# Design Document for the Simple Cloud-Resolving E3SM Atmosphere Model

Peter Caldwell[1], Andy Salinger[2], Luca Bertagna[2], Hassan Beydoun[1], Peter Bogenschutz[1], Andrew Bradley[2], Conrad Clevenger[5], Aaron Donahue[1], Jim Foucar[2], Chris Golaz[1], Oksana Guba[2], Ben Hillman[2], Noel Keen[3], Wuyin Lin[4], Balwinder Singh[5], Andrew Steyer[2], Mark Taylor[2], Chris Terai[1], and Paul Ullrich[6]

[1]Lawrence Livermore National Lab, Livermore CA
[2]Sandia National Laboratories, Albuquerque, NM
[3]Lawrence Berkeley National Laboratory, Berkeley, CA
[4]Brookhaven National Laboratory, Upton, NY
[5]Pacific Northwest National Laboratory, Richland, WA
[6]University of California, Davis, Davis, CA

February 25, 2022

## Contents

# 1 Overview

Put description of model here as well as summary of what will/won't be in this document. Mention that there will be a separate user guide. Also, this doc isn't done until the list of authors is updated. Folks who joined after day 1 aren't included.

# 2 Atmosphere Process Coupler

## 2.1 Introduction

This document details the development of the atmospheric model driver infrastructure for the Simplified Cloud Resolving E3SM Atmosphere Model (SCREAM) project. The atmospheric driver for SCREAM will replace the Energy Exascale Earth System atmospheric model (EAM) driver. The new Atmospheric Driver (AD) will incorporate many of the same features as the current EAM driver, but will simplify and clean up much of the code base and will be written in a modern C++ framework.

Taking advantage of the modern C++ architecture, the SCREAM–AD will introduce an atmospheric process class which will establish a simple and straightforward interface between the driver and individual atmospheric processes. The passing of variables between processes will be greatly improved. Model developers have often complained about the opaque structure of the current physics buffer (PBUF) approach particularly with the inability to properly audit which variables are being accessed and when. In the SCREAM–AD, the PBUF will be replaced with an improved Field Manager (FM) class which will address many of these concerns.

This document is organized as follows: section 2.2 explains how a process will be implemented within the atm_process class. This will be followed by a description of how processes will be coupled together in section 2.3. Section 2.4 will provide a description of how the Field Manager allows variables to be communicated between processes. Specification of runtime options and input/output will then be briefly described in 2.5 and 2.6, respectively, before conclusions in section 2.7.

## 2.2 Within an Atmospheric Process

An atmospheric process is defined in SCREAM as any process that can change the model state. This is in contrast to EAM which treats physics parameterizations and fluid dynamics very differently. Handling fluid dynamics and physics parameterizations identically simplifies the AD and provides greater flexibility in the coupling infrastructure between processes (as described in section 2.3). All atmospheric processes in SCREAM will be instances of an atmospheric process class, atm_process. The atmospheric process class will have three basic functions: initialization, run and finalization. Developers of each parameterization are expected to supply implementations of the three functions customized for their scheme:

- Initialization (called during the initialization stage of the AD) will:
  - initialize all input/output (IO),
  - register all process specific variables with the field manager (FM), see section 2.4,
  - allocate all local arrays and pointers.
- Run (called each timestep during the run stage of the AD) will:
  - retrieve variables from FM to be passed to the main process.
  - conversion from the AD data structure to the data structure used within the parameterization of interest, if applicable.
  - call the main process routine,
  - postprocess output to save updated variables to the FM and to stage output for writing to NetCDF or ADIOS format
- Finalization (called during the finalization stage of the AD) will handle deallocation of all local arrays and pointers.

### 2.2.1 Making Code Portable

To make porting process representations to other models and running processes as standalone executables easy, the initialization, run, and finalize functions of each process will be broken into interface and process tasks. The interface code will grab needed input for the process from the FM and will ensure data is provided to the process calculation in the right format

with the right units. It will then call the actual process code, which will do the actual calculation. After the process code completes, the interface code will convert output to the format required by the AD and/or FM. Because all dependencies on SCREAM-specific code are contained within the interface level, the process code will be easy to use outside of SCREAM. Tasks handled by the interface layer will range from simple tasks like obtaining density from the ideal gas law to complex tasks like switching to a different model grid, changing the geographic locations assigned to each processor, or even switching to a different set of processors to enable parallel execution over processes. Interface code will be distinguished from the actual process code by appending *_interface* to the appropriate file names. Developers can also append the suffix *_utils* to files to indicate these are helper files for that process.

### 2.2.2 Basic SCREAM–AD Utilities

**It will be the responsibility of the process development team** to design the code for all of the steps mentioned so far in this section. **It will be the responsibility of the AD development team** to maintain the atm_process class and to provide a checklist for new process development. In addition, there will be a number of standard utilities developed in the SCREAM–AD that will assist process developers in bringing their processes into the SCREAM framework. The most powerful of these tools will be the field manager, which is the subject of section 2.4. The AD will also supply I/O routines, tools for accessing runtime options, energy and mass conservation checks, and timestepping schemes.

