MPAS-Atmosphere Model User's Guide

Version 5.2

August 1, 2017

Foreword

This user's guide describes the Model for Prediction Across Scales – Atmosphere (MPAS-A) Version 5.2. 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 and the land-ice core, 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 and MYNN
- Boundary-layer: YSU and MYNN PBL schemes
- Convection: Kain-Fritsch Tiedtke, New Tiedtke, and Grell-Freitas convection parameterizations
- Cloud microphysics: WSM6, Kessler, and Thompson 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.

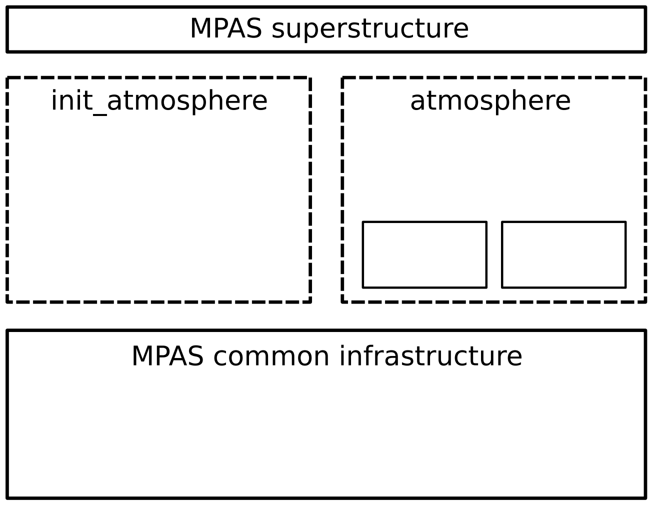


Figure 1.1: The initialization and model components of MPAS-A are built as separate *cores* within the MPAS framework.

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 high-level description of the general process of building and running the model. It is meant merely as a brief overview of the process, with 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`, `atmosphere_model`, and `build_tables` should have been created in the top-level MPAS directory. Once all three executables have been created, a complete model simulation can be generated 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 Prerequisites

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 Laboratory. 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. MPAS-Atmosphere v5.0 introduces the capability to use hybrid parallelism using MPI and OpenMP; however, the use of OpenMP *should be considered experimental* and generally does not offer any performance advantage. The primary reason for releasing a shared-memory capability is to make this code available to collaborators for future development.

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 cannot assume responsibility for third-party libraries.*

Although most recent versions of the NetCDF and Parallel-NetCDF libraries should work, the most tested versions of these libraries are NetCDF 4.4.x and Parallel-NetCDF 1.8.x. Users are strongly encouraged to use either the latest PIO 2.x version from `https://github.com/NCAR/ParallelIO/`, or PIO versions 1.7.1 or 1.9.23, as other versions have not been tested or are known to not work with MPAS. The NetCDF and Parallel-NetCDF libraries must be installed before building the PIO library.

3.2.1 NetCDF

Version 4.4.x of the NetCDF library may be downloaded from `http://www.unidata.ucar.edu/downloads/netcdf/index.jsp`. The Unidata page provides detailed instructions instructions for building the NetCDF C and Fortran libraries; both the C and Fortran interfaces are needed by PIO. If NetCDF-4 support is desired, the zlib and HDF5 libraries will need to be installed prior to building NetCDF. *Before proceeding to compile PIO the* `NETCDF` *environment variable should be set to the NetCDF root installation directory.*

3.2.2 Parallel-NetCDF

Version 1.8.x of the Parallel-NetCDF library may be downloaded from `https://trac.mcs.anl.gov/projects/parallel-netcdf/wiki/Download`. *Before proceeding to compile PIO the* `PNETCDF` *environment variable should be set to the Parallel-NetCDF root installation directory.*

3.2.3 PIO

Beginning with the MPAS v5.2 release, the either of the PIO 1.x or 2.x library versions may be used. The two major versions have slightly different APIs; by default, the MPAS build system assumes the PIO 1.x API, but the PIO 2.x library versions may be used by adding the `USE_PIO2=true` option when compiling MPAS as described in Section 3.3. If compiling with the PIO 1.x library versions, users are strongly encouraged to choose either PIO 1.7.1 or PIO 1.9.23, as other 1.x versions may not work; these two specific versions may be obtained from `https://github.com/NCAR/ParallelIO/releases/tag/pio1_7_1` and `https://github.com/NCAR/ParallelIO/releases/tag/pio1_9_23`.

The PIO 2.x library versions support integrated performance timing with the GPTL library; however, the MPAS infrastructure does not currently provide calls to initialize this library when it is used in PIO 2.x. Therefore, it is recommended to add `-DPIO_ENABLE_TIMING=OFF` when running the cmake command to build PIO 2.x versions.

After PIO is built and installed the `PIO` environment variable should be set to the directory where PIO was installed. Recent versions of PIO support the specification of an installation prefix, while some 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 land-ice 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 )
make[1]: Entering directory ‘/scratch/MPAS-Release’

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

Example targets:
    ifort
    gfortran
    xlf
    pgi

Availabe Cores:
    atmosphere
    init_atmosphere
    landice
    ocean
    sw
    test

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.
    TIMER_LIB=opt - Selects the timer library interface to be used for profiling the model.
                    Options are:
                    TIMER_LIB=native - Uses native built-in timers in MPAS
                    TIMER_LIB=gptl - Uses gptl for the timer interface instead of the native interface
                    TIMER_LIB=tau - Uses TAU for the timer interface instead of the native interface
    OPENMP=true   - builds and links with OpenMP flags. Default is to not use OpenMP.
    USE_PIO2=true - links with the PIO 2 library. Default is to use the PIO 1.x library.
    PRECISION=single - builds with default single-precision real kind.
                       Default is to use double-precision.

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. Beginning with version 5.0, the selection of the model precision can be made on the command-line, with no need to edit the `Makefile`. To compile a single-precision MPAS-Atmosphere executable, add `PRECISION=single` to the build command, e.g.,

```
> make gfortran CORE=atmosphere PRECISION=single
```

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 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. This is useful, e.g., to disable all parameterizations except for microphysics when running some idealized simulations. The details of selecting individual physics parameterizations are explained in Section 6.4.

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	New Tiedtke
Microphysics	WSM6
Land surface	Noah
Boundary layer	YSU
Surface layer	Monin-Obukhov
Radiation, LW	RRTMG
Radiation, SW	RRTMG
Cloud fraction for radiation	Xu-Randall
Gravity wave drag by orography	YSU

6.2 Suite: convection_permitting

The 'convection_permitting' physics suite is appropriate at spatial resolutions allowing for both explicitly resolved hydrostatic and nonhydrostatic motions. It has been tested for mesh spacings from several hundred kilometers down to 3 km in MPAS. The Grell-Freitas convection scheme transitions from a conventional parameterization of deep convection at hydrostatic scales (cell spacings of several tens of kilometers) to a parameterization of precipitating shallow convection at cell spacings less than 10 km. This is the recommended suite for any MPAS applications where convection-permitting meshes (dx $<$ 10 km) are employed, including variable-resolution meshes spanning hydrostatic to nonhydrostatic resolutions.

Table 6.2: The set of parameterization schemes used by the 'convection_permitting' physics suite.

Parameterization	Scheme
Convection	Grell-Freitas
Microphysics	Thompson (non-aerosol aware)
Land surface	Noah
Boundary layer	MYNN
Surface layer	MYNN
Radiation, LW	RRTMG
Radiation, SW	RRTMG
Cloud fraction for radiation	Xu-Randall
Gravity wave drag by orography	YSU

