel, include a `graph.info.part.*` file in the working directory (Section 4.1)
- Edit the `namelist.init_atmosphere` configuration file (described below)
- Edit the `streams.init_atmosphere` I/O configuration file (described below)
- Run `init_atmosphere_model` to create `init.nc`

&nhyd_model	
config_init_case = 7	must be 7
config_start_time = '2010-10-23_00:00:00'	time to process first-guess data
config_theta_adv_order = 3	advection order for theta
/	
&dimensions	
config_nvertlevels = 41	number of vertical levels to be used in MPAS
config_nsoillevels = 4	number of soil layers to be used in MPAS
config_nfglevels = 38	number of vertical levels in intermediate file
config_nfgsoillevels = 4	number of soil layers in intermediate file
/	
&data_sources	
config_met_prefix = 'FILE'	the prefix of the intermediate file to be used for initial conditions
/	
&vertical_grid	
config_ztop = 30000.0	model top height (m)
config_nsmterrain = 1	number of smoothing passes for terrain
config_smooth_surfaces = .true.	whether to smooth zeta surfaces
/	
&preproc_stages	
config_static_interp = .false.	
config_vertical_grid = .true.	only these three stages should be enabled
config_met_interp = .true.	

config_input_sst = .false. /		
&decomposition config_block_decomp_file_prefix 'graph.info.part.' /	=	if running in parallel, needs to match the grid decomposition file prefix

After editing the `namelist.init_atmosphere` namelist file, the name of the static file, as well as the name of the initial condition file to be created, must be set in the XML I/O configuration file, `streams.init_atmosphere`. Specifically, the `filename_template` attribute must be set to the name of the static file in the `"input"` stream definition, and the `filename_template` attribute must be set to name of the initial condition file to be created in the `"output"` stream definition.

7.3 Running the model

With the files `init.nc` and, optionally, `surface.nc`, generated as in the previous sections, we have completed the prerequisites to run the model. The only step remaining before running the model itself is the configuration of `namelist.atmosphere`. When the *atmosphere* core is built, a default `namelist.atmosphere` namelist file will be automatically generated; this namelist can serve as a starting point for any modifications made following the steps below. This section will discuss both running the model from a cold start and restarting the model from some point in a previous run.

The following steps summarize running the model:

- Include an initial condition NetCDF file (e.g., `init.nc`) in the working directory(Section 7.1, Section 7.2)
- If using surface updates, include a surface NetCDF file (e.g., `surface.nc`) in the working directory (Section 7.2.2)
- If running in parallel, include a graph decomposition file in the working directory (Section 4.1)
- If the MPAS directory has not been cleaned since running initialization, run `make clean` with the `atmosphere` core specified
- Compile MPAS with the `atmosphere` core specified (Section 3.3)
- Edit the default `namelist.atmosphere` configuration file (described below)
- Edit the `streams.atmosphere` I/O configuration file (described below)
- Run the `atmosphere_model` executable

Below is a list of variables in `namelist.atmosphere` that pertain to model timestepping, explicit horizontal diffusion, and model restarts. A number of namelist variables are not listed here (specifications for dynamical core configuration, physics parameters, etc.) and Appendix B should be consulted for the purpose and acceptable values of these parameters.

&nhyd_model		

config_dt = 450.0	the model timestep; an appropriate value must be chosen relative to the grid cell spacing
config_start_time = '2010-10-23_00:00:00'	the model start time corresponding to `init.nc`
config_run_duration = '5_00:00:00'	the duration of the model run; for format rules, see Appendix B
config_len_disp = 120000.0	the smallest cell-to-cell distance in the mesh, used for computing a dissipation length scale
/	
&decomposition	
config_block_decomp_file_prefix = 'graph.info.part.'	if running in parallel, must match the prefix of the graph decomposition file
/	
&restart	
config_do_restart = .false.	if true, will select the appropriate `restart.nc` file generated from a previous run
/	

When running the model from a cold start, `config_start_time` should match the time that was used when creating `init.nc`.

Configuration of model input and output is accomplished by editing the `streams.atmosphere` file. The following streams exist by default in the atmosphere core:

input	the stream used to read model initial conditions for cold-start simulations
restart	the stream used to periodically write restart files during model integration, and to read initial conditions when performing a restart model run
output	the stream responsible for writing model prognostic and diagnostic fields to history files
diagnostics	the stream responsible for writing (mostly) 2-d diagnostic fields, typically at higher temporal frequency than the history files
surface	the stream used to read periodic updates of sea-ice and SST from a surface update file created as described in Section 7.2.2

For more information on the options available in the XML I/O configuration file, users are referred to Chapter 5.

During the course of a model run, restart files are created at an interval specified by the `output_interval` attribute in the definition of the `"restart"` stream. Running the model from a restart file is similar to running the model from `init.nc`. The required changes are that `config_do_restart` must be set to `.true.` and `config_start_time` must correspond to a restart file existing in the working directory.

Chapter 8

Visualization

Since the MPAS input and output files are in NetCDF format, a wide variety of software tools may be used to manipulate and visualize fields in these files. As a starting point, several NCL[1] scripts for making the basic types of plots illustrated in Figure 8.1 are provided through the MPAS-Atmosphere download page. Each of these scripts reads the name of the file from which fields should be plotted from the environment variable `FNAME`, and all but the mesh-plotting script read the time frame to be plotted from the environment variable `T`. To plot a field from the first frame (indexed from 0) of the file `output.2010-10-23_00:00:00.nc`, for example, one would set the following environment variables

```
> setenv FNAME output.2010-10-23_00:00:00.nc
> setenv T 0
```

before running one of the scripts. In general, the specific field to be plotted from the NetCDF file must be set within a script before running that script.

8.1 Meshes

A plot showing just an MPAS SCVT mesh can be produced using the `atm_mesh.ncl` script, as in Figure 8.1(a). This script reads a subset of the mesh description fields in Appendix C and uses this information to draw the SCVT mesh over a color-filled map background. Parameters in the script can be used to control the type of map projection (e.g., orthographic, cylindrical equidistant, etc.), the colors used to fill land and water points, and the widths of lines used for the Voronoi cells.

8.2 Horizontal contour plots

Contour plots of horizontal fields can be produced with the `atm_contours.ncl` script, as in Figure 8.1(b). The particular field to be plotted is set in the script and can in principle be drawn on any horizontal surface (e.g., a constant pressure surface, a constant height surface, sea-level, etc.) if suitable vertical interpolation code is added to the script. Not shown in the figure are horizontal wind vectors, which can also be added to the plot using example code provided in the script.

[1] NCAR Command Language; `http://ncl.ucar.edu`

8.3 Horizontal cell-filled plots

For visualizing horizontal fields on their native SCVT grid, the `atm_cells.ncl` script may be used to produce horizontal cell-filled plots, as in Figure 8.1(c). This script draws each MPAS grid cell as a polygon colored according to the value of the field in that cell; the color scale is automatically chosen based on the range of the field and the default NCL color table, though other color tables can be selected instead.

8.4 Vertical cross-sections

Vertical cross-sections of fields can be created using the `atm_xsec.ncl` script, as in Figure 8.1(d). Before running this script, a starting point and an ending point for the cross section must be given as latitude-longitude pairs near the top of the script, and the number of points along the cross section should be specified. The script evenly distributes the specified number of points along the shortest great-circle arc from the starting point to the ending point, and for each point, the script uses values from the grid cell containing that point (i.e., a nearest-neighbor interpolation to the horizontal cross-section points is performed); no vertical interpolation is performed, and the thicknesses and vertical heights of cells are all drawn according to the MPAS vertical grid.