6.3 Suite: none

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'
```

6.4 Selecting individual physics parameterizations

Selecting or disabling an individual physics parameterization may be accomplished by setting the appropriate namelist variable to one of its possible options; possible options for individual parameterizations, along with details of those options, are given in Table 6.3. Note that all parameterization options may be set to `'off'` to disable the parameterization of the associated process.

Table 6.3: Possible options for individual physics parameterizations. Namelist variables should be added to the &physics namelist record.

Parameterization	Namelist variable	Possible options	Details
Convection	config_convection_scheme	`cu_tiedtke`	Tiedtke
		`cu_ntiedtke`	New Tiedtke (WRF 3.8.1)
		`cu_grell_freitas`	Modified version of scale-aware Grell-Freitas (WRF 3.6.1)
		`cu_kain_fritsch`	Kain-Fritsch (WRF 3.2.1)
Microphysics	config_microp_scheme	`mp_wsm6`	WSM 6-class (WRF 3.8.1)
		`mp_thompson`	Thompson non-aerosol aware (WRF 3.8.1)
		`mp_kessler`	Kessler
Land surface	config_lsm_scheme	`noah`	Noah (WRF 3.3.1)
Boundary layer	config_pbl_scheme	`bl_ysu`	YSU (WRF 3.8.1)
		`bl_mynn`	MYNN (WRF 3.6.1)
Surface layer	config_sfclayer_scheme	`sf_monin_obukhov`	Monin-Obukhov (WRF 3.8.1)
		`sf_mynn`	MYNN (WRF 3.6.1)
Radiation, LW	config_radt_lw_scheme	`rrtmg_lw`	RRTMG (WRF 3.8.1)
		`cam_lw`	CAM (WRF 3.3.1)
Radiation, SW	config_radt_sw_scheme	`rrtmg_sw`	RRTMG (WRF 3.8.1)
		`cam_sw`	
Cloud fraction for radiation	config_radt_cld_scheme	`cld_fraction`	Xu and Randall (1996)
		`cld_incidence`	0/1 cloud fraction depending on $q_c + q_i$
Gravity wave drag by orography	config_gwdo_scheme	`bl_ysu_gwdo`	YSU (WRF 3.6.1)

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 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

Creating idealized initial conditions is fairly straightforward, as no external data are required and the starting date/time is irrelevant to building the initial conditions file (hereafter referred to as `init.nc`) 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)

[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.

- 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
config_coef_3rd_order = 0.25	
/	
&dimensions	
config_nvertlevels = 26	the number of vertical levels to be used in the model
/	
&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 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 executions of the `init_atmosphere_model` program, 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 `static.nc` file requires a set of static geographic data. A suitable dataset can be obtained from the WRF model's download page
`http://www2.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 7.2.2 and 7.2.3, 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	the following dimensions should be set to 1 now, and their values will become significant in §7.2.3
config_nvertlevels = 1	
config_nsoillevels = 1	
config_nfglevels = 1	
config_nfgsoillevels = 1	
/	
&data_sources	
config_geog_data_path = '/WPS_GEOG/'	absolute path to static files obtained from the WRF download page
config_landuse_data = 'USGS'	land-use classification to use in MPAS-Atmosphere
/	
&preproc_stages	only the static_interp and native_gwd_static stages should be enabled
config_static_interp = true	

config_native_gwd_static = true	
config_vertical_grid = false	
config_met_interp = false	
config_input_sst = false	
/	

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://www2.mmm.ucar.edu/wrf/users/docs/user_guide_V3.8/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 input_sst and frac_seaice stages should be enabled
config_static_interp = false	
config_native_gwd_static = false	
config_vertical_grid = false	
config_met_interp = false	
config_input_sst = true	
config_frac_seaice = true	

/	
&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 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://www2.mmm.ucar.edu/wrf/users/docs/user_guide_V3.8/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
config_coef_3rd_order = 0.25	
/	
&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
config_landuse_data = 'USGS'	must be consistent with choice in §7.2.1

config_use_spechumd = true	if available, use specific humidity rather than relative humidity
/	
&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_native_gwd_static = false	
config_vertical_grid = true	only these three stages should be enabled
config_met_interp = true	
config_input_sst = false	
config_frac_seaice = true	
/	
&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 = 720.0	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
/	
&physics	
config_sst_update = true	if updating sea-ice and SST with an `surface.nc` file, set to `true`, and edit the "surface" stream in the `streams.atmosphere` file accordingly
config_physics_suite = 'mesoscale_reference'	
/	

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
iau	the stream used to read analysis increments for the Incremental Analysis Update (IAU) scheme

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.

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.

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

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.

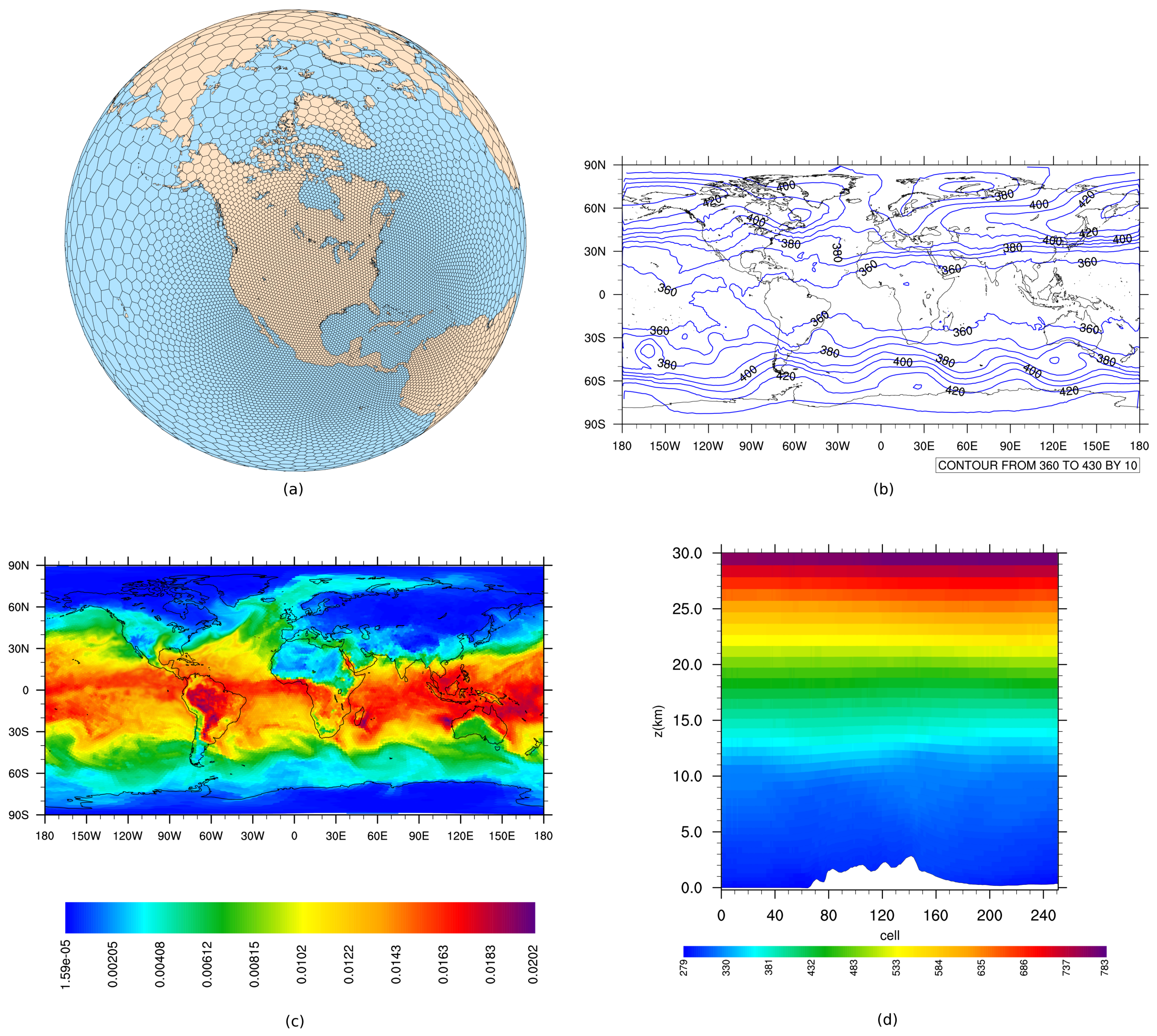


Figure 8.1: Various plot types that can be produced from MPAS input or output files using example scripts provided with the MPAS code. (a) A plot of an MPAS SCVT mesh against a filled map background. (b) A simple horizontal contour plot. (c) A cell-filled horizontal plot, with individual cells of the 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 brackets (`[ ]`) 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 (integer)

Units	-
Description	*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*
Possible Values	1 – 8 *(default: 7)*

config_calendar_type (character)

Units	-
Description	*Simulation calendar type (hidden by default)*
Possible Values	'gregorian','gregorian_noleap' *(default: gregorian)*

config_start_time (character)

Units	-
Description	*Time to begin processing first-guess data (cases 7 and 8 only)*
Possible Values	'YYYY-MM-DD_hh:mm:ss' *(default: 2010-10-23_00:00:00)*

config_stop_time (character)

Units	-
Description	*Time to end processing first-guess data (case 8 only)*
Possible Values	'YYYY-MM-DD_hh:mm:ss' *(default: 2010-10-23_00:00:00)*

config_theta_adv_order (integer)

Units	-
Description	*Horizontal advection order for theta*
Possible Values	2, 3, or 4 *(default: 3)*

config_coef_3rd_order (real)

Units	-
Description	*Upwinding coefficient in the 3rd order advection scheme*
Possible Values	$0 \leq$ config_coef_3rd_order ≤ 1 *(default: 0.25)*

config_num_halos (integer)

Units	-
Description	*Number of halo layers for fields (hidden by default)*
Possible Values	Integer values, typically 2 or 3; DO NOT CHANGE *(default: 2)*

A.2 dimensions

config_nvertlevels (integer)

Units	-
Description	*The number of vertical levels to be used in the model*
Possible Values	Positive integer values *(default: 41)*

config_nsoillevels (integer)

Units	-
Description	*The number of vertical soil levels needed by LSM in the model (case 7 only)*
Possible Values	Positive integer values *(default: 4)*

config_nfglevels (integer)

Units	-
Description	*The number of atmospheric levels (including surface and sea-level) in the first-guess dataset (case 7 only)*
Possible Values	Positive integer values *(default: 38)*

config_nfgsoillevels (integer)

Units	-
Description	*The number of vertical soil levels in the first-guess dataset (case 7 only)*
Possible Values	Positive integer values *(default: 4)*

config_months (integer)

Units	-
Description	*The number of months in a year (hidden by default)*
Possible Values	Positive integer values *(default: 12)*

A.3 data_sources

config_geog_data_path (character)

Units	-
Description	*Path to the WPS static data files (case 7 only)*
Possible Values	Any valid path *(default: /glade/p/work/wrfhelp/WPS_GEOG/)*

config_met_prefix (character)

Units	-
Description	*Filename prefix of ungrib intermediate file to use for initial conditions (case 7 only)*
Possible Values	Any alpha-numeric string *(default: CFSR)*

config_sfc_prefix (character)

Units	-
Description	*Filename prefix of ungrib intermediate file to use for SST and sea-ice (cases 7 and 8 only)*
Possible Values	Any alpha-numeric string *(default: SST)*

config_fg_interval (integer)

Units	-
Description	*Interval between SST and sea-ice files (case 8 only)*
Possible Values	[DDD_]hh:mm:ss *(default: 86400)*

config_landuse_data (character)

Units	-
Description	*The land use classification to use (case 7 only)*
Possible Values	'USGS' or 'MODIFIED_IGBP_MODIS_NOAH' *(default: USGS)*

config_use_spechumd (logical)

Units	-
Description	*Whether to use specific-humidity as the first-guess moisture variable. If this option is False, relative humidity will be used.*
Possible Values	true or false *(default: false)*

A.4 vertical_grid

config_ztop (real)

Units	*m*
Description	*Model top height*
Possible Values	Positive real values *(default: 30000.0)*

config_nsmterrain (integer)

Units	-
Description	*Number of smoothing passes to apply to the interpolated terrain field*
Possible Values	Non-negative integer values *(default: 1)*

config_smooth_surfaces (logical)

Units	-
Description	*Whether to smooth zeta surfaces*
Possible Values	true or false *(default: true)*

config_dzmin (real)

Units	-
Description	*Minimum thickness of layers as a fraction of nominal thickness*
Possible Values	Real values in the interval (0,1) *(default: 0.3)*

config_nsm (integer)

Units	-
Description	*Maximum number of smoothing passes for coordinate surfaces*
Possible Values	Positive integer values *(default: 30)*

config_tc_vertical_grid (logical)

Units	-
Description	*Whether to use the vertical layer profile that was developed for use in real-time TC experiments*
Possible Values	true or false *(default: true)*

A.5 interpolation_control

config_extrap_airtemp (character)

Units	-
Description	*Method of extrapolation of air temperature above/below first-guess levels.*
Possible Values	'constant' (last valid value), 'linear' (linear extrapolation based on last two values), 'lapse-rate' (0.0065 K/m from last valid value) *(default: linear)*

A.6 preproc_stages

config_static_interp (logical)

Units	-
Description	*Whether to interpolate WPS static data (case 7 only)*
Possible Values	true or false *(default: true)*

config_native_gwd_static (logical)

Units	-
Description	*Whether to recompute sub-grid-scale orography statistics directly on the native MPAS mesh (case 7 only)*
Possible Values	true or false *(default: true)*

config_gwd_cell_scaling (real)

Units	-
Description	*Scaling factor for the effective grid cell diameter used in computation of GWD static fields (hidden by default)*
Possible Values	Positive real values *(default: 1.0)*

config_vertical_grid (logical)

Units	-
Description	*Whether to generate vertical grid*
Possible Values	true or false *(default: true)*

config_met_interp (logical)

Units	-
Description	*Whether to interpolate first-guess fields from intermediate file*
Possible Values	true or false *(default: true)*

config_input_sst (logical)

Units	-
Description	*Whether to re-compute SST and sea-ice fields from surface input data set; should be set to .true. when running case 8*
Possible Values	true or false *(default: false)*

config_frac_seaice (logical)

Units	-
Description	*Whether to switch sea-ice threshold from 0.5 to 0.02*
Possible Values	true or false *(default: true)*

A.7 io

config_pio_num_iotasks (integer)

Units	-
Description	*Number of tasks to perform file I/O*
Possible Values	Integer valued, 0 ≤ config_pio_num_iotasks ≤ # MPI tasks, 0 indicates all tasks perform I/O *(default: 0)*

config_pio_stride (integer)

Units	-
Description	*Stride between file I/O tasks*
Possible Values	Integer valued, ≤ (# MPI tasks) / config_pio_num_iotasks *(default: 1)*

A.8 decomposition

config_block_decomp_file_prefix (character)

Units	-
Description	*Prefix of graph decomposition file, to be suffixed with the MPI task count*
Possible Values	Any valid filename *(default: x1.40962.graph.info.part.)*

config_number_of_blocks (integer)

Units	-
Description	*Number of blocks to assign to each MPI task (hidden by default)*
Possible Values	Positive integer values *(default: 0)*

config_explicit_proc_decomp (logical)

Units	-
Description	*Whether to use an explicit mapping of blocks to MPI tasks (hidden by default)*
Possible Values	.true. or .false. *(default: false)*

config_proc_decomp_file_prefix (character)

Units	-
Description	*Prefix of block mapping file (hidden by default)*
Possible Values	Any valid filename *(default: 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_time_integration (character)

Units	-
Description	*Time integration scheme (hidden by default)*
Possible Values	'SRK3' *(default: SRK3)*

config_time_integration_order (integer)

Units	-
Description	*Order for RK time integration*
Possible Values	2 or 3 *(default: 2)*

config_dt (real)

Units	*s*
Description	*Model time step, seconds*
Possible Values	Positive real values *(default: 720.0)*

config_calendar_type (character)

Units	-
Description	*Simulation calendar type (hidden by default)*
Possible Values	'gregorian','gregorian_noleap' *(default: gregorian)*

config_start_time (character)

Units	-
Description	*Starting time for model simulation*
Possible Values	'YYYY-MM-DD_hh:mm:ss' *(default: 2010-10-23_00:00:00)*

config_stop_time (character)

Units	-
Description	*Stopping time for model simulation (hidden by default)*
Possible Values	'YYYY-MM-DD_hh:mm:ss' *(default: none)*

config_run_duration (character)

Units	-
Description	*Length of model simulation*
Possible Values	[DDD_]hh:mm:ss *(default: 5_00:00:00)*

config_split_dynamics_transport (logical)

Units	-
Description	*Whether to super-cycle scalar transport*
Possible Values	Logical values *(default: true)*

config_number_of_sub_steps (integer)

Units	-
Description	*Number of acoustic steps per full RK step*
Possible Values	Positive, even integer values, typically 2 or 6 depending on transport splitting *(default: 2)*

config_dynamics_split_steps (integer)

Units	-
Description	*When config_split_dynamics_transport = T, the number of RK steps per transport step*
Possible Values	Positive integer values *(default: 3)*

config_h_mom_eddy_visc2 (real)

Units	*m^2 s^{-1}*
Description	∇^2 *eddy viscosity for horizontal diffusion of momentum*
Possible Values	Positive real values *(default: 0.0)*

config_h_mom_eddy_visc4 (real)

Units	*m^4 s^{-1}*
Description	∇^4 *eddy hyper-viscosity for horizontal diffusion of momentum*
Possible Values	Positive real values *(default: 0.0)*

config_v_mom_eddy_visc2 (real)

Units	*m^2 s^{-1}*
Description	∇^2 *eddy viscosity for vertical diffusion of momentum*