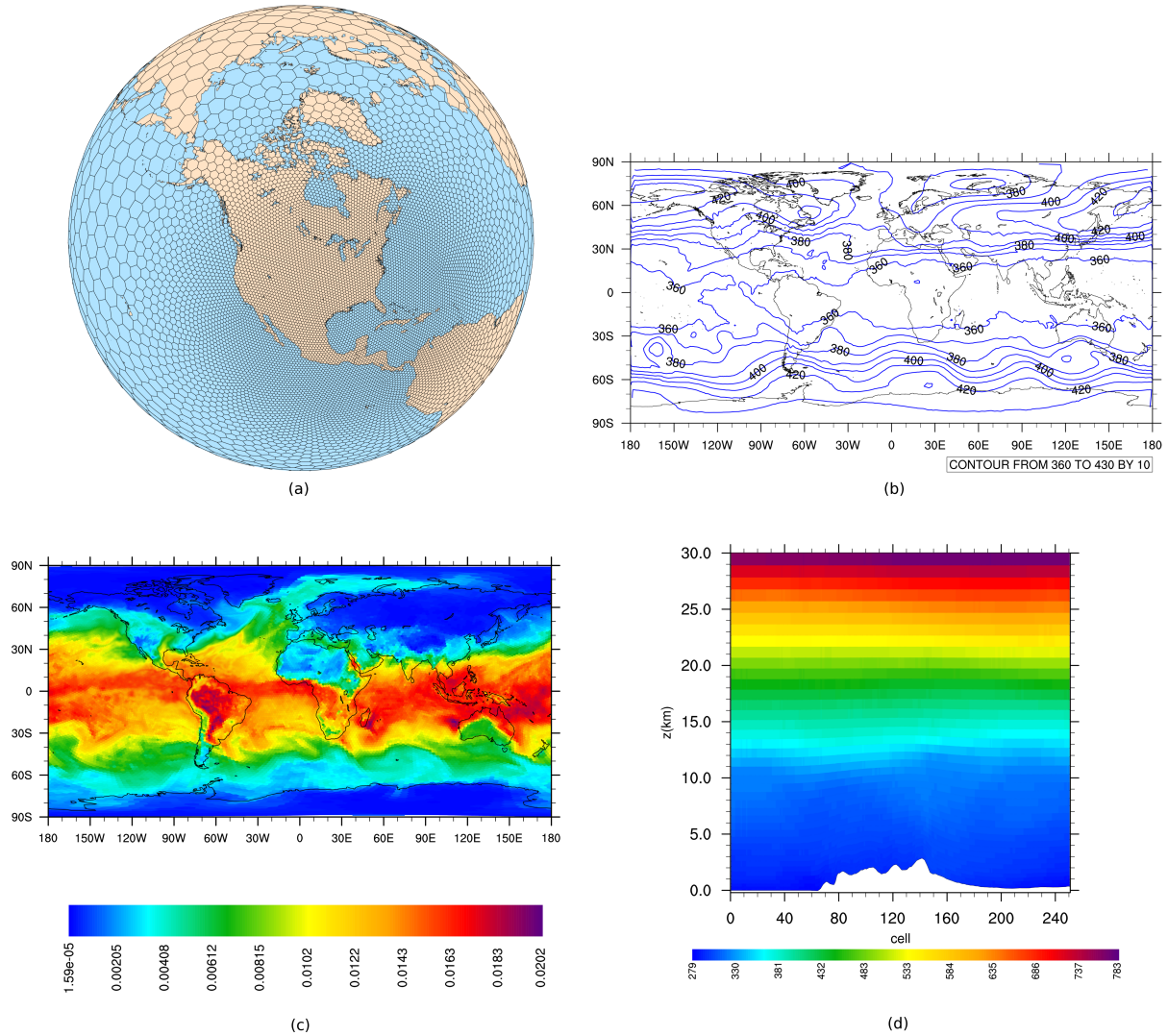


Figure 8.1: Various plot types that can be produced from MPAS input or output files using example scripts provided with the MPAS code. (a) A plot of an MPAS SCVT mesh against a filled map background. (b) A simple horizontal contour plot. (c) A cell-filled horizontal plot, with individual cells of the mesh drawn as polygons colored according to the field value. (d) A vertical cross-section in height, with areas below the terrain left unshaded.

Appendix A

Initialization Namelist Options

This chapter summarizes the complete set of namelist options available when running the MPAS non-hydrostatic atmosphere initialization core. The applicability of certain options depends on the type of initial conditions to be created — idealized or 'real-data' — and such applicability is identified in the description when it exists.

Date-time strings throughout all MPAS namelists assume a common format. Specifically, time intervals are of the form `'[DDD_]HH:MM:SS[.sss]'`, where `DDD` is an integer number of days with any number of digits, `HH` is a two-digit hour value, `MM` is a two-digit minute value, `SS` is a two-digit second value, and `sss` are fractions of a second with any number of digits; any part of the time interval format in square bracktes (`[ ]`) may be omitted, and if days are omitted, `HH` may be either a one- or two-digit hour specification. Time instants (e.g., start time or end time) are of the form `'YYYY-MM-DD[_HH:MM:SS[.sss]]'`, where `YYYY` is an integer year with any number of digits, `MM` is a two-digit month value, `DD` is a two-digit day value, and `HH:MM:SS.sss` is a time with the same format as in a time interval specification. For both time instants and time intervals, a value of `'none'` represents 'no value'.

A.1 nhyd_model

config_init_case	Type of initial conditions to create: 1 = Jablonowski & Williamson barolinic wave (no initial perturbation), 2 = Jablonowski & Williamson barolinic wave (with initial perturbation), 3 = Jablonowski & Williamson barolinic wave (with normal-mode perturbation), 4 = squall line, 5 = super-cell, 6 = mountain wave, 7 = real-data initial conditions from, e.g., GFS, 8 = surface field (SST, sea-ice) update file for use with real-data simulations, *Default value: 7*
config_theta_adv_order	Advection order for theta *Default value: 3*
config_coef_3rd_order	Upwinding coefficient in the 3rd order advection scheme. $0 \leq$ config_coef_3rd_order ≤ 1 *Default value: 0.25*
config_start_time	Time to begin processing first-guess data (case 7 only) *Default value: 'none'*

config_stop_time	Time to end processing first-guess data (case 7 only) *Default value: 'none'*

A.2 dimensions

config_nvertlevels	The number of vertical levels to be used in the model *Default value: 26*
config_nsoillevels	The number of vertical soil levels needed by LSM in the model (case 7 only) *Default value: 4*
config_nfglevels	The number of vertical levels in first-guess data (case 7 only) *Default value: 27*
config_nfgsoillevels	The number of vertical soil levels in first-guess data (case 7 only) *Default value: 4*

A.3 data_sources

config_geog_data_path	Path to the WPS static data files (case 7 only) *Default value: '/mmm/users/wrfhelp/WPS_GEOG/'*
config_met_prefix	Prefix of ungrib intermediate file to use for initial conditons (case 7 only) *Default value: 'FILE'*
config_sfc_prefix	Prefix of ungrib intermediate file to use for SST and sea-ice (cases 7 and 8 only) *Default value: 'FILE'*
config_fg_interval	Interval between intermediate files to use for SST and sea-ice (case 8 only) *Default value: 21600*
config_landuse_data	The land use classification to use, either 'USGS' or 'MODIFIED_IGBP_MODIS_NOAH' (case 7 only) *Default value: 'USGS'*

A.4 vertical_grid

config_ztop	Model top height, in meters *Default value: 28000.0*
config_nsmterrain	Number of smoothing passes to apply to interpolated terrain field *Default value: 2*
config_smooth_surfaces	Whether to smooth zeta surfaces *Default value: .false.*
config_dzmin	Minimum thickness of layers as a fraction of nominal thickness *Default value: 0.3*
config_nsm	Maximum number of smoothing passes for coordinate surfaces *Default value: 30*

A.5 preproc_stages

config_static_interp	Whether to interpolate WPS static data (case 7 only) *Default value: .true.*
config_vertical_grid	Whether to generate vertical grid *Default value: .true.*

config_met_interp	Whether to interpolate first-guess fields from intermediate file *Default value: .true.*
config_input_sst	Whether to re-compute SST and sea-ice fields from surface input data set; should be set to .true. if Case 8 was run *Default value: .false.*
config_frac_seaice	Whether to switch sea-ice threshold from 0.5 to 0.02 *Default value: .false.*

A.6 io

config_pio_num_iotasks	Number of I/O tasks to use 0 implies all MPI tasks are I/O tasks *Default value: 0*
config_pio_stride	The separation (stride) in MPI rank between I/O tasks *Default value: 1*

A.7 decomposition

config_block_decomp_file_prefix	Prefix for mesh decomposition file *Default value: 'graph.info.part.'*

Appendix B

Model Namelist Options

This chapter summarizes the complete set of namelist options available when running the MPAS non-hydrostatic atmosphere model. All date-time string specifications are of the form described at the beginning of Appendix A.

B.1 nhyd_model

config_dt	Model time step, in seconds *Default value: 600.0*
config_start_time	Starting time for model run *Default value: '0000-01-01_00:00:00'*
config_run_duration	Length of model run *Default value: 'none'*
config_stop_time	Stopping time for model run *Default value: 'none'*
config_split_dynamics_transport	Whether to split integration of dynamics equations from scalar transport *Default value: .true.*
config_number_of_sub_steps	Number of acoustic steps per large RK step *Default value: 2*
config_dynamics_split_steps	Number of full RK steps per timestep *Default value: 3*
config_h_mom_eddy_visc2	∇^2 eddy viscosity for horizontal diffusion of momentum *Default value: 0.0*
config_h_mom_eddy_visc4	∇^4 eddy hyper-viscosity for horizontal diffusion of momentum *Default value: 0.0*
config_v_mom_eddy_visc2	∇^2 eddy viscosity for vertical diffusion of momentum *Default value: 0.0*
config_h_theta_eddy_visc2	∇^2 eddy viscosity for horizontal diffusion of theta *Default value: 0.0*
config_h_theta_eddy_visc4	∇^4 eddy hyper-viscosity for horizontal diffusion of theta *Default value: 0.0*
config_v_theta_eddy_visc2	∇^2 eddy viscosity for vertical diffusion of theta *Default value: 0.0*
config_horiz_mixing	Formulation of horizontal mixing: '2d_smagorinsky' = 2-d Smagorinsky formulation, '2d_fixed' = fixed eddy viscosity, *Default value: '2d_smagorinsky'*