Possible Values	Positive real values *(default: 0.0)*

config_h_theta_eddy_visc2 (real)

Units	*m^2 s^{-1}*
Description	∇^2 *eddy viscosity for horizontal diffusion of theta*
Possible Values	Positive real values *(default: 0.0)*

config_h_theta_eddy_visc4 (real)

Units	*m^4 s^{-1}*
Description	∇^4 *eddy hyper-viscosity for horizontal diffusion of theta*
Possible Values	Positive real values *(default: 0.0)*

config_v_theta_eddy_visc2 (real)

Units	*m^2 s^{-1}*
Description	∇^2 *eddy viscosity for vertical diffusion of theta*
Possible Values	Positive real values *(default: 0.0)*

config_horiz_mixing (character)

Units	-
Description	*Formulation of horizontal mixing*
Possible Values	'2d_fixed' or '2d_smagorinsky' *(default: 2d_smagorinsky)*

config_len_disp (real)

Units	m
Description	*Horizontal length scale, used by the Smagorinsky formulation of horizontal diffusion and by 3-d divergence damping*
Possible Values	Positive real values *(default: 120000.0)*

config_visc4_2dsmag (real)

Units	-
Description	*Scaling coefficient of δx^3 to obtain ∇^4 diffusion coefficient*
Possible Values	Non-negative real values *(default: 0.05)*

config_del4u_div_factor (real)

Units	-
Description	*Scaling factor for the divergent component of $\nabla^4 u$ calculation (hidden by default)*
Possible Values	Positive real values *(default: 10.0)*

config_w_adv_order (integer)

Units	-
Description	*Horizontal advection order for w*
Possible Values	2, 3, or 4 *(default: 3)*

config_theta_adv_order (integer)

Units	-
Description	*Horizontal advection order for theta*
Possible Values	2, 3, or 4 *(default: 3)*

config_scalar_adv_order (integer)

Units	-
Description	*Horizontal advection order for scalars*
Possible Values	2, 3, or 4 *(default: 3)*

config_u_vadv_order (integer)

Units	-
Description	*Vertical advection order for normal velocities (u)*
Possible Values	2, 3, or 4 *(default: 3)*

config_w_vadv_order (integer)

Units	-
Description	*Vertical advection order for w*
Possible Values	2, 3, or 4 *(default: 3)*

config_theta_vadv_order (integer)

Units	-
Description	*Vertical advection order for theta*
Possible Values	2, 3, or 4 *(default: 3)*

config_scalar_vadv_order (integer)

Units	-
Description	*Vertical advection order for scalars*
Possible Values	2, 3, or 4 *(default: 3)*

config_scalar_advection (logical)

Units	-
Description	*Whether to advect scalar fields*
Possible Values	.true. or .false. *(default: true)*

config_positive_definite (logical)

Units	-
Description	*Whether to enable positive-definite advection of scalars*
Possible Values	.true. or .false. *(default: false)*

config_monotonic (logical)

Units	-
Description	*Whether to enable monotonic limiter in scalar advection*
Possible Values	.true. or .false. *(default: true)*

config_coef_3rd_order (real)

Units	-
Description	*Upwinding coefficient in the 3rd order advection scheme*
Possible Values	$0 \leq$ config_coef_3rd_order ≤ 1 *(default: 0.25)*

config_smagorinsky_coef (real)

Units	-
Description	*Dimensionless empirical parameter relating the strain tensor to the eddy viscosity in the Smagorinsky turbulence model (hidden by default)*
Possible Values	Real values typically in the range 0.1 to 0.4 *(default: 0.125)*

config_mix_full (logical)

Units	-
Description	*Mix full θ and u fields, or mix perturbation from intitial state (hidden by default)*
Possible Values	.true. or .false. *(default: true)*

config_epssm (real)

Units	-
Description	*Off-centering parameter for the vertically implicit acoustic integration*
Possible Values	Positive real values *(default: 0.1)*

config_smdiv (real)

Units	-
Description	*3-d divergence damping coefficient*
Possible Values	Positive real values *(default: 0.1)*

config_apvm_upwinding (real)

Units	-
Description	*Amount of upwinding in APVM (hidden by default)*
Possible Values	$0 \leq$ config_apvm_upwinding ≤ 1 *(default: 0.5)*

config_h_ScaleWithMesh (logical)

Units	-
Description	*Scale eddy viscosities with mesh-density function for horizontal diffusion (hidden by default)*
Possible Values	.true. or .false. *(default: true)*

config_num_halos (integer)

Units	-
Description	*Number of halo layers for fields (hidden by default)*
Possible Values	Integer values, typically 2 or 3; DO NOT CHANGE *(default: 2)*

B.2 damping

config_zd (real)

Units	*m*
Description	*Height MSL to begin w-damping profile*
Possible Values	Positive real values *(default: 22000.0)*

config_xnutr (real)

Units	-
Description	*Maximum w-damping coefficient at model top*
Possible Values	$0 \leq$ config_xnutr ≤ 1 *(default: 0.2)*

B.3 io

config_restart_timestamp_name (character)

Units	-
Description	*Filename used to store most recent restart time stamp (hidden by default)*
Possible Values	Any valid filename *(default: restart_timestamp)*

config_pio_num_iotasks (integer)

Units	-
Description	*Number of tasks to perform file I/O*
Possible Values	Integer valued, 0 ≤ config_pio_num_iotasks ≤ # MPI tasks, 0 indicates all tasks perform I/O *(default: 0)*

config_pio_stride (integer)

Units	-
Description	*Stride between file I/O tasks*
Possible Values	Integer valued, ≤ (# MPI tasks) / config_pio_num_iotasks *(default: 1)*

B.4 decomposition

config_block_decomp_file_prefix (character)

Units	-
Description	*Prefix of graph decomposition file, to be suffixed with the MPI task count*
Possible Values	Any valid filename *(default: x1.40962.graph.info.part.)*

config_number_of_blocks (integer)

Units	-
Description	*Number of blocks to assign to each MPI task (hidden by default)*
Possible Values	Positive integer values *(default: 0)*

config_explicit_proc_decomp (logical)

Units	-
Description	*Whether to use an explicit mapping of blocks to MPI tasks (hidden by default)*
Possible Values	.true. or .false. *(default: false)*

config_proc_decomp_file_prefix (character)

Units	-
Description	*Prefix of block mapping file (hidden by default)*
Possible Values	Any valid filename *(default: graph.info.part.)*

B.5 restart

config_do_restart (logical)

Units	-
Description	*Whether this run of the model is to restart from a previous restart file or not*
Possible Values	.true. or .false. *(default: false)*

config_do_DAcycling (logical)

Units	-
Description	*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 (hidden by default)*
Possible Values	.true. or .false. *(default: false)*

B.6 printout

config_print_global_minmax_vel (logical)

Units	-
Description	*Whether to print the global min/max of horizontal normal velocity and vertical velocity each timestep*
Possible Values	.true. or .false. *(default: true)*

config_print_detailed_minmax_vel (logical)

Units	-
Description	*Whether to print the global min/max of horizontal normal velocity and vertical velocity each timestep, along with the location in the domain where those extrema occurred*
Possible Values	.true. or .false. *(default: false)*

config_print_global_minmax_sca (logical)

Units	-
Description	*Whether to print the global min/max of scalar fields each timestep (hidden by default)*
Possible Values	.true. or .false. *(default: false)*

B.7 IAU

config_IAU_option (character)

Units	-
Description	*Incremental Analysis Update scheme*
Possible Values	'off' or 'on'; 'off' turns off IAU, 'on' uses equal weighting of increments across the IAU window *(default: off)*

config_IAU_window_length_s (real)

Units	*s*
Description	*Length of window over which analysis increments are applied*
Possible Values	Non-negative real values *(default: 21600.)*

B.8 physics

input_soil_data (character)

Units	-
Description	*input soil use classification (hidden by default)*
Possible Values	'STAS' *(default: STAS)*

input_soil_temperature_lag (integer)

Units	-
Description	*days over which the deep soil temperature is computed using skin temperature (hidden by default)*
Possible Values	Positive integers *(default: 140)*

num_soil_layers (integer)

Units	-
Description	*number of soil layers in Noah land surface scheme (hidden by default)*
Possible Values	Positive integers. For Noah LSM, must be set to 4. *(default: 4)*

months (integer)

Units	-
Description	*number of months per year (hidden by default)*
Possible Values	Positive integer values. DO NOT CHANGE. *(default: 12)*

noznlev (integer)

Units	-
Description	*number of prescribed pressure levels for input climatological ozone volume mixing ratios (hidden by default)*
Possible Values	Positive integers *(default: 59)*

naerlev (integer)

Units	-
Description	*number of prescribed pressure levels for input climatological aerosol mixing ratio (hidden by default)*
Possible Values	Positive integers *(default: 29)*

camdim1 (integer)

Units	-
Description	*dimension of CAM radiation absorption save array (hidden by default)*
Possible Values	Positive integers *(default: 4)*

config_frac_seaice (logical)

Units	-
Description	*logical for configuration of fractional sea-ice (hidden by default)*
Possible Values	.true. for sea-ice between 0 or 1; .false. for sea-ice equal to 0 or 1 (flag). *(default: true)*

config_sfc_albedo (logical)

Units	-
Description	*logical for configuration of surface albedo (hidden by default)*
Possible Values	.true. for climatologically varying surface albedo; .false. for fixed input data *(default: true)*

config_sfc_snowalbedo (logical)

Units	-
Description	*logical for configuration of maximum surface albedo for snow (hidden by default)*
Possible Values	.true. for geographical distribution; .false. for fixed input data *(default: true)*

config_sst_update (logical)

Units	-
Description	*logical for configuration of sea-surface temperature*
Possible Values	.true. for time-varying sea-surface temperatures; .false., otherwise *(default: false)*

config_sstdiurn_update (logical)

Units	-
Description	*logical for configuration of diurnal cycle of sea-surface temperatures*
Possible Values	.true. for applying a diurnal cycle to sea-surface temperatures; .false., otherwise *(default: false)*

config_deepsoiltemp_update (logical)

Units	-
Description	*logical for configuration of deep soil temperatures*
Possible Values	.true. for slowly time-varying deep soil temperatures; .false. otherwise *(default: false)*

config_o3climatology (logical)

Units	-
Description	*logical for configuration of input ozone data in RRMTG long- and short-wave radiation (hidden by default)*
Possible Values	.true. for using monthly-varying ozone data; .false. for using fixed vertical profile *(default: false)*

config_microp_re (logical)

Units	-
Description	*logical for calculation of the effective radii for cloud water, cloud ice, and snow (hidden by default)*
Possible Values	.true. for calculating effective radii; .false. for using defaults in RRTMG radiation *(default: false)*

config_ysu_pblmix (logical)

Units	-
Description	*logical for turning on/off top-down, radiation_driven mixing (hidden by default)*
Possible Values	.true. to turn on top-down radiation_driven mixing; .false. otherwise *(default: false)*

config_n_microp (integer)

Units	-
Description	*number of microphysics time-steps per physics time-steps (hidden by default)*
Possible Values	Positive integers *(default: 1)*

config_radtlw_interval (character)

Units	-
Description	*time interval between calls to parameterization of long-wave radiation*
Possible Values	'DD_HH:MM:SS' or 'none' *(default: 00:30:00)*

config_radtsw_interval (character)

Units	-
Description	*time interval between calls to parameterization of short-wave radiation*
Possible Values	'DD_HH:MM:SS' or 'none' *(default: 00:30:00)*

config_conv_interval (character)

Units	-
Description	*time interval between calls to parameterization of convection (hidden by default)*
Possible Values	'DD_HH:MM:SS' or 'none' *(default: none)*

config_pbl_interval (character)

Units	-
Description	*time interval between calls to parameterization of planetary boundary layer (hidden by default)*
Possible Values	'DD_HH:MM:SS' or 'none' *(default: none)*

config_camrad_abs_update (character)

Units	-
Description	*time interval between updates of absorption/emission coefficients in CAM radiation (hidden by default)*
Possible Values	'DD_HH:MM:SS' or 'none' *(default: 06:00:00)*

config_greeness_update (character)

Units	-
Description	*time interval between updates of greeness fraction (hidden by default)*
Possible Values	'DD_HH:MM:SS' or 'none' *(default: 24:00:00)*

config_bucket_update (character)

Units	-
Description	*time interval between updates of accumulated rain and radiation diagnostics*
Possible Values	'DD_HH:MM:SS' or 'none' *(default: none)*

config_physics_suite (character)

Units	-
Description	*Choice of physics suite*
Possible Values	'mesoscale_reference','convection_permitting','none' *(default: mesoscale_reference)*

config_microp_scheme (character)

Units	-
Description	*configuration for cloud microphysics schemes (hidden by default)*
Possible Values	'suite','mp_wsm6','mp_thompson','mp_kessler','off' *(default: suite)*

config_convection_scheme (character)

Units	-
Description	*configuration for convection schemes (hidden by default)*
Possible Values	'suite','cu_kain_fritsch','cu_tiedtke','cu_ntiedtke','cu_grell_freitas','off' *(default: suite)*

config_lsm_scheme (character)

Units	-
Description	*configuration for land-surface schemes (hidden by default)*
Possible Values	'suite','noah','off' *(default: suite)*

config_pbl_scheme (character)

Units	-
Description	*configuration for planetary boundary layer schemes (hidden by default)*
Possible Values	'suite','bl_ysu','bl_mynn','off' *(default: suite)*

config_gwdo_scheme (character)

Units	-
Description	*configuration of gravity wave drag over orography (hidden by default)*
Possible Values	'suite','bl_ysu_gwdo','off' *(default: suite)*

config_radt_cld_scheme (character)

Units	-
Description	*configuration for calculation of horizontal cloud fraction (hidden by default)*
Possible Values	'suite','cld_fraction','cld_incidence' *(default: suite)*

config_radt_lw_scheme (character)

Units	-
Description	*configuration for long-wave radiation schemes (hidden by default)*
Possible Values	'suite','rrtmg_lw','cam_lw','off' *(default: suite)*

config_radt_sw_scheme (character)

Units	-
Description	*configuration for short-wave radiation schemes (hidden by default)*
Possible Values	'suite','rrtmg_sw','cam_sw','off' *(default: suite)*

config_sfclayer_scheme (character)

Units	-
Description	*configuration for surface layer-scheme (hidden by default)*
Possible Values	'suite','sf_monin_obukhov','sf_mynn','off' *(default: suite)*

config_gfconv_closure_deep (integer)

Units	-
Description	*closure option for deep convection in Grell-Freitas convection scheme (hidden by default)*
Possible Values	0 *(default: 0)*

config_gfconv_closure_shallow (integer)

Units	-
Description	*closure option for shallow convection in Grell-Freitas convection scheme (hidden by default)*
Possible Values	8 *(default: 8)*

config_bucket_radt (real)

Units	-