config_len_disp	Horizontal length scale for Smagorinsky formulation of horizontal diffusion *Default value: 120000.0*
config_theta_adv_order	Horizontal advection order for theta *Default value: 3*
config_scalar_adv_order	Horizontal advection order for scalars *Default value: 3*
config_w_adv_order	Horizontal advection order for w *Default value: 3*
config_u_vadv_order	Vertical advection order for normal velocities (u) *Default value: 3*
config_w_vadv_order	Vertical advection order for w *Default value: 3*
config_theta_vadv_order	Vertical advection order for theta *Default value: 3*
config_scalar_vadv_order	Vertical advection order for scalars *Default value: 3*
config_coef_3rd_order	Upwinding coefficient in the 3rd order advection scheme. $0 \leq$ config_coef_3rd_order ≤ 1 *Default value: 0.25*
config_scalar_advection	Whether to advect scalar fields *Default value: .true.*
config_positive_definite	Whether to enable positive-definite advection of scalars *Default value: .false.*
config_monotonic	Whether to enable monotonic limiter in scalar advection *Default value: .true.*
config_mix_full	mix full θ and u fields, or mix perturbation from initial state *Default value: .true.*
config_epssm	Off-centering parameter for the vertically implicit acoustic integration, dimensionless *Default value: 0.1*
config_smdiv	3D divergence damping coefficient, dimensionless. *Default value: 0.1*
config_h_ScaleWithMesh	Scale eddy viscosities with mesh-density function for horizontal diffusion *Default value: .false.*
config_apvm_upwinding	Amount of upwinding in APVM *Default value: 0.5*

B.2 damping

config_zd	Height MSL to begin w-damping profile *Default value: 22000.0*
config_xnutr	Maximum w-damping coefficient at model top *Default value: 0.0*

B.3 io

config_pio_num_iotasks	Number of I/O tasks to use 0 implies all MPI tasks are I/O tasks *Default value: 0*

config_pio_stride	The separation (stride) in MPI rank between I/O tasks *Default value: 1*

B.4 decomposition

config_block_decomp_file_prefix	Prefix of graph decomposition file, to be suffixed with the MPI task count *Default value: 'graph.info.part.'*

B.5 restart

config_do_restart	Whether this run of the model is a restart run *Default value: .false.*
config_do_DAcycling	Whether to re-compute coupled fields θ_m, $\tilde{\rho}$, ρu, etc. from uncoupled fields when restarting the model; used for cycling DA experiments that analyze uncoupled fields in restart files *Default value: .false.*

B.6 printout

config_print_global_minmax_vel	Whether to print the global min/max of the horizontal normal velocity field at the end of each timestep *Default value: .true.*
config_print_global_minmax_sca	Whether to print the global min/max of scalar fields at the end of each timestep *Default value: .false.*

B.7 physics

config_sst_update	Logical used to update the Sea-Surface Temperatures (SSTs), and fractional sea-ice (if available). If set to true, SSTs are updated using the file config_sfc_update_name. If set to false, SSTs remain fixed during the entire model run. *Default value: .false.*
config_sstdiurn_update	If set to true, a diurnal cycle is applied to the SSTs. If set to false, SSTs remain constant during the entire day. *Default value: .false.*
config_deepsoiltemp_update	If set to true, deep soil temperatures are slowly updated during the model run. If set to false, deep soil temperatures remain fixed during the entire run. *Default value: .false.*
config_radtlw_interval	Temporal interval between calls to the parameterizations of long wave radiation, format *'yyyy-mm-dd_hh:mm:ss'*. *Default value: '00:30:00'*
config_radtsw_interval	Temporal interval between calls to the parameterizations of short wave radiation, format *'yyyy-mm-dd_hh:mm:ss'*. *Default value: '00:30:00'*

config_bucket_update	Temporal interval between updates to restoring the accumulated rain and radiation fields below their respective bucket values, format *'yyyy-mm-dd_hh:mm:ss'*. *Default value: 'none'*
config_physics_suite	Physics suite: 'mesoscale_reference' = a suite of physics tested for mesoscale resolutions, 'none' = no physics parameterizations, *Default value: 'mesoscale_reference'*
config_microp_scheme	Cloud Microphysics schemes: 'off' = no microphysics, 'suite' = scheme determined by the choice of config_physics_suite, 'kessler' = Kessler, 'wsm6 ' = WSM6 *Default value: 'suite'*
config_convection_scheme	Convection schemes: 'off' = no convection scheme, 'suite' = scheme determined by the choice of config_physics_suite, 'kain_fritsch' = Kain-Fritsch, 'tiedtke' = Tiedtke *Default value: 'suite'*
config_lsm_scheme	Land-surface schemes: 'off' = no land surface option, 'suite' = scheme determined by the choice of config_physics_suite, 'noah' = NOAH land-surface scheme *Default value: 'suite'*
config_pbl_scheme	Planetary Boundary Layer schemes: 'off' = no boundary layer scheme, 'suite' = scheme determined by the choice of config_physics_suite, 'ysu' = YSU PBL scheme *Default value: 'suite'*
config_radt_cld_scheme	Parameterization of the cloud fraction for the long wave and short wave radiation schemes: 'off' = if LW and SW radiation schemes are both 'off', 'suite' = scheme determined by the choice of config_physics_suite, 'cld_incidence' = the cloud fraction is equal to 0 or 1, 'cld_fraction' = the cloud fraction varies between 0 and 1, as a function of the relative humidity *Default value: 'suite'*
config_radt_lw_scheme	Long wave (LW) radiation schemes: 'off' = no long-wave radiation scheme, 'suite' = scheme determined by the choice of config_physics_suite, 'cam_lw' = CAM LW radiation scheme, 'rrtmg_lw' = RRTMG LW radiation scheme *Default value: 'suite'*
config_radt_sw_scheme	Short wave (SW) radiation scheme: 'off' = no short-wave radiation scheme, 'suite' = scheme determined by the choice of config_physics_suite, 'cam_sw' = CAM SW radiation scheme, 'rrtmg_sw' = RRTMG SW radiation scheme *Default value: 'suite'*

config_sfclayer_scheme	Surface-layer schemes 'off' = no surface-layer scheme, 'suite' = scheme determined by the choice of config_physics_suite, 'monin_obukhov' = Monin-Obukhov scheme *Default value: 'suite'*
config_bucket_radt	Threshold value below which accumulated long wave and short wave radiation fields are restored to if config_bucket_update is different from *'none'.* Default value: '1.0e9'
config_bucket_rainc	Threshold value below which the accumulated convective precipitation is restored to if config_bucket_update is different from *'none'.* Default value: '100.0'
config_bucket_rainnc	Threshold value below which the accumulated grid-scale precipitation is restored to if config_bucket_update is different from *'none'.* Default value: '100.0'

Appendix C

Grid Description

This chapter provides a brief introduction to the common types of grids used in the MPAS framework.

C.1 Horizontal grid

The MPAS grid system requires the definition of seven elements. These seven elements are composed of two types of *cells*, two types of *lines*, and three types of *points*. These elements are depicted in Figure C.1 and defined in Table C.1. These elements can be defined on either the plane or the surface of the sphere. The two types of cells form two meshes, a primal mesh composed of Voronoi regions and a dual mesh composed of Delaunay triangles. Each corner of a primal mesh cell is uniquely associated with the "center" of a dual mesh cell and vice versa. So we define the two mesh as either a primal mesh (composed of cells P_i) or a dual mesh (composed of cells D_v). The center of any primal mesh cell, P_i, is denoted by $\mathbf{x}_i$ and the center of any the dual mesh cell, D_v, is denoted by $\mathbf{x}_v$. The boundary of a given primal mesh cell P_i is composed of the set of lines that connect the $\mathbf{x}_v$ locations of associated dual mesh cells D_v. Similarly, the boundary of a given dual mesh cell D_v is composed of the set of lines that connect the $\mathbf{x}_i$ locations of the associated primal mesh cells P_i. As shown in Figure C.1, a line segment that connects two primal mesh cell centers is uniquely associated with a line segment that connects two dual mesh cell centers. We assume that these two line segments cross and the point of intersection is labeled as $\mathbf{x}_e$. In addition, we assume that these two line segments are orthogonal as indicated in Figure C.1. Each $\mathbf{x}_e$ is associated with two distances: d_e measures the distance between the primal mesh cells sharing $\mathbf{x}_e$ and l_e measures the distance between the dual mesh cells sharing $\mathbf{x}_e$.

Since the two line segments crossing at $\mathbf{x}_e$ are orthogonal, these line segments form a convenient local coordinate system for each edge. At each $\mathbf{x}_e$ location a unit vector $\mathbf{n}_e$ is defined to be parallel to the line connecting primal mesh cells. A second unit vector $\mathbf{t}_e$ is defined such that $\mathbf{t}_e = \mathbf{k} \times \mathbf{n}_e$.

Table C.2 provides the names of all *elements* and all *sets of elements* as used in the MPAS framework. Elements appear twice in the table when described in the grid file in more than one way, e.g. points are described with both cartesian and latitude/longitude coordinates. An "ncdump -h" of any MPAS grid, output or restart file will contain all variable names shown in second column of Table C.2.

In addition to these seven element types, we require the definition of *sets of elements*. In all, eight different types of sets are required and these are defined and explained in Table C.3 and Figure C.2. The notation is always of the form of, for example, $i \in CE(e)$, where the LHS indicates the type of element to be gathered (cells) based on the RHS relation to another type of element (edges).

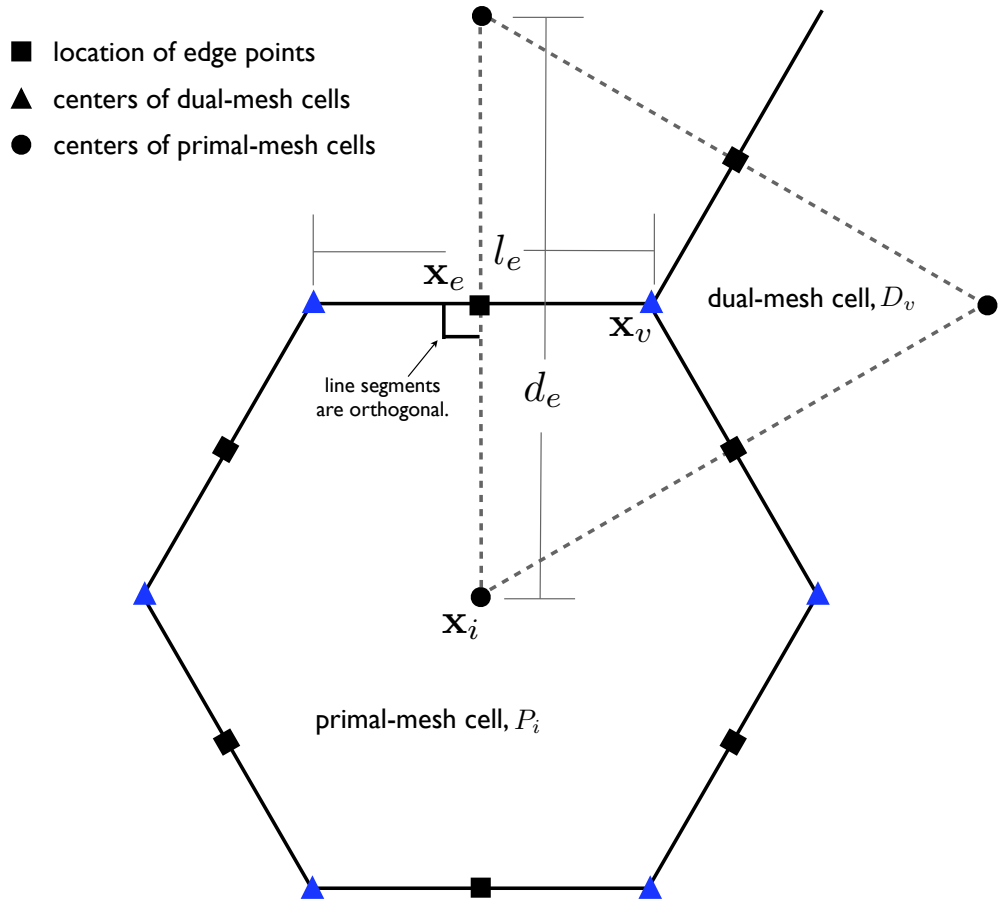


Figure C.1: Definition of elements used to build the MPAS grid. Also see Table C.1.

Table C.1: Definition of elements used to build the MPAS grid.

Element	*Type*	*Definition*
$\mathbf{x}_i$	point	location of center of primal-mesh cells
$\mathbf{x}_v$	point	location of center of dual-mesh cells
$\mathbf{x}_e$	point	location of edge points where velocity is defined
d_e	line segment	distance between neighboring $\mathbf{x}_i$ locations
l_e	line segment	distance between neighboring $\mathbf{x}_v$ locations
P_i	cell	a cell on the primal-mesh
D_v	cell	a cell on the dual-mesh

The angle of each edge in an MPAS grid is provided in the variable *angleEdge*. The angle given is the angle between a vector pointing north and a vector pointing in the positive tangential direction of the edge. Referring to Fig. C.3,

$$\text{angleEdge} = \arcsin \|\hat{\mathbf{n}} \times \hat{\mathbf{v}}\|,$$

where $\hat{\mathbf{n}}$ is the unit vector pointing north and $\hat{\mathbf{v}}$ is the unit vector pointing from verticesOnEdge(1,iEdge) to verticesOnEdge(2,iEdge).

Given a wind vector $(u_\perp, u_\parallel)$ defined in term of components orthogonal to and parallel to the edge, the earth-relative wind (u, v) may be recovered as

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} u_\perp \\ u_\parallel \end{bmatrix},$$

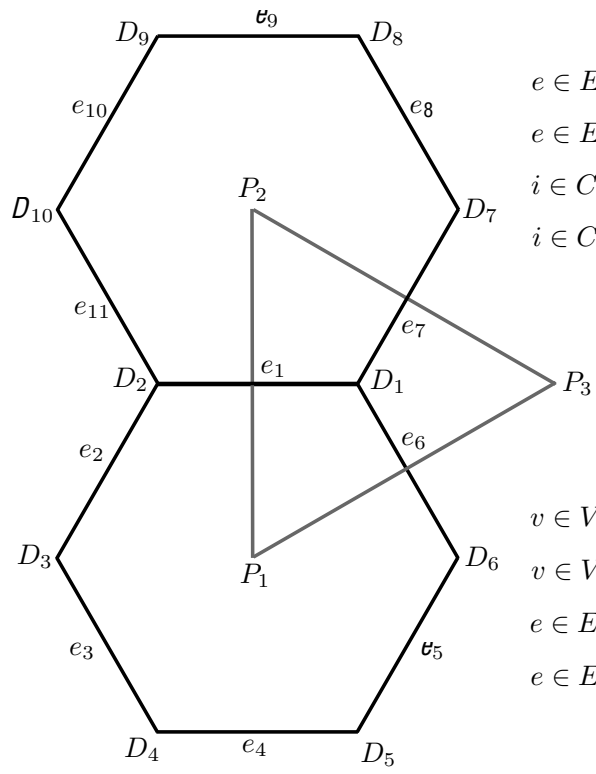
where $\alpha = \text{angleEdge}$.

Table C.2: Variable names used to describe a MPAS grid.

Element	*Name*	*Size*	*Comment*
$\mathbf{x}_i$	{x,y,z}Cell	nCells	cartesian location of $\mathbf{x}_i$
$\mathbf{x}_i$	{lon,lat}Cell	nCells	longitude and latitude of $\mathbf{x}_i$
$\mathbf{x}_v$	{x,y,z}Vertex	nVertices	cartesian location of $\mathbf{x}_v$
$\mathbf{x}_v$	{lon,lat}Vertex	nVertices	longitude and latitude of $\mathbf{x}_v$
$\mathbf{x}_e$	{x,y,z}Edge	nEdges	cartesian location of $\mathbf{x}_e$
$\mathbf{x}_e$	{lon,lat}Edge	nEdges	longitude and latitude of $\mathbf{x}_e$
d_e	dcEdge	nEdges	distance between $\mathbf{x}_i$ locations
l_e	dvEdge	nEdges	distance between $\mathbf{x}_v$ locations
$e \in EC(i)$	edgesOnCell	(nEdgesMax,nCells)	edges that define P_i.
$e \in EV(v)$	edgesOnVertex	(3,nCells)	edges that define D_v.
$i \in CE(e)$	cellsOnEdge	(2,nEdges)	primal-mesh cells that share edge e.
$i \in CV(v)$	cellsOnVertex	(3,nVertices)	primal-mesh cells that define D_v.
$v \in VE(e)$	verticesOnEdge	(2,nEdges)	dual-mesh cells that share edge e.
$v \in VI(i)$	verticesOnCell	(nEdgesMax,nCells)	vertices that define P_i.

Table C.3: Definition of element groups used to reference connections in the MPAS grid. Examples are provided in Figure C.2.

Syntax	*output*
$e \in EC(i)$	set of edges that define the boundary of P_i.
$e \in EV(v)$	set of edges that define the boundary of D_v.
$i \in CE(e)$	two primal-mesh cells that share edge e.
$i \in CV(v)$	set of primal-mesh cells that form the vertices of dual mesh cell D_v.
$v \in VE(e)$	the two dual-mesh cells that share edge e.
$v \in VI(i)$	the set of dual-mesh cells that form the vertices of primal-mesh cell P_i.
$e \in ECP(e)$	edges of cell pair meeting at edge e.
$e \in EVC(v, i)$	edge pair associated with vertex v and mesh cell i.