Description	*threshold above which accumulated radiation diagnostics are reset (hidden by default)*
Possible Values	Positive real values *(default: 1.0e9)*

config_bucket_rainc (real)

Units	-
Description	*threshold above which the accumulated convective precipitation is reset (hidden by default)*
Possible Values	Positive real values *(default: 100.0)*

config_bucket_rainnc (real)

Units	-
Description	*threshold above which the accumulated grid-scale precipitation is reset (hidden by default)*
Possible Values	Positive real values *(default: 100.0)*

config_oml1d (logical)

Units	-
Description	*Whether to activate the 1-d ocean mixed-layer model (hidden by default)*
Possible Values	.true. or .false. *(default: false)*

config_oml_hml0 (real)

Units	*m*
Description	*If greater than zero, provides the constant, initial mixed-layer depth; if zero, initial mixed-layer depth will be taken from the 'oml_initial' field. (hidden by default)*
Possible Values	Non-negative real values *(default: 30.0)*

config_oml_gamma (real)

Units	$K\ m^{-1}$
Description	*Deep water lapse rate in 1-d OML model (hidden by default)*
Possible Values	Real values *(default: 0.14)*

config_oml_relaxation_time (real)

Units	*s*
Description	*Relaxation time to initial values in 1-d OML (hidden by default)*
Possible Values	Non-negative real values *(default: 864000.)*

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).

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).

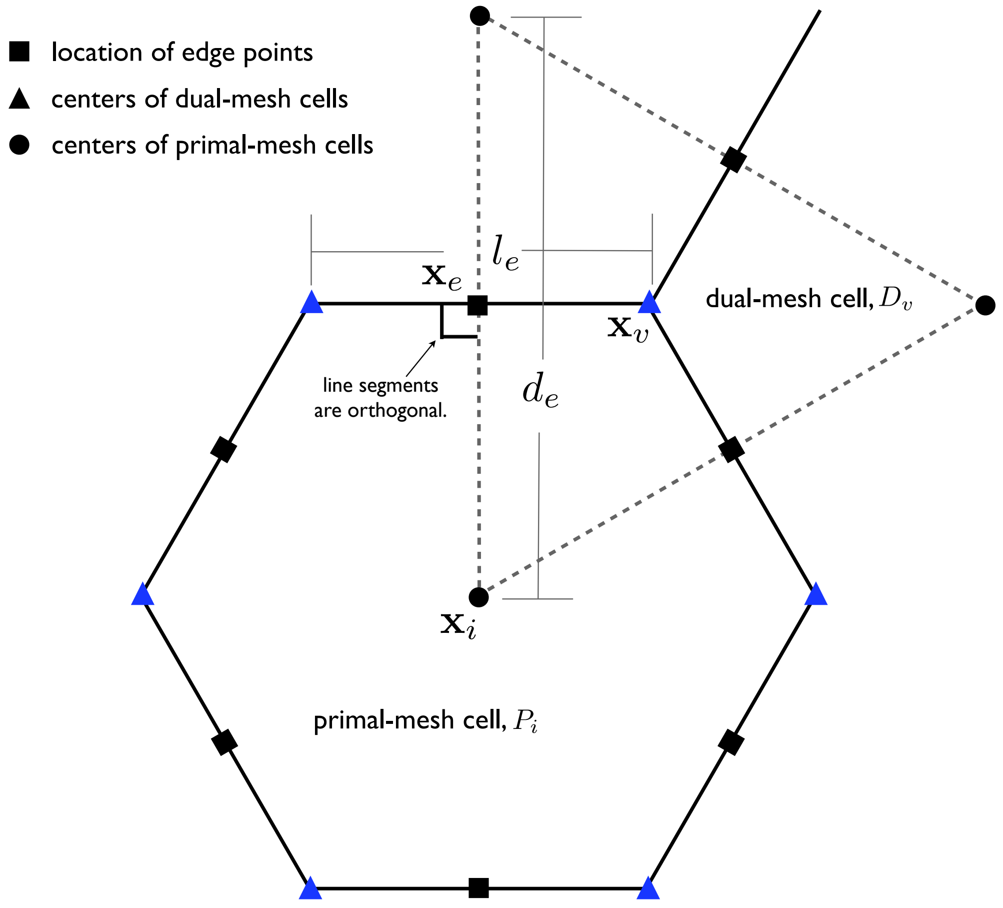

■ location of edge points
▲ centers of dual-mesh cells
● centers of primal-mesh cells
l_e
$\mathbf{x}_e$
$\mathbf{x}_v$
dual-mesh cell, D_v
line segments are orthogonal.
d_e
$\mathbf{x}_i$
primal-mesh cell, P_i

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

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

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

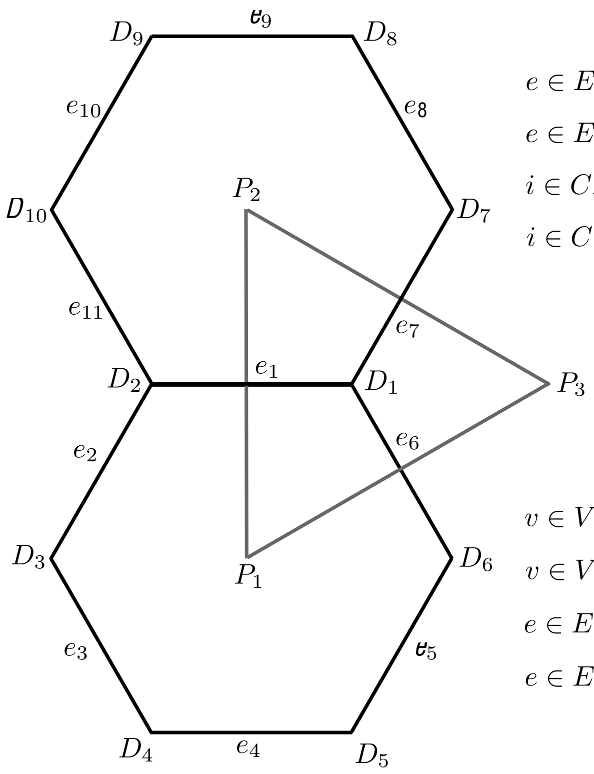
where α = angleEdge.

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.

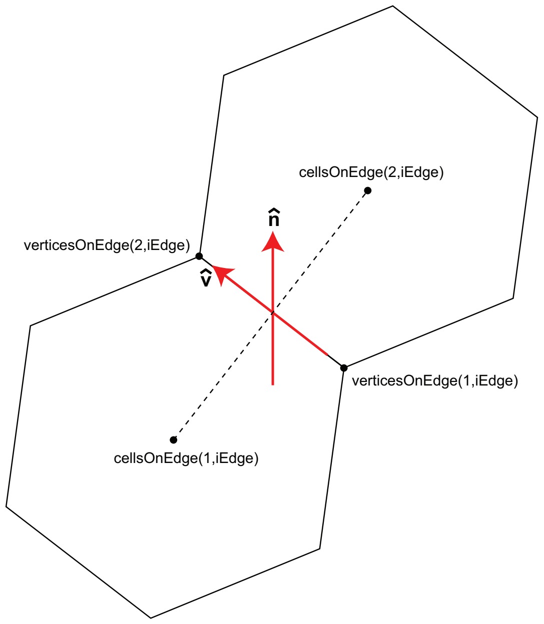


Figure C.3: The angle of an edge refers to the angle between a vector pointing north at an edge location and a vector pointing in the positive tangential velocity direction of the edge.

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

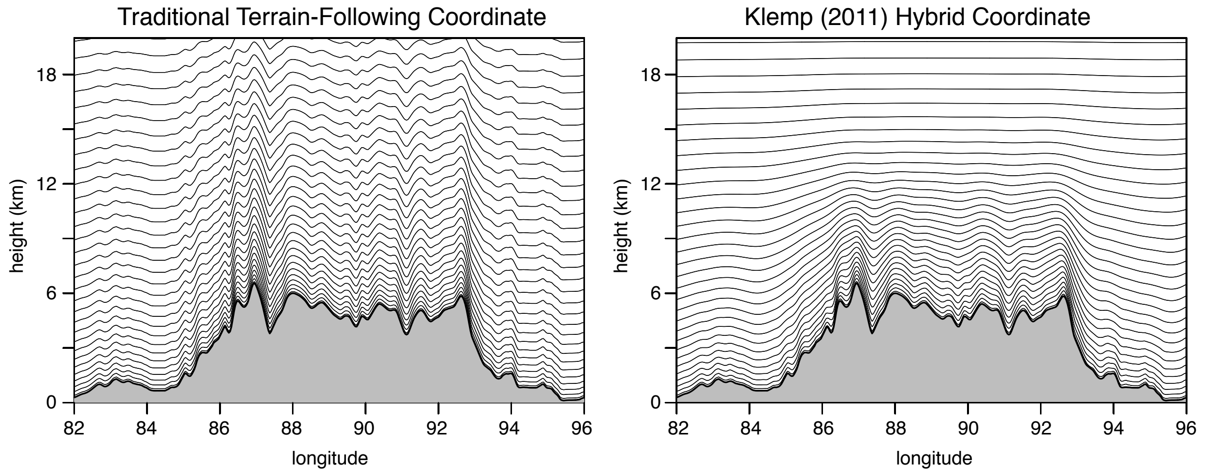


Figure C.4: Example MPAS-A vertical meshes using terrain following (left) and smoothed (right) vertical coordinates.

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.

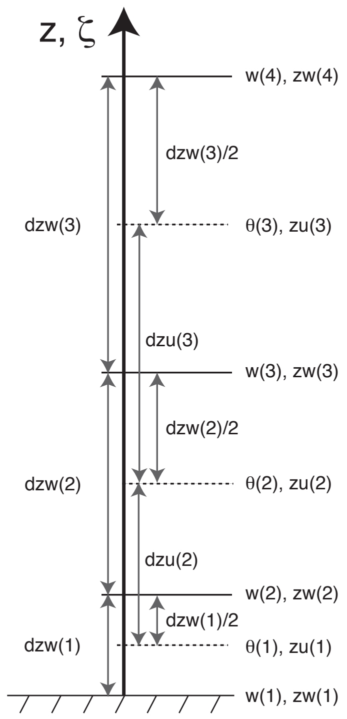


Figure C.5: Vertical distribution of the variables in MPAS-A. Also see Table C.4.

Appendix D

Description of Model Fields

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

a_tri (real) (nVertLevels, nCells, Time)

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

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

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

abstot (real) (nVertLevelsP1, nVertLevelsP1, nCells, Time)

Units	-
Description	*Total absorptivity*
Accessed in code	as ‘abstot’ from the ‘diag_physics’ pool

aclwdnb (real) (nCells, Time)

Units	*W m^{-2}*
Description	*accumulated all-sky downward surface longwave radiation flux*
Accessed in code	as ‘aclwdnb’ from the ‘diag_physics’ pool

aclwdnbc (real) (nCells, Time)

Units	*W m^{-2}*
Description	*accumulated clear-sky downward surface longwave radiation flux*
Accessed in code	as ‘aclwdnbc’ from the ‘diag_physics’ pool

aclwdnt (real) (nCells, Time)

Units	*W m^{-2}*
Description	*accumulated clear-sky downward surface longwave radiation flux*
Accessed in code	as ‘aclwdnt’ from the ‘diag_physics’ pool

aclwdntc (real) (nCells, Time)

Units	*W m^{-2}*
Description	*accumulated clear-sky downward surface longwave radiation flux*
Accessed in code	as ‘aclwdntc’ from the ‘diag_physics’ pool

aclwupb (real) (nCells, Time)

Units	*W m^{-2}*
Description	*accumulated all-sky upward surface longwave radiation flux*
Accessed in code	as ‘aclwupb’ from the ‘diag_physics’ pool

aclwupbc (real) (nCells, Time)

Units	*W m^{-2}*
Description	*accumulated all-sky upward surface longwave radiation flux*
Accessed in code	as ‘aclwupbc’ from the ‘diag_physics’ pool

aclwupt (real) (nCells, Time)

Units	*W m^{-2}*
Description	*accumulated all-sky upward surface longwave radiation flux*
Accessed in code	as ‘aclwupt’ from the ‘diag_physics’ pool

aclwuptc (real) (nCells, Time)

Units	*W m^{-2}*
Description	*accumulated all-sky upward surface longwave radiation flux*
Accessed in code	as ‘aclwuptc’ from the ‘diag_physics’ pool

acsnom (real) (nCells, Time)

Units	*kg m^{-2}*
Description	*accumulated melted snow*
Accessed in code	as ‘acsnom’ from the ‘diag_physics’ pool

acsnow (real) (nCells, Time)

Units	*kg m^{-2}*
Description	*accumulated snow*
Accessed in code	as ‘acsnow’ from the ‘diag_physics’ pool

acswdnb (real) (nCells, Time)

Units	*W m^{-2}*
Description	*accumulated all-sky downward surface shortwave radiation flux*
Accessed in code	as ‘acswdnb’ from the ‘diag_physics’ pool

acswdnbc (real) (nCells, Time)

Units	*W m^{-2}*
Description	*accumulated clear-sky downward surface shortwave radiation flux*
Accessed in code	as ‘acswdnbc’ from the ‘diag_physics’ pool

acswdnt (real) (nCells, Time)

Units	*W m^{-2}*
Description	*accumulated all-sky downward top-of-atmosphere shortwave radiation flux*
Accessed in code	as ‘acswdnt’ from the ‘diag_physics’ pool

acswdntc (real) (nCells, Time)

Units	*W m^{-2}*
Description	*accumulated clear-sky downward top-of-atmosphere shortwave radiation flux*
Accessed in code	as ‘acswdntc’ from the ‘diag_physics’ pool

acswupb (real) (nCells, Time)

Units	*W m*$^{-2}$
Description	*accumulated all-sky upward surface shortwave radiation flux*
Accessed in code	as ‘acswupb’ from the ‘diag_physics’ pool

acswupbc (real) (nCells, Time)

Units	*W m*$^{-2}$
Description	*accumulated clear-sky upward surface shortwave radiation flux*
Accessed in code	as ‘acswupbc’ from the ‘diag_physics’ pool

acswupt (real) (nCells, Time)

Units	*W m*$^{-2}$
Description	*accumulated all-sky upward top-of-atmosphere shortwave radiation flux*
Accessed in code	as ‘acswupt’ from the ‘diag_physics’ pool

acswuptc (real) (nCells, Time)

Units	*W m*$^{-2}$
Description	*accumulated clear-sky upward top-of-atmosphere shortwave radiation flux*
Accessed in code	as ‘acswuptc’ from the ‘diag_physics’ pool

aerosols (real) (nAerLevels, nCells, Time)

Description	*‘aerosols’ is a variable array used in code which is made up of the constituent fields listed below.*
Accessed in code	as ‘aerosols’ from the ‘state’ pool

Contains constituents:

sul — *constituent field of var_array* ***aerosols***

Description	*Sulfate soluble (SUL) aerosol concentration*
Units	*kg m*$^{-3}$
Array Group	aer_cam

sslt — *constituent field of var_array* ***aerosols***

Description	*Sea-salt (SSLT) aerosol concentration*
Units	*kg m^{-3}*
Array Group	aer_cam

dust1 — *constituent field of var_array* ***aerosols***

Description	*Dust type 1 (DUST1) aerosol concentration*
Units	*kg m^{-3}*
Array Group	aer_cam

dust2 — *constituent field of var_array* ***aerosols***

Description	*Dust type 2 (DUST2) aerosol concentration*
Units	*kg m^{-3}*
Array Group	aer_cam

dust3 — *constituent field of var_array* ***aerosols***

Description	*Dust type 3 (DUST3) aerosol concentration*
Units	*kg m^{-3}*
Array Group	aer_cam

dust4 — *constituent field of var_array* ***aerosols***

Description	*Dust type 4 (DUST4) aerosol concentration*
Units	*kg m^{-3}*
Array Group	aer_cam

ocpho — *constituent field of var_array* ***aerosols***

Description	*Hydrophobic organic carbon (OCPHO) aerosol concentration*
Units	*kg m^{-3}*
Array Group	aer_cam

bcpho — *constituent field of var_array* ***aerosols***

Description	*Hydrophobic black carbon (BCPHO) aerosol concentration*
Units	*kg m^{-3}*
Array Group	aer_cam

ocphi — *constituent field of var_array* ***aerosols***

Description	*Hydrophilic organic carbon (OCPHI) aerosol concentration*
Units	*kg m^{-3}*
Array Group	aer_cam

bcphi — *constituent field of var_array* ***aerosols***

Description	*Hydrophilic black carbon (BCPHI) aerosol concentration*
Units	*kg m^{-3}*
Array Group	aer_cam

bg — *constituent field of var_array* ***aerosols***

Description	*Background (BG) aerosol concentration*
Units	*kg m^{-3}*
Array Group	aer_cam

volc — *constituent field of var_array* ***aerosols***

Description	*Volcanic (VOLC) aerosol concentration*
Units	*kg m^{-3}*
Array Group	aer_cam

albedo12m (real) (nMonths, nCells)

Units	*unitless*
Description	*monthly-mean climatological aurface albedo*
Accessed in code	as ‘albedo12m’ from the ‘sfc_input’ pool

alpha_tri (real) (nVertLevels, nCells, Time)

Units	*unitless*
Description	*implicit tridiagonal solve coefficients*
Accessed in code	as ‘alpha_tri’ from the ‘diag’ pool

bcphi — *see ‘aerosols’*

bcpho — *see ‘aerosols’*

bg — *see 'aerosols'*

br (real) (nCells, Time)

Units	*unitless*
Description	*Richardson number*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as 'br' from the 'diag_physics' pool

canwat (real) (nCells, Time)

Units	*kg m^{-2}*
Description	*water in canopy*
Accessed in code	as 'canwat' from the 'diag_physics' pool

cd (real) (nCells, Time)

Units	*unitless*
Description	*drag coefficient at 10-meter*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as 'cd' from the 'diag_physics' pool

cda (real) (nCells, Time)

Units	*unitless*
Description	*drag coefficient at lowest model level*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as 'cda' from the 'diag_physics' pool

ch (real) (nCells, Time)

Units	*m s^{-1}*
Description	*surface exchange coefficient for heat*
Allocated by	bl_mynn_in
Accessed in code	as 'ch' from the 'diag_physics' pool

chklowq (real) (nCells, Time)

Units	*unitless*
Description	*surface saturation flag*
Accessed in code	as ‘chklowq’ from the ‘diag_physics’ pool

chs (real) (nCells, Time)

Units	*m s^{-1}*
Description	*surface exchange coefficient for heat and moisture*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as ‘chs’ from the ‘diag_physics’ pool

chs2 (real) (nCells, Time)

Units	*m s^{-1}*
Description	*surface exchange coefficient for heat at 2-meter*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as ‘chs2’ from the ‘diag_physics’ pool

circulation (real) (nVertLevels, nVertices, Time)

Units	*m^2 s^{-1}*
Description	*Horizontal circulation at vertices*
Accessed in code	as ‘circulation’ from the ‘diag’ pool

ck (real) (nCells, Time)

Units	*unitless*
Description	*enthalpy exchange coeff at 10-meter*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as ‘ck’ from the ‘diag_physics’ pool

cka (real) (nCells, Time)

Units	*unitless*
Description	*enthalpy exchange coefficient at lowest model level*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as ‘cka’ from the ‘diag_physics’ pool

cldfrac (real) (nVertLevels, nCells, Time)

Units	*unitless*
Description	*horizontal cloud fraction*
Accessed in code	as ‘cldfrac’ from the ‘diag_physics’ pool

cofrz (real) (nVertLevels, Time)

Units	*s m*$^{-1}$
Description	*coefficient for implicit contribution of Omega to density update*
Accessed in code	as ‘cofrz’ from the ‘diag’ pool

coftz (real) (nVertLevelsP1, nCells, Time)

Units	*s K*
Description	*coefficient for implicit contribution of omega vertical derivative to the theta_m update*
Accessed in code	as ‘coftz’ from the ‘diag’ pool

cofwr (real) (nVertLevels, nCells, Time)

Units	*m s*$^{-1}$
Description	*coefficient for implicit contribution of density to the vertical velocity update*
Accessed in code	as ‘cofwr’ from the ‘diag’ pool

cofwt (real) (nVertLevels, nCells, Time)

Units	*m s*$^{-1}$ *K*$^{-1}$
Description	*coefficient for implicit contribution of density to the vertical velocity update*
Accessed in code	as ‘cofwt’ from the ‘diag’ pool

cofwz (real) (nVertLevels, nCells, Time)

Units	*m s*$^{-1}$ *K*$^{-1}$
Description	*coefficient for implicit contribution of density to the vertical velocity update*
Accessed in code	as ‘cofwz’ from the ‘diag’ pool

con (real) (nCells)

Units	*m^2*
Description	*convexity of orography*
Accessed in code	as 'con' from the 'sfc_input' pool

coszr (real) (nCells, Time)

Units	*unitless*
Description	*cosine of zenith solar angle*
Accessed in code	as 'coszr' from the 'diag_physics' pool

cov (real) (nVertLevels, nCells, Time)

Units	*$K\ kg\ kg^{-1}$*
Description	*liquid water - liquid water potential temperature covariance*
Allocated by	bl_mynn_in
Accessed in code	as 'cov' from the 'diag_physics' pool

cpm (real) (nCells, Time)

Units	*$J\ K^{-1}\ kg^{-1}$*
Description	*specific heat of dry air at constant pressure at lowest model level*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as 'cpm' from the 'diag_physics' pool

cqs2 (real) (nCells, Time)

Units	*$m\ s^{-1}$*
Description	*surface exchange coefficient for moisture at 2-meter*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as 'cqs2' from the 'diag_physics' pool

cqu (real) (nVertLevels, nEdges, Time)

Units	*unitless*
Description	*rho_d/rho_m at cell edge (u points)*
Accessed in code	as 'cqu' from the 'diag' pool

cqw (real) (nVertLevels, nCells, Time)

Units	*unitless*
Description	*rho_d/rho_m at w points*
Accessed in code	as ‘cqw’ from the ‘diag’ pool

cubot (real) (nCells, Time)

Units	*unitless*
Description	*index of highest level of convection*
Allocated by	cu_grell_freitas_in, cu_kain_fritsch_in
Accessed in code	as ‘cubot’ from the ‘diag_physics’ pool

cuprec (real) (nCells, Time)

Units	*mm s^{-1}*
Description	*convective precipitation rate*
Allocated by	cu_grell_freitas_in, cu_kain_fritsch_in, cu_tiedtke_in
Accessed in code	as ‘cuprec’ from the ‘diag_physics’ pool

cutop (real) (nCells, Time)

Units	*unitless*
Description	*index of lowest level of convection*
Allocated by	cu_grell_freitas_in, cu_kain_fritsch_in
Accessed in code	as ‘cutop’ from the ‘diag_physics’ pool

delta (real) (nCells, Time)

Units	*m*
Description	*entrainment layer depth from PBL scheme*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as ‘delta’ from the ‘diag_physics’ pool

divergence (real) (nVertLevels, nCells, Time)

Units	*s^{-1}*
Description	*Horizontal velocity divergence at cell center*
Accessed in code	as ‘divergence’ from the ‘diag’ pool

dqke (real) (nVertLevels, nCells, Time)

Units	*m^2 s^{-2}*
Description	*TKE change*
Allocated by	bl_mynn_in
Accessed in code	as ‘dqke’ from the ‘diag_physics’ pool

dtaux3d (real) (nVertLevels, nCells, Time)

Units	*m s^{-1}*
Description	*gravity wave drag over orography u-stress*
Accessed in code	as ‘dtaux3d’ from the ‘diag_physics’ pool

dtauy3d (real) (nVertLevels, nCells, Time)

Units	*m s^{-1}*
Description	*gravity wave drag over orography v-stress*
Accessed in code	as ‘dtauy3d’ from the ‘diag_physics’ pool

dusfcg (real) (nCells, Time)

Units	*Pa m s^{-1}*
Description	*vertically-integrated gravity wave drag over orography u-stress*
Accessed in code	as ‘dusfcg’ from the ‘diag_physics’ pool

dust1 — *see ‘aerosols’*

dust2 — *see ‘aerosols’*

dust3 — *see ‘aerosols’*

dust4 — *see ‘aerosols’*

dvsfcg (real) (nCells, Time)

Units	*Pa m s^{-1}*
Description	*vertically-integrated gravity wave drag over orography v-stress*
Accessed in code	as ‘dvsfcg’ from the ‘diag_physics’ pool

dzs (real) (nSoilLevels, nCells, Time)

Units	*m*
Description	*soil layer thickness*
Accessed in code	as ‘dzs’ from the ‘sfc_input’ pool

el_pbl (real) (nVertLevels, nCells, Time)

Units	*m*
Description	*mixing length from PBL scheme*
Allocated by	bl_mynn_in
Accessed in code	as ‘el_pbl’ from the ‘diag_physics’ pool

emstot (real) (nVertLevelsP1, nCells, Time)

Units	-
Description	*Total emissivity*
Accessed in code	as ‘emstot’ from the ‘diag_physics’ pool

euler_tend_theta (real) (nVertLevels, nCells, Time)

Units	*kg K m^{-3} s^{-1}*
Description	*tendency of coupled potential temperature rho*theta_m/zz from dynamics and physics that does not change over a Runge-Kutta timestep (it excludes the flux divergence)*
Accessed in code	as ‘theta_euler’ from the ‘tend’ pool

euler_tend_u (real) (nVertLevels, nEdges, Time)

Units	*m s^{-2}*
Description	*Tendency of u from dynamics*
Accessed in code	as ‘u_euler’ from the ‘tend’ pool

euler_tend_w (real) (nVertLevelsP1, nCells, Time)

Units	*m s^{-2}*
Description	*Tendency of w from dynamics*
Accessed in code	as ‘w_euler’ from the ‘tend’ pool

evapprod (real) (nVertLevels, nCells, Time)

Units	*s^{-1}*
Description	*rain evaporation rate*
Allocated by	mp_thompson_in
Accessed in code	as ‘evapprod’ from the ‘diag_physics’ pool

exch_h (real) (nVertLevels, nCells, Time)

Units	*$m^2\ s^{-1}$*
Description	*exchange coefficient*
Allocated by	bl_ysu_in
Accessed in code	as ‘exch_h’ from the ‘diag_physics’ pool

exner (real) (nVertLevels, nCells, Time)

Units	*unitless*
Description	*Exner function*
Accessed in code	as ‘exner’ from the ‘diag’ pool

exner_base (real) (nVertLevels, nCells, Time)

Units	*unitless*
Description	*Base-state Exner function*
Accessed in code	as ‘exner_base’ from the ‘diag’ pool

fh (real) (nCells, Time)

Units	*unitless*
Description	*integrated stability function for heat*
Allocated by	bl_ysu_in
Accessed in code	as ‘fh’ from the ‘diag_physics’ pool

flhc (real) (nCells, Time)

Units	*$W\ m^{-2}\ K^{-1}$*
Description	*exchange coefficient for heat*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as ‘flhc’ from the ‘diag_physics’ pool

flqc (real) (nCells, Time)

Units	*kg m^{-2} s^{-1}*
Description	*exchange coefficient for moisture*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as ‘flqc’ from the ‘diag_physics’ pool

fm (real) (nCells, Time)

Units	*unitless*
Description	*integrated stability function for moisture*
Allocated by	bl_ysu_in
Accessed in code	as ‘fm’ from the ‘diag_physics’ pool

gamma_tri (real) (nVertLevels, nCells, Time)

Units	*unitless*
Description	*implicit tridiagonal solve coefficients*
Accessed in code	as ‘gamma_tri’ from the ‘diag’ pool

glw (real) (nCells, Time)

Units	*W m^{-2}*
Description	*all downward surface longwave radiation*
Accessed in code	as ‘glw’ from the ‘diag_physics’ pool

gradPVn (real) (nVertLevels, nEdges, Time)

Units	*???*
Description	*Gradient in the normal direction of PV at edge locations*
Accessed in code	as ‘gradPVn’ from the ‘diag’ pool

gradPVt (real) (nVertLevels, nEdges, Time)

Units	*???*
Description	*Gradient in the tangential direction of PV at edge locations*
Accessed in code	as ‘gradPVt’ from the ‘diag’ pool

graupelnc (real) (nCells, Time)

Units	*mm*
Description	*accumulated grid-scale precipitation of graupel*
Allocated by	mp_thompson_in, mp_wsm6_in
Accessed in code	as 'graupelnc' from the 'diag_physics' pool

graupelncv (real) (nCells, Time)

Units	*mm*
Description	*time-step grid-scale precipitation of graupel*
Allocated by	mp_thompson_in, mp_wsm6_in
Accessed in code	as 'graupelncv' from the 'diag_physics' pool

grdflx (real) (nCells, Time)

Units	*W m^{-2}*
Description	*ground heat flux*
Accessed in code	as 'grdflx' from the 'diag_physics' pool

greenfrac (real) (nMonths, nCells)

Units	*unitless*
Description	*monthly-mean climatological greeness fraction*
Accessed in code	as 'greenfrac' from the 'sfc_input' pool

gsw (real) (nCells, Time)

Units	*W m^{-2}*
Description	*net surface shortwave radiation flux*
Accessed in code	as 'gsw' from the 'diag_physics' pool

gz1oz0 (real) (nCells, Time)

Units	*unitless*
Description	*log(z/z0) where z0 is roughness length*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as 'gz1oz0' from the 'diag_physics' pool

h_divergence (real) (nVertLevels, nCells, Time)

Units	*???*
Description	*???*
Accessed in code	as 'h_divergence' from the 'diag' pool

h_oml (real) (nCells, Time)

Units	*m*
Description	*ocean mixed layer depth*
Accessed in code	as 'h_oml' from the 'diag_physics' pool

h_oml_initial (real) (nCells, Time)

Units	*m*
Description	*ocean mixed layer depth at initial time*
Accessed in code	as 'h_oml_initial' from the 'diag_physics' pool

hfx (real) (nCells, Time)

Units	$W\ m^{-2}$
Description	*upward heat flux at the surface*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as 'hfx' from the 'diag_physics' pool

hpbl (real) (nCells, Time)

Units	*m*
Description	*Planetary Boundary Layer (PBL) height*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as 'hpbl' from the 'diag_physics' pool

hu_oml (real) (nCells, Time)

Units	$m^2\ s^{-1}$
Description	*ocean mixed layer integrated u (zonal velocity)*
Accessed in code	as 'hu_oml' from the 'diag_physics' pool

hv_oml (real) (nCells, Time)

Units	*m^2 s^{-1}*
Description	*ocean mixed layer integrated v (meridional velocity)*
Accessed in code	as 'hv_oml' from the 'diag_physics' pool

i_aclwdnb (integer) (nCells, Time)

Units	*unitless*
Description	*incidence of accumulated all-sky downward surface longwave radiation greater than config_bucket_radt*
Accessed in code	as 'i_aclwdnb' from the 'diag_physics' pool

i_aclwdnbc (integer) (nCells, Time)

Units	*unitless*
Description	*incidence of accumulated clear-sky downward surface longwave radiation greater than config_bucket_radt*
Accessed in code	as 'i_aclwdnbc' from the 'diag_physics' pool

i_aclwdnt (integer) (nCells, Time)

Units	*unitless*
Description	*incidence of accumulated all-sky downward top-of-atmosphere longwave radiation greater than config_bucket_radt*
Accessed in code	as 'i_aclwdnt' from the 'diag_physics' pool

i_aclwdntc (integer) (nCells, Time)

Units	*unitless*
Description	*incidence of accumulated clear-sky downward top-of-atmosphere longwave radiation greater than config_bucket_radt*
Accessed in code	as 'i_aclwdntc' from the 'diag_physics' pool

i_aclwupb (integer) (nCells, Time)

Units	*unitless*
Description	*incidence of accumulated all-sky upward surface longwave radiation greater than config_bucket_radt*
Accessed in code	as 'i_aclwupb' from the 'diag_physics' pool

i_aclwupbc (integer) (nCells, Time)

Units	*unitless*
Description	*incidence of accumulated clear-sky upward surface longwave radiation greater than config_bucket_radt*
Accessed in code	as 'i_aclwupbc' from the 'diag_physics' pool

i_aclwupt (integer) (nCells, Time)

Units	*unitless*
Description	*incidence of accumulated all-sky upward top-of-atmosphere longwave radiation greater than config_bucket_radt*
Accessed in code	as 'i_aclwupt' from the 'diag_physics' pool

i_aclwuptc (integer) (nCells, Time)

Units	*unitless*
Description	*incidence of accumulated clear-sky upward top-of-atmosphere longwave radiation greater than config_bucket_radt*
Accessed in code	as 'i_aclwuptc' from the 'diag_physics' pool

i_acswdnb (integer) (nCells, Time)

Units	*unitless*
Description	*incidence of accumulated all-sky downward surface shortwave radiation greater than config_bucket_radt*
Accessed in code	as 'i_acswdnb' from the 'diag_physics' pool

i_acswdnbc (integer) (nCells, Time)

Units	*unitless*
Description	*incidence of accumulated clear-sky downward surface shortwave radiation greater than config_bucket_radt*
Accessed in code	as 'i_acswdnbc' from the 'diag_physics' pool

i_acswdnt (integer) (nCells, Time)

Units	*unitless*
Description	*incidence of accumulated all-sky downward top-of-atmosphere shortwave radiation greater than config_bucket_radt*
Accessed in code	as 'i_acswdnt' from the 'diag_physics' pool

i_acswdntc (integer) (nCells, Time)

Units	*unitless*
Description	*incidence of accumulated clear-sky downward top-of-atmosphere shortwave radiation greater than config_bucket_radt*
Accessed in code	as 'i_acswdntc' from the 'diag_physics' pool

i_acswupb (integer) (nCells, Time)

Units	*unitless*
Description	*incidence of accumulated all-sky upward surface shortwave radiation greater than config_bucket_radt*
Accessed in code	as 'i_acswupb' from the 'diag_physics' pool

i_acswupbc (integer) (nCells, Time)

Units	*unitless*
Description	*incidence of accumulated clear-sky upward surface shortwave radiation greater than config_bucket_radt*
Accessed in code	as 'i_acswupbc' from the 'diag_physics' pool

i_acswupt (integer) (nCells, Time)

Units	*unitless*
Description	*incidence of accumulated all-sky upward top-of-atmosphere shortwave radiation greater than config_bucket_radt*
Accessed in code	as 'i_acswupt' from the 'diag_physics' pool

i_acswuptc (integer) (nCells, Time)

Units	*unitless*
Description	*incidence of accumulated clear-sky upward top-of-atmosphere shortwave radiation greater than config_bucket_radt*
Accessed in code	as 'i_acswuptc' from the 'diag_physics' pool

i_rainc (integer) (nCells, Time)

Units	*unitless*
Description	*incidence of accumulated convective precipitation greater than config_bucket_rainc*
Allocated by	cu_grell_freitas_in, cu_kain_fritsch_in, cu_tiedtke_in
Accessed in code	as 'i_rainc' from the 'diag_physics' pool

i_rainnc (integer) (nCells, Time)

Units	*unitless*
Description	*incidence of accumulated grid-scale precipitation greater than config_bucket_rainnc*
Allocated by	mp_kessler_in, mp_thompson_in, mp_wsm6_in
Accessed in code	as ‘i_rainnc’ from the ‘diag_physics’ pool

isltyp (integer) (nCells)

Units	*unitless*
Description	*dominant soil category*
Accessed in code	as ‘isltyp’ from the ‘sfc_input’ pool

ivgtyp (integer) (nCells)

Units	*unitless*
Description	*dominant vegetation category*
Accessed in code	as ‘ivgtyp’ from the ‘sfc_input’ pool

k22_shallow (integer) (nCells, Time)

Units	*unitless*
Description	*index of convection at k22 level in Grell-Freitas convection scheme*
Allocated by	cu_grell_freitas_in
Accessed in code	as ‘k22_shallow’ from the ‘diag_physics’ pool

kbcon_shallow (integer) (nCells, Time)

Units	*unitless*
Description	*index of lowest level of shallow convection in Grell-Freitas convection scheme*
Allocated by	cu_grell_freitas_in
Accessed in code	as ‘kbcon_shallow’ from the ‘diag_physics’ pool

kdiff (real) (nVertLevels, nCells, Time)

Units	$m^2\ s^{-1}$
Description	*Smagorinsky horizontal eddy viscosity*
Accessed in code	as ‘kdiff’ from the ‘diag’ pool

ke (real) (nVertLevels, nCells, Time)

Units	*m^2 s^{-2}*
Description	*Kinetic energy at a cell center*
Accessed in code	as ‘ke’ from the ‘diag’ pool

kpbl (integer) (nCells, Time)

Units	*unitless*
Description	*index level of PBL top*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as ‘kpbl’ from the ‘diag_physics’ pool

ktop_deep (integer) (nCells, Time)

Units	*unitless*
Description	*index of highest level of deep convection in Grell-Freitas convection scheme*
Allocated by	cu_grell_freitas_in
Accessed in code	as ‘ktop_deep’ from the ‘diag_physics’ pool

ktop_shallow (integer) (nCells, Time)

Units	*unitless*
Description	*index of highest level of shallow convection in Grell-Freitas convection scheme*
Allocated by	cu_grell_freitas_in
Accessed in code	as ‘ktop_shallow’ from the ‘diag_physics’ pool

kzh (real) (nVertLevels, nCells, Time)

Units	*m^2 s^{-1}*
Description	*vertical diffusion coefficient of potential temperature*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as ‘kzh’ from the ‘diag_physics’ pool

kzm (real) (nVertLevels, nCells, Time)

Units	*m^2 s^{-1}*
Description	*vertical diffusion coefficient of mommentum*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as ‘kzm’ from the ‘diag_physics’ pool

kzq (real) (nVertLevels, nCells, Time)

Units	*m^2 s^{-1}*
Description	*vertical diffusion coefficient of water vapor and cloud condensates*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as ‘kzq’ from the ‘diag_physics’ pool

lai (real) (nCells, Time)

Units	*m^{-2} m^{-2}*
Description	*leaf area index*
Accessed in code	as ‘lai’ from the ‘diag_physics’ pool

landmask (integer) (nCells)

Units	*unitless*
Description	*land-ocean mask (1=land ; 0=ocean)*
Accessed in code	as ‘landmask’ from the ‘sfc_input’ pool

lh (real) (nCells, Time)

Units	*W m^{-2}*
Description	*latent heat flux at the surface*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as ‘lh’ from the ‘diag_physics’ pool

lwcf (real) (nCells, Time)

Units	*W m^{-2}*
Description	*top-of-atmosphere cloud longwave radiative forcing*
Accessed in code	as ‘lwcf’ from the ‘diag_physics’ pool

lwdnb (real) (nCells, Time)

Units	*W m^{-2}*
Description	*all-sky downward surface longwave radiation flux*
Accessed in code	as ‘lwdnb’ from the ‘diag_physics’ pool

lwdnbc (real) (nCells, Time)

Units	*W m^{-2}*
Description	*clear-sky downward surface longwave radiation flux*
Accessed in code	as ‘lwdnbc’ from the ‘diag_physics’ pool

lwdnt (real) (nCells, Time)

Units	*W m^{-2}*
Description	*clear-sky downward surface longwave radiation flux*
Accessed in code	as ‘lwdnt’ from the ‘diag_physics’ pool

lwdntc (real) (nCells, Time)

Units	*W m^{-2}*
Description	*clear-sky downward surface longwave radiation flux*
Accessed in code	as ‘lwdntc’ from the ‘diag_physics’ pool

lwupb (real) (nCells, Time)

Units	*W m^{-2}*
Description	*all-sky upward surface longwave radiation flux*
Accessed in code	as ‘lwupb’ from the ‘diag_physics’ pool

lwupbc (real) (nCells, Time)

Units	*W m^{-2}*
Description	*all-sky upward surface longwave radiation flux*
Accessed in code	as ‘lwupbc’ from the ‘diag_physics’ pool

lwupt (real) (nCells, Time)

Units	*W m^{-2}*
Description	*all-sky upward surface longwave radiation flux*
Accessed in code	as ‘lwupt’ from the ‘diag_physics’ pool

lwuptc (real) (nCells, Time)

Units	*W m^{-2}*
Description	*all-sky upward surface longwave radiation flux*
Accessed in code	as 'lwuptc' from the 'diag_physics' pool

m_ps (real) (nCells, Time)

Units	*Pa*
Description	*Surface pressure from match on MPAS grid*
Accessed in code	as 'm_ps' from the 'state' pool

mavail (real) (nCells, Time)

Units	*unitless*
Description	*surface moisture availability (between 0 and 1)*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as 'mavail' from the 'diag_physics' pool

mminlu (text) ()

Units	*unitless*
Description	*land use classification*
Accessed in code	as 'mminlu' from the 'sfc_input' pool