$e \in EC(P_1) = [e_1, e_2, e_3, e_4, e_5, e_6]$

$e \in EV(D_1) = [e_1, e_6, e_7]$

$i \in CE(e_1) = [P_1, P_2]$

$i \in CV(D_1) = [P_1, P_2, P_3]$

$v \in VE(e_1) = [D_1, D_2]$

$v \in VC(P_1) = [D_1, D_2, D_3, D_4, D_4, D_5, D_6]$

$e \in ECP(e_1) = [e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}]$

$e \in ECV(P_1, D_1) = [e_1, e_6]$

Figure C.2: Definition of element groups used to reference connections in the MPAS grid. Also see Table C.3.

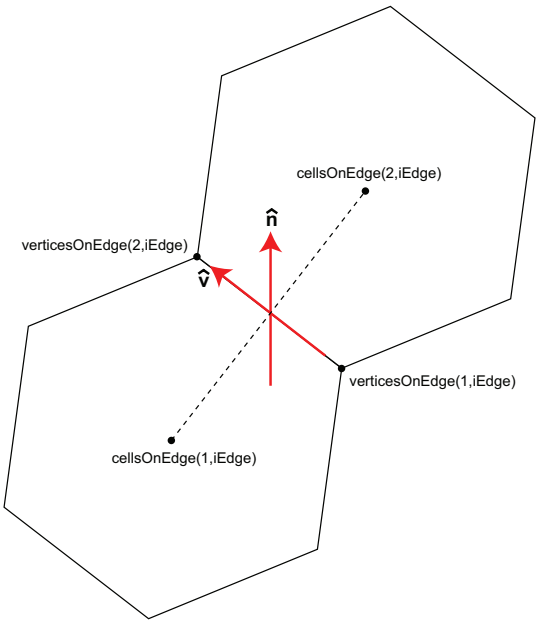


Figure C.3: The angle of an edge refers to the angle between a vector pointing north at an edge location and a vector pointing in the positive tangential velocity direction of the edge.

C.2 Vertical grid

The vertical coordinate in MPAS-Atmosphere is ζ and has units of length, where $0 \leq \zeta \leq z_t$ and z_t is the height of the model top. The relationship between the vertical coordinate and height in the physical domain is given as

$$z = \zeta + A\, h_s(x, y, \zeta) \tag{C.1}$$

where (x, y) denotes a location on the horizontal mesh and ζ is the vertical coordinate (ζ is directed radially outward from the surface of the sphere, or perpendicular to the horizontal (x, y) plane in a Cartesian coordinate MPAS-A configuration). MPAS-A can be configured with the traditional Gal-Chen and Somerville terrain-following coordinate by setting $h_s(x, y, \zeta) = h(x, y)$ and $A = 1 - \zeta/z_t$, where $h(x, y)$ is the terrain height. Alternatively, A can be modified to allow a more rapid or less rapid transition to the constant-height upper boundary condition. Additionally, a constant-height coordinate can be specified at some intermediate height below h_t.

The influence of the terrain on any coordinate surface ζ can be influenced by the specification of $h_s(x, y, \zeta)$. Specifically, h_s can be set such that $h_s(x, y, 0) = h(x, y)$ (i.e. terrain following at the surface), and progressively filtered fields of $h(x, y)$ can be used at $\zeta > 0$ in $h_s(x, y, \zeta)$, such that the small-scale features in the topography are quickly filtered from the coordinate. Example MPAS-A vertical meshes are given in Figure C.4

On the MPAS-A mesh C-grid staggering, the state variables u, ρ, θ and scalars are located halfway between w levels in both physical height and in the coordinate ζ. Variables associated with the coordinate systems used in the MPAS-A solver, and possibly appearing in its input, output or history files, are defined in Table C.4 and depicted in Figure C.5.

Further information about the vertical coordinate can be found in
Klemp, J. B. (2011). A Terrain-Following Coordinate with Smoothed Coordinate Surfaces. *Mon. Wea. Rev.*, **139**, 2163-2169. doi:10.1175/MWR-D-10-05046.1

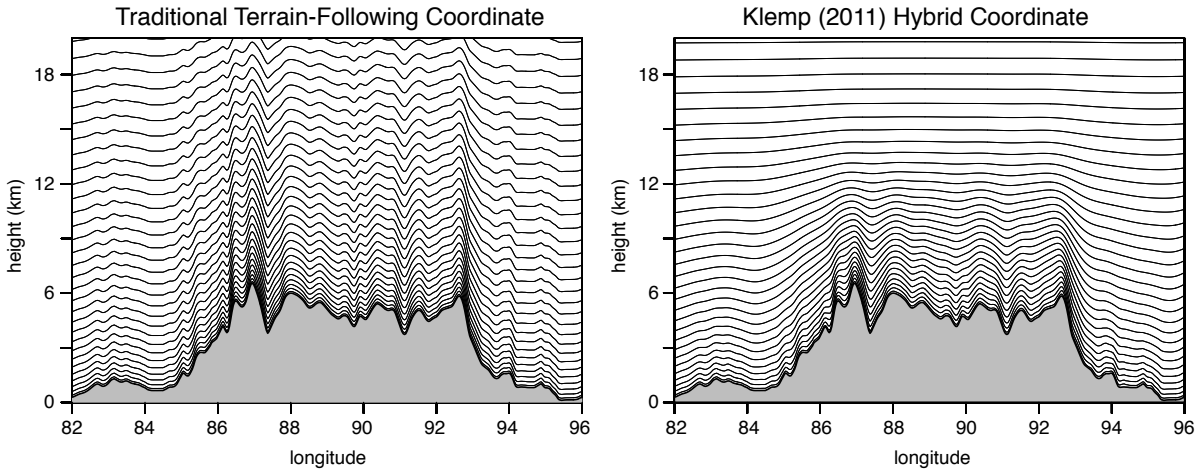


Figure C.4: Example MPAS-A vertical meshes using terrain following (left) and smoothed (right) vertical coordinates.

Table C.4: Vertical coordinate variables in MPAS-Atmosphere. *level* is the integer model level (usually specified with index k where $k = 1$ is the lowest model level and physical height increases with increasing k). Δ denotes a vertical difference between levels, and *cell* is a given mesh cell on the primary mesh.

Variable	*Definition*
zgrid($level, cell$)	physical height of the w points in meters.
zw($level$)	ζ at w levels.
zu($level$)	ζ at u levels; $zu(k) = [zw(k+1) + zw(k)]/2$.
dzw($level$)	$\Delta\zeta$ at u levels; $dzw(k) = zw(k+1) - zw(k)$.
dzu($level$)	$\Delta\zeta$ at w levels; $dzu(k) = [dzw(k+1) + dzw(k)]/2$.
rdzw($level$)	$1/dzw$.
rdzu($level$)	$1/dzu$.
zz($level, cell$)	$\Delta\zeta/\Delta z$ at u levels; $(zw(k+1) - zw(k))/(zgrid(k+1, cell) - zgrid(k, cell))$.
fzm($level$)	weight for linear interpolation to $w(k)$ point for $u(k)$ level variable.
fzp($level$)	weight for linear interpolation to $w(k)$ point for $u(k-1)$ level variable.

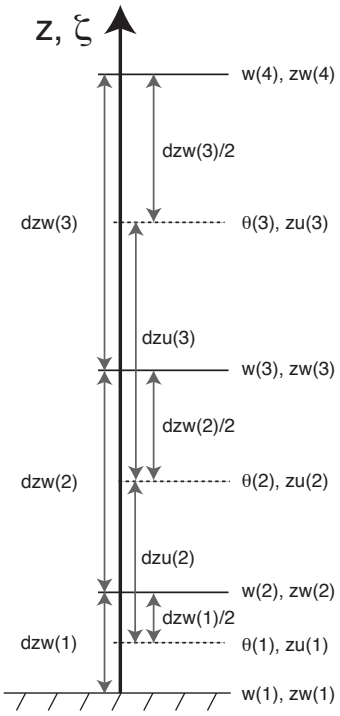


Figure C.5: Vertical distribution of the variables in MPAS-A. Also see Table C.4.

Appendix D

Description of Model Fields

In this chapter, the fields required in an MPAS input file are described. The dimensionality of each field is given in the name of each field, with the dimensions in C storage order (i.e., the fastest-varying dimension is outer-most).