mol (real) (nCells, Time)

Units	*K*
Description	*T* in similarity theory*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as 'mol' from the 'diag_physics' pool

mu_c (real) (nCells)

Units	*unitless*
Description	*gamma shape parameter*
Allocated by	mp_thompson_in
Accessed in code	as 'mu_c' from the 'diag_physics' pool

nca (real) (nCells, Time)

Units	*s*
Description	*relaxation time for KF parameterization of convection*
Allocated by	cu_kain_fritsch_in
Accessed in code	as ‘nca’ from the ‘diag_physics’ pool

ndays_accum (real) (nCells, Time)

Units	*unitless*
Description	*number of accumulated days in a year*
Accessed in code	as ‘ndays_accum’ from the ‘diag_physics’ pool

ni — *see ‘scalars’*

nlrad (integer) (nCells, Time)

Units	-
Description	*number of layers added above the model-top in the RRTMG lw radiation code*
Accessed in code	as ‘nlrad’ from the ‘diag_physics’ pool

noahres (real) (nCells, Time)

Units	$W\ m^{-2}$
Description	*residual of the Noah surface energy budget*
Accessed in code	as ‘noahres’ from the ‘diag_physics’ pool

nr — *see ‘scalars’*

nsteps_accum (real) (nCells, Time)

Units	*unitless*
Description	*number of accumulated time-steps in a day*
Accessed in code	as ‘nsteps_accum’ from the ‘diag_physics’ pool

nt_c (real) (nCells)

Units	*nb kg^{-1}*
Description	*one-moment constant cloud water droplet concentration*
Allocated by	mp_thompson_in
Accessed in code	as ‘nt_c’ from the ‘diag_physics’ pool

o3clim (real) (nOznLevels, nCells, Time)

Units	*mol mol^{-1}*
Description	*climatological ozone on prescribed pressure levels at current time*
Accessed in code	as ‘o3clim’ from the ‘diag_physics’ pool

o3vmr (real) (nVertLevels, nCells, Time)

Units	*mol mol^{-1}*
Description	*ozone volume mixing ratio*
Accessed in code	as ‘o3vmr’ from the ‘diag_physics’ pool

oa1 (real) (nCells)

Units	*unitless*
Description	*directional asymmetry function of orography*
Accessed in code	as ‘oa1’ from the ‘sfc_input’ pool

oa2 (real) (nCells)

Units	*unitless*
Description	*directional asymmetry function of orography*
Accessed in code	as ‘oa2’ from the ‘sfc_input’ pool

oa3 (real) (nCells)

Units	*unitless*
Description	*directional asymmetry function of orography*
Accessed in code	as ‘oa3’ from the ‘sfc_input’ pool

oa4 (real) (nCells)

Units	*unitless*
Description	*directional asymmetry function of orography*
Accessed in code	as 'oa4' from the 'sfc_input' pool

ocphi — *see 'aerosols'*

ocpho — *see 'aerosols'*

ol1 (real) (nCells)

Units	*unitless*
Description	*directional asymmetry function of orography*
Accessed in code	as 'ol1' from the 'sfc_input' pool

ol2 (real) (nCells)

Units	*unitless*
Description	*directional asymmetry function of orography*
Accessed in code	as 'ol2' from the 'sfc_input' pool

ol3 (real) (nCells)

Units	*unitles*
Description	*directional asymmetry function of orography*
Accessed in code	as 'ol3' from the 'sfc_input' pool

ol4 (real) (nCells)

Units	*unitless*
Description	*directional asymmetry function of orography*
Accessed in code	as 'ol4' from the 'sfc_input' pool

olrtoa (real) (nCells, Time)

Units	*W m*$^{-2}$
Description	*top-of-atmosphere outgoing longwave radiation flux*
Accessed in code	as 'olrtoa' from the 'diag_physics' pool

ozmixm (real) (nMonths, nOznLevels, nCells)

Units	*mol mol^{-1}*
Description	*monthly-mean climatological ozone defined at fixed pressure levels*
Accessed in code	as ‘ozmixm’ from the ‘atm_input’ pool

pin (real) (nOznLevels)

Units	*Pa*
Description	*fixed pressure levels at which climatological ozone is defined*
Accessed in code	as ‘pin’ from the ‘atm_input’ pool

plrad (real) (nCells, Time)

Units	*Pa*
Description	*pressure at model-top*
Accessed in code	as ‘plrad’ from the ‘diag_physics’ pool

potevp (real) (nCells, Time)

Units	*m*
Description	*accumulated potential evaporation*
Accessed in code	as ‘potevp’ from the ‘diag_physics’ pool

precipw (real) (nCells, Time)

Units	*kg m^{-2}*
Description	*precipitable water*
Accessed in code	as ‘precipw’ from the ‘diag_physics’ pool

pressure (real) (nVertLevels, nCells, Time)

Units	*Pa*
Description	*Pressure*
Accessed in code	as ‘pressure’ from the ‘diag’ pool

pressure_base (real) (nVertLevels, nCells, Time)

Units	*Pa*
Description	*Base state pressure*
Accessed in code	as ‘pressure_base’ from the ‘diag’ pool

pressure_p (real) (nVertLevels, nCells, Time)

Units	*Pa*
Description	*Perturbation pressure*
Accessed in code	as 'pressure_p' from the 'diag' pool

psih (real) (nCells, Time)

Units	*unitless*
Description	*similarity stability function for heat*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as 'psih' from the 'diag_physics' pool

psim (real) (nCells, Time)

Units	*unitless*
Description	*similarity stability function for momentum*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as 'psim' from the 'diag_physics' pool

pv_cell (real) (nVertLevels, nCells, Time)

Units	s^{-1} kg^{-1} m^3
Description	*absolute vorticity/rho_zz averaged to the cell center from the vertices*
Accessed in code	as 'pv_cell' from the 'diag' pool

pv_edge (real) (nVertLevels, nEdges, Time)

Units	s^{-1} kg^{-1} m^3
Description	*absolute vorticity/rho_zz averaged to the cell edge from the vertices*
Accessed in code	as 'pv_edge' from the 'diag' pool

pv_vertex (real) (nVertLevels, nVertices, Time)

Units	s^{-1} kg^{-1} m^3
Description	*absolute vorticity/rho_zz at a vertex*
Accessed in code	as 'pv_vertex' from the 'diag' pool

q2 (real) (nCells, Time)

Units	*kg kg^{-1}*
Description	*2-meter specific humidity*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as 'q2' from the 'diag_physics' pool

qbuoy (real) (nVertLevels, nCells, Time)

Units	*m^2 s^{-2}*
Description	*TKE production - buoyancy*
Allocated by	bl_mynn_in
Accessed in code	as 'qbuoy' from the 'diag_physics' pool

qc — *see 'scalars'*

qc_amb — *see 'scalars_amb'*

qc_cu (real) (nVertLevels, nCells, Time)

Units	*kg kg^{-1}*
Description	*in-cloud cloud water mixing ratio in Grell-Freitas cloud model*
Allocated by	cu_grell_freitas_in
Accessed in code	as 'qc_cu' from the 'diag_physics' pool

qcg (real) (nCells, Time)

Units	*kg kg^{-1}*
Description	*cloud water mixing ratio at the ground surface*
Allocated by	bl_mynn_in
Accessed in code	as 'qcg' from the 'diag_physics' pool

qdiss (real) (nVertLevels, nCells, Time)

Units	*m^2 s^{-2}*
Description	*TKE dissipation*
Allocated by	bl_mynn_in
Accessed in code	as 'qdiss' from the 'diag_physics' pool

qfx (real) (nCells, Time)

Units	*kg m^{-2} s^{-1}*
Description	*upward moisture flux at the surface*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as 'qfx' from the 'diag_physics' pool

qg — *see 'scalars'*

qg_amb — *see 'scalars_amb'*

qgh (real) (nCells, Time)

Units	*kg kg^{-1}*
Description	*lowest level saturation mixing ratio*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as 'qgh' from the 'diag_physics' pool

qi — *see 'scalars'*

qi_amb — *see 'scalars_amb'*

qi_cu (real) (nVertLevels, nCells, Time)

Units	*kg kg^{-1}*
Description	*in-cloud cloud ice mixing ratio in Grell-Freitas cloud model*
Allocated by	cu_grell_freitas_in
Accessed in code	as 'qi_cu' from the 'diag_physics' pool

qke (real) (nVertLevels, nCells, Time)

Units	*m^2 s^{-2}*
Description	*twice turbulent kinetic energy*
Allocated by	bl_mynn_in
Accessed in code	as 'qke' from the 'diag_physics' pool

qke_adv (real) (nVertLevels, nCells, Time)

Units	*m^2 s^{-2}*
Description	*twice turbulent kinetic energy*
Allocated by	bl_mynn_in
Accessed in code	as ‘qke_adv’ from the ‘diag_physics’ pool

qr — *see ‘scalars’*

qr_amb — *see ‘scalars_amb’*

qs — *see ‘scalars’*

qs_amb — *see ‘scalars_amb’*

qsfc (real) (nCells, Time)

Units	*kg kg^{-1}*
Description	*specific humidity at lower boundary*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as ‘qsfc’ from the ‘diag_physics’ pool

qshear (real) (nVertLevels, nCells, Time)

Units	*m^2 s^{-2}*
Description	*TKE production - shear*
Allocated by	bl_mynn_in
Accessed in code	as ‘qshear’ from the ‘diag_physics’ pool

qsq (real) (nVertLevels, nCells, Time)

Units	*kg kg^{-12}*
Description	*liquid water variance*
Allocated by	bl_mynn_in
Accessed in code	as ‘qsq’ from the ‘diag_physics’ pool

qv — *see ‘scalars’*

qv_amb — *see 'scalars_amb'*

qwt (real) (nVertLevels, nCells, Time)

Units	m^2 s^{-2}
Description	*TKE vertical distribution*
Allocated by	bl_mynn_in
Accessed in code	as 'qwt' from the 'diag_physics' pool

rainc (real) (nCells, Time)

Units	*mm*
Description	*accumulated convective precipitation*
Allocated by	cu_grell_freitas_in, cu_kain_fritsch_in, cu_tiedtke_in
Accessed in code	as 'rainc' from the 'diag_physics' pool

raincv (real) (nCells, Time)

Units	*mm*
Description	*time-step convective precipitation*
Allocated by	cu_grell_freitas_in, cu_kain_fritsch_in, cu_tiedtke_in
Accessed in code	as 'raincv' from the 'diag_physics' pool

rainnc (real) (nCells, Time)

Units	*mm*
Description	*accumulated total grid-scale precipitation*
Allocated by	mp_kessler_in, mp_thompson_in, mp_wsm6_in
Accessed in code	as 'rainnc' from the 'diag_physics' pool

rainncv (real) (nCells, Time)

Units	*mm*
Description	*time-step total grid-scale precipitation*
Allocated by	mp_kessler_in, mp_thompson_in, mp_wsm6_in
Accessed in code	as 'rainncv' from the 'diag_physics' pool

rainprod (real) (nVertLevels, nCells, Time)

Units	*s^{-1}*
Description	*rain production rate*
Allocated by	mp_thompson_in
Accessed in code	as 'rainprod' from the 'diag_physics' pool

re_cloud (real) (nVertLevels, nCells, Time)

Units	*m*
Description	*effective radius of cloud water droplets*
Allocated by	mp_thompson_in, mp_wsm6_in
Accessed in code	as 're_cloud' from the 'diag_physics' pool

re_ice (real) (nVertLevels, nCells, Time)

Units	*m*
Description	*effective radius of cloud ice crystals*
Allocated by	mp_thompson_in, mp_wsm6_in
Accessed in code	as 're_ice' from the 'diag_physics' pool

re_snow (real) (nVertLevels, nCells, Time)

Units	*m*
Description	*effective radius of snow crystals*
Allocated by	mp_thompson_in, mp_wsm6_in
Accessed in code	as 're_snow' from the 'diag_physics' pool

refl10cm_1km (real) (nCells, Time)

Units	*dBZ*
Description	*diagnosed 10 cm radar reflectivity at 1 km AGL*
Allocated by	mp_thompson_in, mp_wsm6_in
Accessed in code	as 'refl10cm_1km' from the 'diag_physics' pool

refl10cm_1km_max (real) (nCells, Time)

Units	*dBZ*
Description	*maximum diagnosed 10 cm radar reflectivity at 1 km AGL since last output time*
Allocated by	mp_thompson_in, mp_wsm6_in
Accessed in code	as 'refl10cm_1km_max' from the 'diag_physics' pool

refl10cm_max (real) (nCells, Time)

Units	*dBZ*
Description	*10 cm maximum radar reflectivity*
Allocated by	mp_thompson_in, mp_wsm6_in
Accessed in code	as 'refl10cm_max' from the 'diag_physics' pool

regime (real) (nCells, Time)

Units	*unitless*
Description	*flag indicating the PBL regime (stable,unstable,...)*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as 'regime' from the 'diag_physics' pool

relhum (real) (nVertLevels, nCells, Time)

Units	*percent*
Description	*Relative humidity*
Accessed in code	as 'relhum' from the 'diag' pool

rho (real) (nVertLevels, nCells, Time)

Units	*kg* m^{-3}
Description	*Dry air density*
Accessed in code	as 'rho' from the 'diag' pool

rho_amb (real) (nVertLevels, nCells, Time)

Units	*kg* m^{-3}
Description	*Dry air density increment*
Accessed in code	as 'rho' from the 'tend_iau' pool

rho_base (real) (nVertLevels, nCells, Time)

Units	*kg m^{-3}*
Description	*Base state dry air density*
Accessed in code	as 'rho_base' from the 'diag' pool

rho_edge (real) (nVertLevels, nEdges, Time)

Units	*kg m^{-3}*
Description	*rho_zz averaged from cell centers to the cell edge*
Accessed in code	as 'rho_edge' from the 'diag' pool

rho_p (real) (nVertLevels, nCells, Time)

Units	*kg m^{-3}*
Description	*rho/zz perturbation from the reference state value, advanced over acoustic steps*
Accessed in code	as 'rho_p' from the 'diag' pool

rho_p_save (real) (nVertLevels, nCells, Time)

Units	*kg m^{-3}*
Description	*predicted value rho_p, saved before acoustic steps*
Accessed in code	as 'rho_p_save' from the 'diag' pool

rho_pp (real) (nVertLevels, nCells, Time)

Units	*kg m^{-3}*
Description	*rho/zz perturbation from rho_pp, advanced over acoustic steps*
Accessed in code	as 'rho_pp' from the 'diag' pool

rho_zz (real) (nVertLevels, nCells, Time)

Units	*kg m^{-3}*
Description	*Dry air density divided by d(zeta)/dz*
Accessed in code	as 'rho_zz' from the 'state' pool

rho_zz_old_split (real) (nVertLevels, nCells, Time)

Units	*kg m^{-3}*
Description	*rho/zz*
Accessed in code	as 'rho_zz_old_split' from the 'diag' pool

rmol (real) (nCells, Time)

Units	*unitless*
Description	*1./L Monin Obukhov length*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as 'rmol' from the 'diag_physics' pool

rniblten (real) (nVertLevels, nCells, Time)

Units	*nb kg^{-1} s^{-1}*
Description	*tendency of cloud ice number concentration due to pbl processes*
Allocated by	bl_mynn_in
Accessed in code	as 'rniblten' from the 'tend_physics' pool

rqcblten (real) (nVertLevels, nCells, Time)

Units	*kg kg^{-1} s^{-1}*
Description	*tendency of cloud water mixing ratio due to pbl processes*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as 'rqcblten' from the 'tend_physics' pool

rqccuten (real) (nVertLevels, nCells, Time)

Units	*kg kg^{-1} s^{-1}*
Description	*tendency of cloud water mixing ratio due to cumulus convection*
Allocated by	cu_grell_freitas_in, cu_kain_fritsch_in, cu_tiedtke_in
Accessed in code	as 'rqccuten' from the 'tend_physics' pool

rqiblten (real) (nVertLevels, nCells, Time)

Units	*kg kg^{-1} s^{-1}*
Description	*tendency of cloud ice mixing ratio due to pbl processes*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as 'rqiblten' from the 'tend_physics' pool

rqicuten (real) (nVertLevels, nCells, Time)

Units	*kg kg^{-1} s^{-1}*
Description	*tendency of cloud ice mixing ratio due to cumulus convection*
Allocated by	cu_grell_freitas_in, cu_kain_fritsch_in, cu_tiedtke_in
Accessed in code	as ‘rqicuten’ from the ‘tend_physics’ pool

rqrcuten (real) (nVertLevels, nCells, Time)

Units	*kg kg^{-1} s^{-1}*
Description	*tendency of rain mixing ratio due to cumulus convection*
Allocated by	cu_kain_fritsch_in
Accessed in code	as ‘rqrcuten’ from the ‘tend_physics’ pool

rqscuten (real) (nVertLevels, nCells, Time)

Units	*kg kg^{-1} s^{-1}*
Description	*tendency of snow mixing ratio due to cumulus convection*
Allocated by	cu_kain_fritsch_in
Accessed in code	as ‘rqscuten’ from the ‘tend_physics’ pool

rqvblten (real) (nVertLevels, nCells, Time)

Units	*kg kg^{-1} s^{-1}*
Description	*tendency of water vapor mixing ratio due to pbl processes*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as ‘rqvblten’ from the ‘tend_physics’ pool

rqvcuten (real) (nVertLevels, nCells, Time)

Units	*kg kg^{-1} s^{-1}*
Description	*tendency of water vapor mixing ratio due to cumulus convection*
Allocated by	cu_grell_freitas_in, cu_kain_fritsch_in, cu_tiedtke_in
Accessed in code	as ‘rqvcuten’ from the ‘tend_physics’ pool

rqvdynten (real) (nVertLevels, nCells, Time)

Units	*kg kg^{-1} s^{-1}*
Description	*tendency of water vapor due to horizontal and vertical advections*
Allocated by	cu_grell_freitas_in, cu_tiedtke_in
Accessed in code	as 'rqvdynten' from the 'tend_physics' pool

rre_cloud (real) (nVertLevels, nCells, Time)

Units	*microns*
Description	*effective radius of cloud water droplets calculated in RRTMG radiation*
Accessed in code	as 'rre_cloud' from the 'diag_physics' pool

rre_ice (real) (nVertLevels, nCells, Time)

Units	*microns*
Description	*effective radius of cloud ice crystals calculated in RRTMG radiation*
Accessed in code	as 'rre_ice' from the 'diag_physics' pool

rre_snow (real) (nVertLevels, nCells, Time)

Units	
Description	
Accessed in code	as 'rre_snow' from the 'diag_physics' pool

rt_diabatic_tend (real) (nVertLevels, nCells, Time)

Units	*kg K s^{-1}*
Description	*Tendency of coupled potential temperature from physics*
Accessed in code	as 'rt_diabatic_tend' from the 'tend' pool

rthblten (real) (nVertLevels, nCells, Time)

Units	*K s^{-1}*
Description	*tendency of potential temperature due to pbl processes*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as 'rthblten' from the 'tend_physics' pool

rthcuten (real) (nVertLevels, nCells, Time)

Units	*K s^{-1}*
Description	*tendency of potential temperature due to cumulus convection*
Allocated by	cu_grell_freitas_in, cu_kain_fritsch_in, cu_tiedtke_in
Accessed in code	as ‘rthcuten’ from the ‘tend_physics’ pool

rthdynten (real) (nVertLevels, nCells, Time)

Units	*K s^{-1}*
Description	*tendency of temperature due to horizontal and vertical advections*
Allocated by	cu_grell_freitas_in, cu_tiedtke_in
Accessed in code	as ‘rthdynten’ from the ‘tend_physics’ pool

rtheta_base (real) (nVertLevels, nCells, Time)

Units	*kg K m^{-3}*
Description	*reference state rho*theta/zz*
Accessed in code	as ‘rtheta_base’ from the ‘diag’ pool

rtheta_p (real) (nVertLevels, nCells, Time)

Units	*kg K m^{-3}*
Description	*rho*theta_m/zz perturbation from the reference state value*
Accessed in code	as ‘rtheta_p’ from the ‘diag’ pool

rtheta_p_save (real) (nVertLevels, nCells, Time)

Units	*kg K m^{-3}*
Description	*predicted value rtheta_p, saved before acoustic steps*
Accessed in code	as ‘rtheta_p_save’ from the ‘diag’ pool

rtheta_pp (real) (nVertLevels, nCells, Time)

Units	*kg K m^{-3}*
Description	*rho*theta_m/zz perturbation from rtheta_p*
Accessed in code	as ‘rtheta_pp’ from the ‘diag’ pool

rtheta_pp_old (real) (nVertLevels, nCells, Time)

Units	*kg K m^{-3}*
Description	*old time level values of rho*theta_m/zz perturbation from rtheta_p, used in 3D divergence damping*
Accessed in code	as 'rtheta_pp_old' from the 'diag' pool

rthratenlw (real) (nVertLevels, nCells, Time)

Units	*K s^{-1}*
Description	*tendency of potential temperature due to short wave radiation*
Accessed in code	as 'rthratenlw' from the 'tend_physics' pool

rthratensw (real) (nVertLevels, nCells, Time)

Units	*K s^{-1}*
Description	*tendency of potential temperature due to short wave radiation*
Accessed in code	as 'rthratensw' from the 'tend_physics' pool

ru (real) (nVertLevels, nEdges, Time)

Units	*kg m^{-2} s^{-1}*
Description	*horizontal momentum at cell edge (rho*u/zz)*
Accessed in code	as 'ru' from the 'diag' pool

ruAvg (real) (nVertLevels, nEdges, Time)

Units	*kg m^{-2} s^{-1}*
Description	*time-averaged rho*u/zz used in scalar transport*
Accessed in code	as 'ruAvg' from the 'diag' pool

ruAvg_split (real) (nVertLevels, nEdges, Time)

Units	*kg m^{-2} s^{-1}*
Description	*time-averaged rho*u/zz used in scalar transport*
Accessed in code	as 'ruAvg_split' from the 'diag' pool

ru_p (real) (nVertLevels, nEdges, Time)

Units	*kg m^{-2} s^{-1}*
Description	*acoustic perturbation horizontal momentum at cell edge (rho*u/zz)*
Accessed in code	as 'ru_p' from the 'diag' pool

ru_save (real) (nVertLevels, nEdges, Time)

Units	*kg m^{-2} s^{-1}*
Description	*predicted value of horizontal momentum, saved before acoustic steps*
Accessed in code	as 'ru_save' from the 'diag' pool

rubldiff (real) (nVertLevels, nCells, Time)

Units	*m s^{-2}*
Description	*change in PBL zonal wind tendency due to gravity wave drag over orography*
Accessed in code	as 'rubldiff' from the 'diag_physics' pool

rublten (real) (nVertLevels, nCells, Time)

Units	*m s^{-1} s^{-1}*
Description	*tendency of zonal wind due to pbl processes*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as 'rublten' from the 'tend_physics' pool

rublten_Edge (real) (nVertLevels, nEdges, Time)

Units	
Description	
Accessed in code	as 'rublten_Edge' from the 'tend_physics' pool

rucuten (real) (nVertLevels, nCells, Time)

Units	*m s^{-1} s^{-1}*
Description	*tendency of zonal wind due to cumulus convection*
Allocated by	cu_grell_freitas_in, cu_tiedtke_in
Accessed in code	as 'rucuten' from the 'tend_physics' pool

rucuten_Edge (real) (nVertLevels, nEdges, Time)

Units	
Description	
Accessed in code	as ‘rucuten_Edge’ from the ‘tend_physics’ pool

rv (real) (nVertLevels, nEdges, Time)

Units	*???*
Description	*???*
Accessed in code	as ‘rv’ from the ‘diag’ pool

rvbldiff (real) (nVertLevels, nCells, Time)

Units	*m s^{-2}*
Description	*change in PBL meridional wind tendency due to gravity wave drag over orography*
Accessed in code	as ‘rvbldiff’ from the ‘diag_physics’ pool

rvblten (real) (nVertLevels, nCells, Time)

Units	*m s^{-1} s^{-1}*
Description	*tendency of meridional wind due to pbl processes*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as ‘rvblten’ from the ‘tend_physics’ pool

rvcuten (real) (nVertLevels, nCells, Time)

Units	*m s^{-1} s^{-1}*
Description	*tendency of meridional wind due to cumulus convection*
Allocated by	cu_grell_freitas_in, cu_tiedtke_in
Accessed in code	as ‘rvcuten’ from the ‘tend_physics’ pool

rw (real) (nVertLevelsP1, nCells, Time)

Units	*kg m^{-2} s^{-1}*
Description	*rho*omega/zz carried at w points*
Accessed in code	as ‘rw’ from the ‘diag’ pool

rw_p (real) (nVertLevelsP1, nCells, Time)

Units	*kg m^{-2} s^{-1}*
Description	*acoustic perturbation rho*omega/zz carried at w points*
Accessed in code	as 'rw_p' from the 'diag' pool

rw_save (real) (nVertLevelsP1, nCells, Time)

Units	*kg m^{-2} s^{-1}*
Description	*predicted value of rho*omega/zz, saved before acoustic steps*
Accessed in code	as 'rw_save' from the 'diag' pool

scalars (real) (nVertLevels, nCells, Time)

Description	*'scalars' is a variable array used in code which is made up of the constituent fields listed below.*
Accessed in code	as 'scalars' from the 'state' pool

Contains constituents:

qv — *constituent field of var_array **scalars***

Description	*Water vapor mixing ratio*
Units	*kg kg^{-1}*
Array Group	moist

qc — *constituent field of var_array **scalars***

Description	*Cloud water mixing ratio*
Units	*kg kg^{-1}*
Array Group	moist
Allocated by	bl_mynn_in, bl_ysu_in, cu_tiedtke_in, mp_kessler_in, mp_thompson_in, mp_wsm6_in

qr — *constituent field of var_array **scalars***

Description	*Rain water mixing ratio*
Units	*kg kg^{-1}*
Array Group	moist
Allocated by	mp_kessler_in, mp_thompson_in, mp_wsm6_in

qi — *constituent field of var_array **scalars***

Description	*Ice mixing ratio*
Units	*kg kg*$^{-1}$
Array Group	moist
Allocated by	bl_mynn_in, bl_ysu_in, cu_tiedtke_in, mp_thompson_in, mp_wsm6_in

qs — *constituent field of var_array **scalars***

Description	*Snow mixing ratio*
Units	*kg kg*$^{-1}$
Array Group	moist
Allocated by	mp_thompson_in, mp_wsm6_in

qg — *constituent field of var_array **scalars***

Description	*Graupel mixing ratio*
Units	*kg kg*$^{-1}$
Array Group	moist
Allocated by	mp_thompson_in, mp_wsm6_in

ni — *constituent field of var_array **scalars***

Description	*Cloud ice number concentration*
Units	*nb kg*$^{-1}$
Array Group	number
Allocated by	bl_mynn_in, mp_thompson_in

nr — *constituent field of var_array **scalars***

Description	*Rain number concentration*
Units	*nb kg*$^{-1}$
Array Group	number
Allocated by	mp_thompson_in

scalars_amb (real) (nVertLevels, nCells, Time)

Description	*'scalars_amb' is a variable array used in code which is made up of the constituent fields listed below.*
Accessed in code	as 'scalars' from the 'tend_iau' pool

Contains constituents:

qv_amb — *constituent field of var_array* ***scalars_amb***

Description	*Water vapor mixing ratio increment*
Units	*kg kg*$^{-1}$
Array Group	moist

qc_amb — *constituent field of var_array* ***scalars_amb***

Description	*Cloud water mixing ratio increment*
Units	*kg kg*$^{-1}$
Array Group	moist
Allocated by	bl_mynn_in, bl_ysu_in, cu_tiedtke_in, mp_kessler_in, mp_thompson_in, mp_wsm6_in

qr_amb — *constituent field of var_array* ***scalars_amb***

Description	*Rain water mixing ratio increment*
Units	*kg kg*$^{-1}$
Array Group	moist
Allocated by	mp_kessler_in, mp_thompson_in, mp_wsm6_in

qi_amb — *constituent field of var_array* ***scalars_amb***

Description	*Ice mixing ratio increment*
Units	*kg kg*$^{-1}$
Array Group	moist
Allocated by	bl_mynn_in, bl_ysu_in, cu_tiedtke_in, mp_thompson_in, mp_wsm6_in

qs_amb — *constituent field of var_array* ***scalars_amb***

Description	*Snow mixing ratio increment*
Units	*kg kg*$^{-1}$
Array Group	moist
Allocated by	mp_thompson_in, mp_wsm6_in

qg_amb — *constituent field of var_array **scalars_amb***

Description	*Graupel mixing ratio increment*
Units	*kg kg^{-1}*
Array Group	moist
Allocated by	mp_thompson_in, mp_wsm6_in

scalars_tend (real) (nVertLevels, nCells, Time)

Description	*'scalars_tend' is a variable array used in code which is made up of the constituent fields listed below.*
Accessed in code	as 'scalars_tend' from the 'tend' pool

Contains constituents:

tend_qv — *constituent field of var_array **scalars_tend***

Description	*Tendency of water vapor mixing ratio*
Units	*kg kg^{-1} s^{-1}*
Array Group	moist

tend_qc — *constituent field of var_array **scalars_tend***

Description	*Tendency of cloud water mixing ratio*
Units	*kg kg^{-1} s^{-1}*
Array Group	moist
Allocated by	bl_mynn_in, bl_ysu_in, cu_tiedtke_in, mp_kessler_in, mp_thompson_in, mp_wsm6_in

tend_qr — *constituent field of var_array **scalars_tend***

Description	*Tendency of rain water mixing ratio*
Units	*kg kg^{-1} s^{-1}*
Array Group	moist
Allocated by	mp_kessler_in, mp_thompson_in, mp_wsm6_in

tend_qi — *constituent field of var_array* ***scalars_tend***

Description	*Tendency of ice mixing ratio*
Units	*kg kg^{-1} s^{-1}*
Array Group	moist
Allocated by	bl_mynn_in, bl_ysu_in, cu_tiedtke_in, mp_thompson_in, mp_wsm6_in

tend_qs — *constituent field of var_array* ***scalars_tend***

Description	*Tendency of snow ratio*
Units	*kg kg^{-1} s^{-1}*
Array Group	moist
Allocated by	mp_thompson_in, mp_wsm6_in

tend_qg — *constituent field of var_array* ***scalars_tend***

Description	*Tendency of graupel mixing ratio*
Units	*kg kg^{-1} s^{-1}*
Array Group	moist
Allocated by	mp_thompson_in, mp_wsm6_in

tend_ni — *constituent field of var_array* ***scalars_tend***

Description	*Tendency of cloud ice number concentration*
Units	*nb kg^{-1} s^{-1}*
Array Group	number
Allocated by	bl_mynn_in, mp_thompson_in

tend_nr — *constituent field of var_array* ***scalars_tend***

Description	*Tendency of rain number concentration*
Units	*nb kg^{-1} s^{-1}*
Array Group	number
Allocated by	mp_thompson_in

seaice (real) (nCells, Time)

Units	*unitless*
Description	*sea-ice flag (0=no seaice; =1 otherwise)*
Accessed in code	as ‘seaice’ from the ‘sfc_input’ pool

sfc_albbck (real) (nCells, Time)

Units	*unitless*
Description	*background surface albedo*
Accessed in code	as 'sfc_albbck' from the 'sfc_input' pool

sfc_albedo (real) (nCells, Time)

Units	*unitless*
Description	*surface albedo*
Accessed in code	as 'sfc_albedo' from the 'diag_physics' pool

sfc_emibck (real) (nCells, Time)

Units	*unitless*
Description	*background surface emissivity*
Accessed in code	as 'sfc_emibck' from the 'diag_physics' pool

sfc_emiss (real) (nCells, Time)

Units	*unitless*
Description	*surface emissivity*
Accessed in code	as 'sfc_emiss' from the 'diag_physics' pool

sfcrunoff (real) (nCells, Time)

Units	*mm*
Description	*surface runoff*
Accessed in code	as 'sfcrunoff' from the 'diag_physics' pool

sh2o (real) (nSoilLevels, nCells, Time)

Units	$m3\ m^{-3}$
Description	*soil equivalent liquid water*
Accessed in code	as 'sh2o' from the 'sfc_input' pool

sh3d (real) (nVertLevels, nCells, Time)

Units	*unitless*
Description	*stability function for heat*
Allocated by	bl_mynn_in
Accessed in code	as ‘sh3d’ from the ‘diag_physics’ pool

shdmax (real) (nCells)

Units	*unitless*
Description	*maximum fractional coverage of annual green vegetation fraction*
Accessed in code	as ‘shdmax’ from the ‘sfc_input’ pool

shdmin (real) (nCells)

Units	*unitless*
Description	*minimum fractional coverage of annual green vegetation fraction*
Accessed in code	as ‘shdmin’ from the ‘sfc_input’ pool

skintemp (real) (nCells, Time)

Units	*K*
Description	*ground or water surface temperature*
Accessed in code	as ‘skintemp’ from the ‘sfc_input’ pool

smcrel (real) (nSoilLevels, nCells, Time)

Units	*m3 m*$^{-3}$
Description	*soil moisture threshold below which transpiration begins to stress*
Accessed in code	as ‘smcrel’ from the ‘sfc_input’ pool

smois (real) (nSoilLevels, nCells, Time)

Units	*m3 m*$^{-3}$
Description	*soil moisture*
Accessed in code	as ‘smois’ from the ‘sfc_input’ pool

smstav (real) (nCells, Time)

Units	*unitless*
Description	*surface moisture availability*
Accessed in code	as ‘smstav’ from the ‘diag_physics’ pool

smstot (real) (nCells, Time)

Units	$m^3\ m^{-3}$
Description	*total soil mositure*
Accessed in code	as ‘smstot’ from the ‘diag_physics’ pool

snoalb (real) (nCells)

Units	*unitless*
Description	*annual maximum snow albedo*
Accessed in code	as ‘snoalb’ from the ‘sfc_input’ pool

snopcx (real) (nCells, Time)

Units	$W\ m^{-2}$
Description	*snow phase change heat flux*
Accessed in code	as ‘snopcx’ from the ‘diag_physics’ pool

snotime (real) (nCells, Time)

Units	*unitless*
Description	*initial number of time-steps since last snow fall*
Accessed in code	as ‘snotime’ from the ‘diag_physics’ pool

snow (real) (nCells, Time)

Units	$kg\ m^{-2}$
Description	*snow water equivalent*
Accessed in code	as ‘snow’ from the ‘sfc_input’ pool

snowc (real) (nCells, Time)

Units	*unitless*
Description	*flag for snow on ground (=0 no snow; =1,otherwise*
Accessed in code	as ‘snowc’ from the ‘sfc_input’ pool

snowh (real) (nCells, Time)

Units	*m*
Description	*physical snow depth*
Accessed in code	as 'snowh' from the 'sfc_input' pool

snownc (real) (nCells, Time)

Units	*mm*
Description	*accumulated grid-scale precipitation of snow*
Allocated by	mp_thompson_in, mp_wsm6_in
Accessed in code	as 'snownc' from the 'diag_physics' pool

snowncv (real) (nCells, Time)

Units	*mm*