D.1 Field dimensions

nSoilLevels	Number of soil layers
nMonths	Constant value 12
StrLen	Length of strings

D.2 Vertical grid fields

double hx(nCells, nVertLevelsP1)	h_s field used in smoothed terrain-following coordinate
double zgrid(nCells, nVertLevelsP1)	height (m) at vertical cell interfaces (i.e. at w points)
double rdzw(nVertLevels)	$1/(\Delta\zeta$ between w-levels)
double dzu(nVertLevels)	$\Delta\zeta$ between u-levels (layer centers)
double rdzu(nVertLevels)	1/dzu
double fzm(nVertLevels)	dzw_{k-1} / dzu_k, level $k-1$ interp weight from u to w points
double fzp(nVertLevels)	dzw_k / dzu_k, level k interp weight from u to w points
double zx(nEdges, nVertLevelsP1)	$\partial z/\partial x$
double zz(nCells, nVertLevelsP1)	$\partial\zeta/\partial z$
double zb(nEdges, TWO, nVertLevelsP1)	coefficients for contribution from U to w diagnosis
double zb3(nEdges, TWO, nVertLevelsP1)	coefficients for 3rd-order contribution from U to Ω diagnosis

D.3 Initial fields

char xtime(Time, StrLen)	Time stamp for each time record of fields in the file
double theta(Time, nCells, nVertLevels)	Potential temperature (K)
double rho(Time, nCells, nVertLevels)	Dry density (kg m^{-3})
double u(Time, nEdges, nVertLevels)	Normal wind velocity at edges (m s^{-1})
double w(Time, nCells, nVertLevelsP1)	Vertical velocity at vertical cell faces (m s^{-1})
double qv(Time, nCells, nVertLevels)	Water vapor mixing ratio (kg kg^{-1})
double qc(Time, nCells, nVertLevels)	Cloud water mixing ratio (kg kg^{-1})
double qr(Time, nCells, nVertLevels)	Rain water mixing ratio (kg kg^{-1})
double ter(nCells)	Terrain height (m)
int landmask(nCells)	Land-water mask (1=land, 0=water)
int ivgtyp(nCells)	Vegetation type category
int isltyp(nCells)	Soil type category
double snoalb(nCells)	Annual maximum snow albedo
double soiltemp(nCells)	Climatological average soil temperature
double greenfrac(nCells, nMonths)	Climatological monthly mean greeness fraction
double shdmin(nCells)	Annual minimum greeness fraction
double shdmax(nCells)	Annual maximum greeness fraction
double var2d (nCells)	Orographic variance
double con (nCells)	Orographic convexity
double oa1 (nCells)	Orographic direction asymmetry function
double oa2 (nCells)	Orographic direction asymmetry function
double oa3 (nCells)	Orographic direction asymmetry function
double oa4 (nCells)	Orographic direction asymmetry function
double ol1 (nCells)	Orographic direction asymmetry function
double ol2 (nCells)	Orographic direction asymmetry function
double ol3 (nCells)	Orographic direction asymmetry function
double ol4 (nCells)	Orographic direction asymmetry function
double albedo12m(nCells, nMonths)	Climatological monthly mean background albedo
double dss(nCells, nVertLevels)	w-damping coefficient
double dzs(Time, nCells, nSoilLevels)	Thickness of soil layers (m)
double zs(Time, nCells, nSoilLevels)	Depth of center of soil layers (m)
double sh2o(Time, nCells, nSoilLevels)	Soil liquid water (m^3 m^{-3})
double smois(Time, nCells, nSoilLevels)	Soil moisture (m^3 m^{-3})
double tslb(Time, nCells, nSoilLevels)	Soil layer temperature (K)
double tmn(Time, nCells)	Deep soil temperature (K)
double skintemp(Time, nCells)	Surface skin temperature (K)
double sst(Time, nCells)	Sea-surface temperature (K)
double snow(Time, nCells)	Snow water equivalent (kg m-2)
double snowc(Time, nCells)	Flag indicating snow coverage (1 for snow; 0 otherwise)
double snowh(Time, nCells)	Physical snow depth (m)
double xice(Time, nCells)	Sea-ice fraction
double seaice(Time, nCells)	Sea-ice mask (1 when xice > 0, 0 otherwise)
double vegfra(Time, nCells)	Vegetation fraction
double sfc_albbck(Time, nCells)	Surface background albedo
double xland(Time, nCells)	Land-ocean mask (1=land, 2=ocean)
double qv_init(nVertLevels)	qv reference profile

double t_init(nCells, nVertLevels)	theta reference profile for each cell
double u_init(nVertLevels)	u reference profile
double coeffs_reconstruct (nCells, maxEdges, R3)	Coefficients for reconstructing wind vectors at cell centers.
double defc_b(nCells, maxEdges)	coefficients for computing the diagonal components of the horizontal deformation
double defc_a(nCells, maxEdges)	coefficients for computing the off-diagonal components of the horizontal deformation
int advCells(nCells, TWENTYONE)	indices of cells used to compute cell-centered 2nd derivatives for transport scheme
double deriv_two(nEdges, TWO, FIFTEEN)	weights for cell centered 2nd derivatives, normal to edge, for the transport scheme
double surface_pressure(Time, nCells)	Surface pressure (Pa)
double theta_base(Time, nCells, nVertLevels)	Reference-state potential temperature (K)
double rho_base(Time, nCells, nVertLevels)	Reference-state dry density (kg m^{-3})
double pressure_base (Time, nCells, nVertLevels)	Reference-state dry pressure (Pa)
double exner_base (Time, nCells, nVertLevels)	Reference-state Exner function (-)
double fEdge(nEdges)	Coriolis parameter at edge locations
double fVertex(nVertices)	Coriolis parameter at vertex locations
double cf1	surface interp weight for level $k = 1$ values
double cf2	surface interp weight for level $k = 2$ values
double cf3	surface interp weight for level $k = 3$ values

D.4 History fields

double v(Time, nEdges, nVertLevels)	Tangential wind velocity at edges (m s^{-1})
double uReconstructMeridional (Time, nCells, nVertLevels)	Reconstructed meridional velocity at cell centers (m s^{-1})
double uReconstructZonal (Time, nCells, nVertLevels)	Reconstructed zonal velocity at cell centers (m s^{-1})
double uReconstructZ (Time, nCells, nVertLevels)	Reconstructed z-component of velocity at cell centers (m s^{-1})
double uReconstructY (Time, nCells, nVertLevels)	Reconstructed y-component of velocity at cell centers (m s^{-1})
double uReconstructX (Time, nCells, nVertLevels)	Reconstructed x-component of velocity at cell centers (m s^{-1})
double rho_zz(Time, nCells, nVertLevels)	Dry density divided by d(zeta)/dz
double theta_m(Time, nCells, nVertLevels)	Modified moist potential temperature (K)

D.4.1 Deep soil temperature update

integer nsteps_accum (Time,nCells)	Number of accumulated time-step in a day	(-)

integer ndays_accum (Time,nCells)	Number of accumulated days in a year	(-)
double tlag (Time,nCells)	Daily mean surface temperature for prior days	(K)
double tday_accum (Time,nCells)	Accumulated daily surface temperature for current day	(K)
double tyear_mean (Time,nCells)	Annual mean surface temperature	(K)
double tyear_accum (Time,nCells)	Accumulated yearly surface temperature for current year	(K)

D.4.2 Cloud microphysics schemes

integer i_rainnc (Time nCells)	Counter for bucket updates	(-)
double refl10cm_max (Time,nCells)	Maximum 10 cm radar reflectivity	(dBz)
double precipw (Time,nCells)	Precipitable water	(kg m^{-2})
double rainncv (Time,nCells)	Time-step total precipitation due to microphysics	(mm)
double snowncv (Time,nCells)	Time-step precipitation of snow due to microphysics	(mm)
double graupelncv (Time,nCells)	Time-step precipitation of graupel due to microphysics	(mm)
double rainnc (Time,nCells)	Accumulated total precipitation due to microphysics	(mm)
double snownc (Time,nCells)	Accumulated precipitation of snow due to microphysics	(mm)
double graupelnc (Time,nCells)	Accumulated precipitation of graupel due to microphysics	(mm)
double sr (Time,nCells)	Ratio frozen to total precipitation due to microphysics	(-)

D.4.3 Convection schemes

integer i_rainc (Time nCells)	Counter for bucket updates	(-)
double cuprec (Time,nCells)	Time-step total precipitation rate due to convection	(mm s^{-1})
double raincv (Time,nCells)	Time-step total precipitation due to convection	(mm)
double rain (Time,nCells)	Accumulated total precipitation due to convection	(mm)
double rqccuten (Time,nCells,nVertLevels)	Tendency of cloud water due to convection	(kg kg^{-1} s^{-1})
double rqicuten (Time,nCells,nVertLevels)	Tendency of cloud ice due to convection	(kg kg^{-1} s^{-1})
double rqvcuten (Time,nCells,nVertLevels)	Tendency of water vapor due to convection	(kg kg s^{-1})
double rthcuten (Time,nCells,nVertLevels)	Tendency of potential temperature due to convection	(K s^{-1})
KAIN-FRITSCH SCHEME SPECIFICS:		
double nca (Time,nCells)	Lifetime of convective clouds	(s)
double cubot (Time,nCells)	Index level of cloud base for convective clouds	(-)
double cutop (Time,nCells)	Index level of cloud top for convective clouds	(-)
double wavg0 (Time,nCells,nVertLevels)	Temporal running-average vertical velocity	(m s^{-1})
double rqrcuten (Time,nCells,nVertLevels)	Tendency of rain due to convection	(kg kg^{-1} s^{-1})
double rqscuten (Time,nCells,nVertLevels)	Tendency of snow due to convection	(kg kg^{-1} s^{-1})
TIEDTKE SCHEME SPECIFICS:		

double rucuten (Time,nCells,nVertLevels)	Tendency of zonal wind due to convection	(m s^{-1} s^{-1})
double rvcuten (Time,nCells,nVertLevels)	Tendency of meridional wind due to convection	(m s^{-1} s^{-1})
double rqvdynten (Time,nCells,nVertLevels)	Tendency of water vapor due to total dynamical advection	(kg kg s^{-1})