Description	*time-step grid-scale precipitation of snow*
Allocated by	mp_thompson_in, mp_wsm6_in
Accessed in code	as 'snowncv' from the 'diag_physics' pool

sr (real) (nCells, Time)

Units	*unitless*
Description	*time-step ratio of frozen versus total grid-scale precipitation*
Accessed in code	as 'sr' from the 'diag_physics' pool

sslt — *see 'aerosols'*

sst (real) (nCells, Time)

Units	*K*
Description	*sea-surface temperature*
Accessed in code	as 'sst' from the 'sfc_input' pool

sstsk (real) (nCells, Time)

Units	*K*
Description	*skin sea-aurface temperature*
Accessed in code	as 'sstsk' from the 'diag_physics' pool

sstsk_dtc (real) (nCells, Time)

Units	*K*
Description	*skin sea-surface temperature cooling*
Accessed in code	as ‘sstsk_dtc’ from the ‘diag_physics’ pool

sstsk_dtw (real) (nCells, Time)

Units	*K*
Description	*skin sea-surface temperature warming*
Accessed in code	as ‘sstsk_dtw’ from the ‘diag_physics’ pool

sul — *see ‘aerosols’*

surface_pressure (real) (nCells, Time)

Units	*Pa*
Description	*Diagnosed surface pressure*
Accessed in code	as ‘surface_pressure’ from the ‘diag’ pool

swcf (real) (nCells, Time)

Units	*W m*$^{-2}$
Description	*top-of-atmosphere cloud shortwave radiative forcing*
Accessed in code	as ‘swcf’ from the ‘diag_physics’ pool

swdnb (real) (nCells, Time)

Units	*W m*$^{-2}$
Description	*all-sky downward surface shortwave radiation flux*
Accessed in code	as ‘swdnb’ from the ‘diag_physics’ pool

swdnbc (real) (nCells, Time)

Units	*W m*$^{-2}$
Description	*clear-sky downward surface shortwave radiation flux*
Accessed in code	as ‘swdnbc’ from the ‘diag_physics’ pool

swdnflx (real) (nVertLevelsP2, nCells, Time)

Units	-
Description	-
Accessed in code	as ‘swdnflx’ from the ‘diag_physics’ pool

swdnflxc (real) (nVertLevelsP2, nCells, Time)

Units	-
Description	-
Accessed in code	as ‘swdnflxc’ from the ‘diag_physics’ pool

swdnt (real) (nCells, Time)

Units	*W m*$^{-2}$
Description	*all-sky downward top-of-atmosphere shortwave radiation flux*
Accessed in code	as ‘swdnt’ from the ‘diag_physics’ pool

swdntc (real) (nCells, Time)

Units	*W m*$^{-2}$
Description	*clear-sky downward top-of-atmosphere shortwave radiation flux*
Accessed in code	as ‘swdntc’ from the ‘diag_physics’ pool

swupb (real) (nCells, Time)

Units	*W m*$^{-2}$
Description	*all-sky upward surface shortwave radiation flux*
Accessed in code	as ‘swupb’ from the ‘diag_physics’ pool

swupbc (real) (nCells, Time)

Units	*W m*$^{-2}$
Description	*clear-sky upward surface shortwave radiation flux*
Accessed in code	as ‘swupbc’ from the ‘diag_physics’ pool

swupflx (real) (nVertLevelsP2, nCells, Time)

Units	-
Description	-
Accessed in code	as ‘swupflx’ from the ‘diag_physics’ pool

swupflxc (real) (nVertLevelsP2, nCells, Time)

Units	-
Description	-
Accessed in code	as 'swupflxc' from the 'diag_physics' pool

swupt (real) (nCells, Time)

Units	*W m^{-2}*
Description	*all-sky upward top-of-atmosphere shortwave radiation flux*
Accessed in code	as 'swupt' from the 'diag_physics' pool

swuptc (real) (nCells, Time)

Units	*W m^{-2}*
Description	*clear-sky upward top-of-atmosphere shortwave radiation flux*
Accessed in code	as 'swuptc' from the 'diag_physics' pool

t2m (real) (nCells, Time)

Units	*K*
Description	*2-meter temperature*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as 't2m' from the 'diag_physics' pool

t_oml (real) (nCells, Time)

Units	*K*
Description	*ocean mixed layer temperature*
Accessed in code	as 't_oml' from the 'diag_physics' pool

t_oml_200m_initial (real) (nCells, Time)

Units	*K*
Description	*ocean mixed layer 200 m mean temperature at initial time*
Accessed in code	as 't_oml_200m_initial' from the 'diag_physics' pool

t_oml_initial (real) (nCells, Time)

Units	*K*
Description	*ocean mixed layer temperature at initial time*
Accessed in code	as 't_oml_initial' from the 'diag_physics' pool

tday_accum (real) (nCells, Time)

Units	*K*
Description	*accumulated daily surface temperature for current day*
Accessed in code	as 'tday_accum' from the 'diag_physics' pool

tend_ni — *see 'scalars_tend'*

tend_nr — *see 'scalars_tend'*

tend_qc — *see 'scalars_tend'*

tend_qg — *see 'scalars_tend'*

tend_qi — *see 'scalars_tend'*

tend_qr — *see 'scalars_tend'*

tend_qs — *see 'scalars_tend'*

tend_qv — *see 'scalars_tend'*

tend_rho (real) (nVertLevels, nCells, Time)

Units	$kg\ m^{-3}\ s^{-1}$
Description	*Tendency of dry density from dynamics*
Accessed in code	as 'rho_zz' from the 'tend' pool

tend_rtheta_adv (real) (nVertLevels, nCells, Time)

Units	*kg K m^{-3} s^{-1}*
Description	*flux divergence for rho*theta_m/zz, used in the Tiedtke convective parameterization*
Accessed in code	as ‘tend_rtheta_adv’ from the ‘diag’ pool

tend_sfc_pressure (real) (nCells, Time)

Units	*Pa s^{-1}*
Description	*Tendency of surface pressure*
Accessed in code	as ‘tend_sfc_pressure’ from the ‘tend’ pool

tend_theta (real) (nVertLevels, nCells, Time)

Units	*kg K m^{-3} s^{-1}*
Description	*tendency of coupled potential temperature rho*theta_m/zz from dynamics and physics, updated each RK step*
Accessed in code	as ‘theta_m’ from the ‘tend’ pool

tend_u (real) (nVertLevels, nEdges, Time)

Units	*m s^{-2}*
Description	*Tendency of u from dynamics*
Accessed in code	as ‘u’ from the ‘tend’ pool

tend_w (real) (nVertLevelsP1, nCells, Time)

Units	*m s^{-2}*
Description	*Tendency of w from dynamics*
Accessed in code	as ‘w’ from the ‘tend’ pool

tend_w_buoy (real) (nVertLevelsP1, nCells, Time)

Units	*m s^{-2}*
Description	*Tendency of w due to buoyancy force*
Accessed in code	as ‘w_buoy’ from the ‘tend’ pool

tend_w_pgf (real) (nVertLevelsP1, nCells, Time)

Units	*m s^{-2}*
Description	*Tendency of w due to pressure gradient force*
Accessed in code	as 'w_pgf' from the 'tend' pool

ter (real) (nCells)

Units	*m*
Description	*terrain height*
Accessed in code	as 'ter' from the 'sfc_input' pool

th2m (real) (nCells, Time)

Units	*K*
Description	*2-meter potential temperature*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as 'th2m' from the 'diag_physics' pool

thc (real) (nCells, Time)

Units	*Cal cm^{-2} K^{-1} s$^{-0.5}$*
Description	*thermal inertia*
Accessed in code	as 'thc' from the 'diag_physics' pool

theta (real) (nVertLevels, nCells, Time)

Units	*K*
Description	*Potential temperature*
Accessed in code	as 'theta' from the 'diag' pool

theta_amb (real) (nVertLevels, nCells, Time)

Units	*K*
Description	*Potential temperature increment*
Accessed in code	as 'theta' from the 'tend_iau' pool

theta_base (real) (nVertLevels, nCells, Time)

Units	*K*
Description	*Base state potential temperature*
Accessed in code	as ‘theta_base’ from the ‘diag’ pool

theta_m (real) (nVertLevels, nCells, Time)

Units	*K*
Description	*Moist potential temperature: theta*(1+q_v*R_v/R_d)*
Accessed in code	as ‘theta_m’ from the ‘state’ pool

tke_pbl (real) (nVertLevels, nCells, Time)

Units	m^2 s^{-2}
Description	*turbulent kinetic energy from PBL*
Allocated by	bl_mynn_in
Accessed in code	as ‘tke_pbl’ from the ‘diag_physics’ pool

tlag (real) (nLags, nCells, Time)

Units	*K*
Description	*daily mean surface temperature of prior days*
Accessed in code	as ‘tlag’ from the ‘diag_physics’ pool

tmn (real) (nCells, Time)

Units	*K*
Description	*deep soil temperature*
Accessed in code	as ‘tmn’ from the ‘sfc_input’ pool

tslb (real) (nSoilLevels, nCells, Time)

Units	*K*
Description	*soil layer temperature*
Accessed in code	as ‘tslb’ from the ‘sfc_input’ pool

tsq (real) (nVertLevels, nCells, Time)

Units	K^2
Description	*liquid water potential temperature variance*
Allocated by	bl_mynn_in
Accessed in code	as 'tsq' from the 'diag_physics' pool

tyear_accum (real) (nCells, Time)

Units	K
Description	*accumulated yearly surface temperature for current year*
Accessed in code	as 'tyear_accum' from the 'diag_physics' pool

tyear_mean (real) (nCells, Time)

Units	K
Description	*annual mean surface temperature*
Accessed in code	as 'tyear_mean' from the 'diag_physics' pool

u (real) (nVertLevels, nEdges, Time)

Units	$m\ s^{-1}$
Description	*Horizontal normal velocity at edges*
Accessed in code	as 'u' from the 'state' pool

u10 (real) (nCells, Time)

Units	$m\ s^{-1}$
Description	*10-meter zonal wind*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as 'u10' from the 'diag_physics' pool

uReconstructMeridional (real) (nVertLevels, nCells, Time)

Units	$m\ s^{-1}$
Description	*Meridional component of reconstructed horizontal velocity at cell centers*
Accessed in code	as 'uReconstructMeridional' from the 'diag' pool

uReconstructX (real) (nVertLevels, nCells, Time)

Units	*m s*$^{-1}$
Description	*Cartesian x-component of reconstructed horizontal velocity at cell centers*
Accessed in code	as 'uReconstructX' from the 'diag' pool

uReconstructY (real) (nVertLevels, nCells, Time)

Units	*m s*$^{-1}$
Description	*Cartesian y-component of reconstructed horizontal velocity at cell centers*
Accessed in code	as 'uReconstructY' from the 'diag' pool

uReconstructZ (real) (nVertLevels, nCells, Time)

Units	*m s*$^{-1}$
Description	*Cartesian z-component of reconstructed horizontal velocity at cell centers*
Accessed in code	as 'uReconstructZ' from the 'diag' pool

uReconstructZonal (real) (nVertLevels, nCells, Time)

Units	*m s*$^{-1}$
Description	*Zonal component of reconstructed horizontal velocity at cell centers*
Accessed in code	as 'uReconstructZonal' from the 'diag' pool

u_amb (real) (nVertLevels, nEdges, Time)

Units	*m s*$^{-1}$
Description	*Horizontal normal velocity increment*
Accessed in code	as 'u' from the 'tend_iau' pool

udrunoff (real) (nCells, Time)

Units	*mm*
Description	*underground runoff*
Accessed in code	as 'udrunoff' from the 'diag_physics' pool

ust (real) (nCells, Time)

Units	*m s^{-1}*
Description	*U* in similarity theory*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as ‘ust’ from the ‘diag_physics’ pool

ustm (real) (nCells, Time)

Units	*m s^{-1}*
Description	*U* in similarity theory without vconv*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as ‘ustm’ from the ‘diag_physics’ pool

v (real) (nVertLevels, nEdges, Time)

Units	*m s^{-1}*
Description	*Horizontal tangential velocity at edges*
Accessed in code	as ‘v’ from the ‘diag’ pool

v10 (real) (nCells, Time)

Units	*m s^{-1}*
Description	*10-meter meridional wind*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as ‘v10’ from the ‘diag_physics’ pool

var2d (real) (nCells)

Units	*m^{2}*
Description	*variance of orography*
Accessed in code	as ‘var2d’ from the ‘sfc_input’ pool

vegfra (real) (nCells, Time)

Units	*unitless*
Description	*vegetation fraction*
Accessed in code	as ‘vegfra’ from the ‘sfc_input’ pool

volc — *see ‘aerosols’*

vorticity (real) (nVertLevels, nVertices, Time)

Units	*s^{-1}*
Description	*Relative vorticity at vertices*
Accessed in code	as ‘vorticity’ from the ‘diag’ pool

w (real) (nVertLevelsP1, nCells, Time)

Units	*$m\ s^{-1}$*
Description	*Vertical velocity at vertical cell faces*
Accessed in code	as ‘w’ from the ‘state’ pool

w0avg (real) (nVertLevels, nCells, Time)

Units	*$m\ s^{-1}$*
Description	*time running-averaged vertical velocity*
Allocated by	cu_kain_fritsch_in
Accessed in code	as ‘w0avg’ from the ‘diag_physics’ pool

wspd (real) (nCells, Time)

Units	*$m\ s^{-1}$*
Description	*wind speed at lowest model level*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as ‘wspd’ from the ‘diag_physics’ pool

wstar (real) (nCells, Time)

Units	*$m\ s^{-1}$*
Description	*mixed velocity scale from PBL scheme*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as ‘wstar’ from the ‘diag_physics’ pool

wwAvg (real) (nVertLevelsP1, nCells, Time)

Units	*$kg\ m^{-2}\ s^{-1}$*
Description	*time-averaged rho*omega/zz used in scalar transport*
Accessed in code	as ‘wwAvg’ from the ‘diag’ pool

wwAvg_split (real) (nVertLevelsP1, nCells, Time)

Units	*kg m^{-2} s^{-1}*
Description	*time-averaged rho*omega/zz used in scalar transport*
Accessed in code	as 'wwAvg_split' from the 'diag' pool

xice (real) (nCells, Time)

Units	*unitless*
Description	*fractional area coverage of sea-ice*
Accessed in code	as 'xice' from the 'sfc_input' pool

xicem (real) (nCells, Time)

Units	*unitless*
Description	*sea-ice flag from previsous time-step*
Accessed in code	as 'xicem' from the 'diag_physics' pool

xland (real) (nCells, Time)

Units	*unitless*
Description	*land-ocean mask (1=land including sea-ice ; 2=ocean)*
Accessed in code	as 'xland' from the 'sfc_input' pool

xmb_shallow (real) (nCells, Time)

Units	*kg m^{-2} s^{-1}*
Description	*cloud base mass flux for shallow convection*
Allocated by	cu_grell_freitas_in
Accessed in code	as 'xmb_shallow' from the 'diag_physics' pool

xmb_total (real) (nCells, Time)

Units	*kg m^{-2} s^{-1}*
Description	*cloud-base mass flux*
Allocated by	cu_grell_freitas_in
Accessed in code	as 'xmb_total' from the 'diag_physics' pool

xtime (text) (Time)

Units	*YYYY-MM-DD_hh:mm:ss*
Description	*Model valid time*
Accessed in code	as ‘xtime’ from the ‘state’ pool

z0 (real) (nCells, Time)

Units	*m*
Description	*roughness height*
Accessed in code	as ‘z0’ from the ‘diag_physics’ pool

znt (real) (nCells, Time)

Units	*m*
Description	*roughness length*
Accessed in code	as ‘znt’ from the ‘diag_physics’ pool

zol (real) (nCells, Time)

Units	*unitless*
Description	*z/L height over Monin-Obukhov length*
Allocated by	bl_mynn_in, bl_ysu_in
Accessed in code	as ‘zol’ from the ‘diag_physics’ pool

zs (real) (nCells, Time)

Units	*m*
Description	*depth of centers of soil layers*
Accessed in code	as ‘zs’ from the ‘diag_physics’ pool

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:

Appendix F

Revision History

1 August 2017

- Update documentation in Chapter 3 for building with PIO 2.x versions
- Correct other minor typographical errors and unclear wording

12 May 2017

- Add documentation for new config_extrap_airtemp initialization namelist option
- Remove documentation for config_smdiv_p_forward model option, which was removed in v5.1
- Add note that config_len_disp is also used by 3-d divergence damping in v5.1
- Change default value of config_smdiv model option to 0.1, reflecting changes in v5.1
- Other minor changes to match the MPAS-Atmosphere v5.1 release

23 December 2016

- Added description of the new 'convection_permitting' physics suite, as well as a list of all possible physics parameterizations, to Chapter 6.
- Updated the chapter on building MPAS to mention specific versions of PIO that are known to work with MPAS, information on the method used to build single-precision executables in v5.0, and a mention of the *experimental* OpenMP capability in v5.0.
- Appendices A, B, and D are now automatically generated based on XML Registry files.
- Many other small changes to reflect the state of the v5.0 release.
- Minor wording changes and corrections of typographical errors throughout document.

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.