D.4.4 Planetary boundary (PBL) schemes

integer kpbl (Time,nCells)	Index level of PBL top	(-)
double hpbl (Time,nCells)	Height of PBL top	(m)
double rublten (Time,nCells,nVertLevels)	Tendency of zonal wind due to PBL processes	(m s^{-1} s^{-1})
double rvblten (Time,nCells,nVertLevels)	Tendency of meridional wind due to PBL processes	(m s^{-1} s^{-1})
double rthblten (Time,nCells,nVertLevels)	Tendency of potential temperature due to PBL processes	(K s^{-1})
double rqvblten (Time,nCells,nVertLevels)	Tendency of water vapor due to PBL processes	(kg kg^{-1} s^{-1})
double rqcblten (Time,nCells,nVertLevels)	Tendency of cloud water due to PBL processes	(kg kg^{-1} s^{-1})
double rqiblten (Time,nCells,nVertLevels)	Tendency of cloud ice due to PBL processes	(kg kg^{-1} s^{-1})
double kzh (Time,nCells,nVertLevels)	Vertical diffusion coefficient for heat	
double kzm (Time,nCells,nVertLevels)	Vertical diffusion coefficient for momentum	
double kzq (Time,nCells,nVertLevels)	Vertical diffusion coefficient for moisture	

D.4.5 Horizontal cloud fraction

double cldfrac (Time,nCells,nVertLevels)	Cloud fraction	(-)

D.4.6 Radiation schemes

SHORT-WAVE RADIATION SCHEMES:		
double coszr (Time,nCells)	Cosine of solar zenith angle	(-)
double gsw (Time,nCells)	SFC all-sky net shortwave radiation	(W m^{-2})
double swdnb (Time,nCells)	SFC all-sky downward shortwave radiation	(W m^{-2})
double swdnbc (Time,nCells)	SFC clear-sky downward shortwave radiation	(W m^{-2})
double swupb (Time,nCells)	SFC all-sky upward shortwave radiation	(W m^{-2})
double swupbc (Time,nCells)	SFC clear-sky upward shortwave radiation	(W m^{-2})
double swdnt (Time,nCells)	TOA all-sky downward shortwave radiation	(W m^{-2})
double swdntc (Time,nCells)	TOA clear-sky downward shortwave radiation	(W m^{-2})
double swupt (Time,nCells)	TOA all-sky upward shortwave radiation	(W m^{-2})
double swuptc (Time,nCells)	TOA clear-sky upward shortwave radiation	(W m^{-2})
double swcf (Time,nCells)	TOA all-sky shortwave radiative cloud forcing	(W m^{-2})

double i_acswdnb (Time,nCells)	Counter for bucket update of swdnb	(-)
double i_acswdnbc (Time,nCells)	Counter for bucket update of swdnbc	(-)
double i_acswdnt (Time,nCells)	Counter for bucket update of swdnt	(-)
double i_acswdntc (Time,nCells)	Counter for bucket update of swdntc	(-)
double i_acswupb (Time,nCells)	Counter for bucket update of swdupb	(-)
double i_acswupbc (Time,nCells)	Counter for bucket update of swupbc	(-)
double i_acswupt (Time,nCells)	Counter for bucket update of swupt	(-)
double i_acswuptc (Time,nCells)	Counter for bucket update of swuptc	(-)
double acswdnb (Time,nCells)	Accumulated swdnb	(W m^{-2})
double acswdnbc (Time,nCells)	Accumulated swdnbc	(W m^{-2})
double acswdnt (Time,nCells)	Accumulated swdnt	(W m^{-2})
double acswdntc (Time,nCells)	Accumulated swdntc	(W m^{-2})
double acswupb (Time,nCells)	Accumulated swupb	(W m^{-2})
double acswupbc (Time,nCells)	Accumulated swupbc	(W m^{-2})
double acswupt (Time,nCells)	Accumulated swupt	(W m^{-2})
double acswuptc (Time,nCells)	Accumulated swuptc	(W m^{-2})
double rthratensw (Time,nCells,nVertLevels)	Tendency of potential temperature due to all-sky SW radiation	(K s^{-1})
LONG-WAVE RADIATION SCHEMES:		
double glw (Time,nCells)	SFC all-sky net longwave radiation	(W m^{-2})
double lwdnb (Time,nCells)	SFC all-sky downward longwave radiation	(W m^{-2})
double lwdnbc (Time,nCells)	SFC clear-sky downward longwave radiation	(W m^{-2})
double lwupb (Time,nCells)	SFC all-sky upward longwave radiation	(W m^{-2})
double lwupbc (Time,nCells)	SFC clear-sky upward longwave radiation	(W m^{-2})
double lwdnt (Time,nCells)	TOA all-sky downward longwave radiation	(W m^{-2})
double lwdntc (Time,nCells)	TOA clear-sky downward longwave radiation	(W m^{-2})
double lwupt (Time,nCells)	TOA all-sky upward longwave radiation	(W m^{-2})
double lwuptc (Time,nCells)	TOA clear-sky upward longwave radiation	(W m^{-2})
double lwcf (Time,nCells)	TOA all-sky longwave radiative cloud forcing	(W m^{-2})
double olrtoa (Time,nCells)	TOA outgoing longwave radiation	(W m^{-2})
double i_aclwdnb (Time,nCells)	Counter for bucket update of lwdnb	(-)
double i_aclwdnbc (Time,nCells)	Counter for bucket update of lwdnbc	(-)
double i_aclwdnt (Time,nCells)	Counter for bucket update of lwdnt	(-)
double i_aclwdntc (Time,nCells)	Counter for bucket update of lwdntc	(-)
double i_aclwupb (Time,nCells)	Counter for bucket update of lwdupb	(-)
double i_aclwupbc (Time,nCells)	Counter for bucket update of lwupbc	(-)
double i_aclwupt (Time,nCells)	Counter for bucket update of lwupt	(-)
double i_aclwuptc (Time,nCells)	Counter for bucket update of lwuptc	(-)
double aclwdnb (Time,nCells)	Accumulated lwdnb	(W m^{-2})
double aclwdnbc (Time,nCells)	Accumulated lwdnbc	(W m^{-2})
double aclwdnt (Time,nCells)	Accumulated lwdnt	(W m^{-2})
double aclwdntc (Time,nCells)	Accumulated lwdntc	(W m^{-2})
double aclwupb (Time,nCells)	Accumulated lwupb	(W m^{-2})
double aclwupbc (Time,nCells)	Accumulated lwupbc	(W m^{-2})
double aclwupt (Time,nCells)	Accumulated lwupt	(W m^{-2})
double aclwuptc (Time,nCells)	Accumulated lwuptc	(W m^{-2})
double rthratenlw (Time,nCells,nVertLevels)	Tendency of potential temperature due to all-sky LW radiation	(K s^{-1})
RRTMG RADIATION SPECIFICS:		

integer nlrad (Time,nCells)	Number of layers added between model top and TOA	(-)
double plrad (Time nCells)	Model top pressure	(Pa)
CAM RADIATION SPECIFICS:		
double absnxt (Time,nCells,cam_dim1,nVertLevels)	Total nearest layer absorptivity	(-)
double abstot (Time,nCells,nVertLevelsP1,nVertLevelsP1)	Total absorptivity	(-)
double emstot (Time,nCells,nVertLevelsP1)	Total emissivity	(-)
double pin (nOznLevels)	Prescribed pressure levels for ozone mixing ratio	(hPa)
double ozmixm (nMonths nOznLevels nCells)	Climatological monthly ozone mixing ratio	(kg kg^{-1})
double m_hybi	Matched hybi (need to rechecked)	(-)
double m_ps	Surface pressure from match on MPAS grid	(Pa)
double sul (Time,nCells,nAerLevels)	Sulfate soluble (SUL) aerosol concentration	(kg m^{-3})
double sslt (Time,nCells,nAerLevels)	Sea-salt (SSLT) aerosol concentration	(kg m^{-3})
double dust1 (Time,nCells,nAerLevels)	Dust type 1 (DUST1) aerosol concentration	(kg m^{-3})
double dust2 (Time,nCells,nAerLevels)	Dust type 2 (DUST2) aerosol concentration	(kg m^{-3})
double dust3 (Time,nCells,nAerLevels)	Dust type 3 (DUST3) aerosol concentration	(kg m^{-3})
double dust4 (Time,nCells,nAerLevels)	Dust type 4 (DUST4) aerosol concentration	(kg m^{-3})
double ocpho (Time,nCells,nAerLevels)	Hydrophobic organic carbon (OCPHO) aerosol concentration	(kg m^{-3})
double bcpho (Time,nCells,nAerLevels)	Hydrophobic black carbon (BCPHO) aerosol concentration	(kg m^{-3})
double ocphi (Time,nCells,nAerLevels)	Hydrophilic organic carbon (OCPHI) aerosol concentration	(kg m^{-3})
double bcphi (Time,nCells,nAerLevels)	Hydrophilci black carbon (BCPHI) aerosol concentration	(kg m^{-3})
double bg (Time,nCells,nAerLevels)	Background (BG) aerosol concentration	(kg m^{-3})
double volc (Time,nCells,nAerLevels)	Volcanic (VOLC) aerosol concentration	(kg m^{-3})

D.4.7 Surface layer scheme

double br (Time,nCells)	Bulk Richardson number	(-)
double cd (Time,nCells)	10-meters drag coefficient	(-)
double cda (Time,nCells)	Drag coefficient at lowest model level	(-)
double chs (Time,nCells)	Heat exchange coefficient	
double chs2 (Time,nCells)	Heat exchange coefficient	
double cqs2 (Time,nCells)	Moisture exchange coefficient	
double ck (Time,nCells)	10-meters enthalpy exchange coefficient	(-)
double cka (Time,nCells)	Enthalpy exchange coefficient at lowest model level	(-)

double cpm (Time,nCells)		(-)
double fh (Time,nCells)	Integrated stability function for heat	
double fm (Time,nCells)	Integrated stability function for momentum	
double flhc (Time,nCells)	Exchange coefficient for heat	(-)
double flqc (Time,nCells)	Exchange coefficient for moisture	(-)
double gz1oz0 (Time,nCells)	Log of z1 over z0	(-)
double hfx (Time,nCells)	Surface upward heat flux	(W m^{-2})
double lh (Time,nCells)	Surface latent heat flux	(W m^{-2})
double mavail (Time,nCells)	Surface moisture availability	(-)
double mol (Time,nCells)	T^* in similarity theory	(K)
double psim (Time,nCells)	Similarity function for heat	(-)
double psih (Time,nCells)	Similarity function for momentum	(-)
double q2 (Time,nCells)	2-meters specific humidity	(kg kg^{-1})
double qfx (Time,nCells)	Upward moisture flux at the surface	(kg m^{-2} s^{-1})
double qgh (Time,nCells)	*To be defined as local. Will need to be removed from Registry*	
double qsfc (Time,nCells)	Specific humidity at lower boundary	(kg kg^{-1})
double regime (Time,nCells)	Flag indicating the PBL regime	(-)
double rmol (Time,nCells)	Inverse of Monin Obukhov length scale	(m^{-1})
double u10 (Time,nCells)	10-meters zonal wind	(m s^{-1})
double ust (Time,nCells)	u^* in similarity theory	(m s^{-1})
double ustm (Time,nCells)	u^* in similarity theory without the vconv term	(m s^{-1})
double t2m (Time,nCells)	2-meters temperature	(K)
double th2m (Time,nCells)	2-meters potential temperature	(K)
double v10 (Time,nCells)	10-meters meridional wind	(m s^{-1})
double wspd (Time,nCells)	Wind speed	(m s^{-1})
double zol (Time,nCells)	Height over Monin-Obukhov length scale	(-)
double znt (Time,nCells)	Roughness length	(m)

D.4.8 Gravity wave drag over orography

double var2d (nCells)	Orographic variance	
double con (nCells)	Orographic convexity	
double oa1 (nCells)	Orographic direction asymmetry function	
double oa2 (nCells)	Orographic direction asymmetry function	
double oa3 (nCells)	Orographic direction asymmetry function	
double oa4 (nCells)	Orographic direction asymmetry function	
double ol1 (nCells)	Orographic direction asymmetry function	
double ol2 (nCells)	Orographic direction asymmetry function	
double ol3 (nCells)	Orographic direction asymmetry function	
double ol4 (nCells)	Orographic direction asymmetry function	
double dusfcg (Time nCells)	Vertically-integrated gravity wave drag over orography u-stress	(Pa m s^{-1})
double dvsfcg (Time nCells)	Vertically-integrated gravity wave drag over orography v-stress	(Pa m s^{-1})
double dtaux3d (Time nCells)	Gravity wave drag over orography u-stress	(m s^{-1})
double dtauy3d (Time nCells)	Gravity wave drag over orography v-stress	(m s^{-1})

D.4.9 Land surface scheme

double acsnom (Time,nCells)	Accumulated melted snow on the ground	(kg m^{-2})
double acsnow (Time,nCells)	Accumulated snow on the ground	(kg m^{-2})
double canwat (Time,nCells)	Water content in canopy	(kg m^{-2})
double chklowq (Time,nCells)	Flag indicating surface saturation	(-)
double grdflx (Time,nCells)	Ground heat flux	(W m^{-2})
double lai (Time,nCells)	Leaf area index	(area/area)
double noahres (Time,nCells)	Residual for the Noah scheme surface energy budget	(W m^{-2})
double potevp (Time,nCells)	Accumulated potential evaporation	(W m^{-2})
double qz0 (Time,nCells)	Specific humidity at znt	(kg kg^{-1})
double rib (Time,nCells)	*To be defined as local. Will need to be removed from Registry*	
double sfc_albedo (Time,nCells)	Surface albedo	(-)
double sfc_emiss (Time,nCells)	Surface emissivity	(-)
double sfc_emibck (Time,nCells)	Background surface emissivity	(-)
double sfcrunoff (Time,nCells)	Surface runoff	(mm)
double smstav (Time,nCells)	Surface moisture availability	(-)
double smstot (Time,nCells)	Surface total moisture	(m^3 m^{-3})
double snopcx (Time,nCells)	Snow phase change heat flux	(W m^{-2})
double snotime (Time,nCells)	*Do not know*	
double sstsk (Time,nCells)	Skin sea-surface temperarute	(K)
double sstsk_diurn (Time,nCells)	Skin sea-surface temperature difference	(K)
double thc (Time,nCells)	Thermal inertia	(Cal cm^{-1} K^{-1} $s^{-0.5}$)
double udrunoff (Time,nCells)	Sub-surface runoff	(mm)
double xicem (Time,nCells)	Sea-ice flag from previous time-step	(-)
double z0 (Time,nCells)	Background roughness length	(m)
double zs (Time,nCells)	Depth of centers of soil layers	(m)

integer isltyp (nCells)	Soil type category	(-)
integer ivgtyp (nCells)	Vegetation type category	(-)
integer landmask (nCells)	Land-water mask (1=land, 0=water)	(-)
double shdmin (nCells)	Annual minimum greeness fraction	(-)
double shdmax (nCells)	Annual maximum greeness fraction	(-)
double snoalb (nCells)	Annual maximum snow albedo	(-)
double ter (nCells)	Terrain height	(m)
double albedo12m (nMonths,nCells)	Climatological monthly mean background albedo	(-)
double greenfrac (nMonths,nCells)	Climatological monthly mean greeness fraction	(-)
double sfc_albbck (Time,nCells)	Surface background albedo	(-)
double skintemp (Time,nCells)	Surface skin temperature	(K)
double snow (Time,nCells)	Snow water equivalent	(kg m^{-2})
double snowc (Time,nCells)	Flag indicating snow coverage (1 for snow; 0 otherwise)	(-)
double snowh (Time,nCells)	Physical snow depth	(m)
double sst (Time,nCells)	Sea-surface temperature	(K)
double tmn (Time,nCells)	Deep soil temperature	(K)
double vegfra (Time,nCells)	Vegetation fraction	(-)
double seaice (Time,nCells)	Sea-ice flag	(-)
double xice (Time,nCells)	Sea-ice fraction	(-)

double xland (Time,nCells)	Land ocean mask (1=land, 2=ocean)	(-)
double dzs (Time,nCells)	Soil layer thickness	(m)
double smcrel (Time,nCells)	Relative soil moisture	(-)
double sh2o (Time,nCells)	Soil liquid water	(m^3 m^{-3})
double smois (Time,nCells)	Soil layer moisture	(m^3 m^{-3})
double tslb (Time,nCells)	Soil layer temperature	(K)

Appendix E

MPAS Copyright

The MPAS-specific source code developed under this project has been copyrighted under a BSD license. MPAS is freely available and is open-source. Any external software components have their own copyright statement directly in their source code included with this release. The full copyright statement is:

OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Appendix F

Revision History

19 May 2015

- Added chapter describing the MPAS-Atmosphere physics suites introduced in MPAS v4.0.
- Added a section on building the model with single-precision reals.
- Updated the I/O chapter to include the new io_type option introduced in MPAS v4.0.
- Updated a few namelist options to match MPAS v4.0 code.
- Minor wording changes and corrections of typographical errors throughout document.

18 November 2014

- Added chapter describing the MPAS runtime I/O system available in MPAS v3.0.
- Updated the chapter on running MPAS-Atmosphere to match MPAS v3.0 code.
- Updated a few namelist options to match MPAS v3.0 code.
- Minor wording changes throughout document.

13 June 2013

Initial version.