To promote consistency across the entire model, the AD will also provide a single location for all universal constants. Developers will be encouraged to search this location for any constants that their specific process uses before adding new, process specific constants. If a constant isn't really universal (for example, parameters used only by a process-specific function) then they will be included in the *<process_name_>_utils* code. Similarly, there will be a C++ class for universal functions like linear regression or conversion from potential temperature to temperature.

## 2.3 Coupling Between Atmospheric Processes

The initialization, run, and finalize methods for all processes included in the model will be called from top-level functions of the same name. The top-level initialization routine will also load the grid information for the SE dycore and for the physics mesh which depends on it. Our first implementation will use the existing Fortran code (with C++ wrappers) for domain initialization. To further simplify our initial version, we will use the dynamics grid decomposition for physics even though physics benefits from disbursing columns geographically while dynamics does best when all elements on a processor are neighbors. Future versions of SCREAM–AD will explore more sophisticated grid decomposition methods for physics, including use of a physics grid which is independent of dynamics (like Herrington et al. (2019)). Loading restart data will also be initially handled in the SCREAM–AD using C++ wrappers of Fortran routines from EAM. The top-level run routine will be called once per timestep, and will itself call the run interface function for each included process. These interface functions are themselves expected to call the actual process calculation as described above. It will also be responsible for ingesting inputdata, for writing output, and for interacting with the component coupler in order to obtain surface fluxes. At the finalization step the AD will call the finalization routine for each atmospheric process, deallocate the variables in the field manager (replacement for PBUF) and will deallocate any variables specific to the AD itself.

The top-level initialization, run, and finalize methods will initially handle processes one-at-a-time, but we will ensure these methods can be extended to operate on all processes simultaneously in the future. Because all processes are instances of the generic atm_process class, it will be trivial to change process ordering or to add new processes. The SCREAM–AD will be designed to follow the process order given as a runtime option in an external YAML file (as described in section 2.5). Because there are physical reasons for preferring certain process orders Donahue and Caldwell (2018), we envision ability to change ordering primarily as a tool for sensitivity testing and for ease of adding new processes. Attempts to use uninitialized variables will be caught by the FM.

If all processes are going to be treated identically, we can't have physics parameterizations return tendencies and dynamics return updated state. Our solution is to keep track of the state at the beginning of the timestep and a state after one or more processes have acted. All atmospheric processes

will receive these two states as input. This allows processes to operate on the most recent state or to compute the tendencies from other processes (by differencing the initial and most-recent state) which can then be dribbled into the new process calculation over a series of substeps. Storing states rather than tendencies is better because converting from tendency to state can result in negative concentrations due to rounding errors which are avoided by insisting that all processes return non-negative values.

For more advanced timestepping and splitting methods, the SCREAM–AD will back out tendencies from differences in state before and after parameterizations were called, then apply those tendencies to obtain a new state for the beginning of the next step.

## 2.4 Passing Data Between Processes: The Field Manager

Passing variables between processes is an important task for the AD. There are many options for how to do this, but most of them are problematic. For example, including all variables in a single derived datatype is unworkable because thousands of variables are passed between parameterizations and passing them all into/out of each parameterization would severely impact performance. Defining all variables required by multiple parameterizations in the AD-level "run" routine and passing just the variables needed between routines violates the "uniform input/output" requirement needed for each process to be a generic member of the atm_process class. Including all variables in a single module loaded by all processes makes it too easy to access variables which aren't initialized yet. All of these methods are also problematic from the standpoint of writing restarts. Managing variables in a way which allows all variables needed for restarts to be written out in an automated loop has huge advantages. Ability to associate metadata with each variable is also very helpful. Not only does metadata allow for writing data to a more intelligible format than pure binary (e.g. a netCDF file), it also enables error checking that data dimensions and variable definitions are compatible.

The Physics BUFfer (PBUF) used in E3SMv1 provides a solution to these problems. It requires each parameterization that creates a field to give it the necessary data to allocate the field and assign it a name, as well as whether or not it will have to be written to a restart file. Then any other process

can request access to the same field through a pointer. The use of pointers instead of full data arrays reduces the memory footprint of storing multiple copies of the same data. The Field Manager (FM) in SCREAM will extend the PBUF from EAM, with all the functionality of the PBUF along with new features. The essential differences between PBUF and the FM will be described in the next subsections.

### 2.4.1 Specified intent of variables

When an atmospheric process requests access to a variable in the FM it will also designate how the variable will be used, similar to intent IN, OUT, and IN/OUT in Fortran. This provides the FM with the ability to designate how a variable will be used when it is accessed.

Designating how a variable is used will make it easier to track which atmospheric processes access the variable and where it is changed. Each atmospheric process will have a set of two methods, *required_fields* and *computed_fields*, which will designate the list of fields needed by the process and changed by the process. This will enable the field manager to provide a list of pointers to access those fields that includes that appropriate const correctness, depending on if we expect the field to be changed or not. The field manager will have routines that create a schematic of all field managed variables, what routines use them and for what purpose. By default, variables declared as *required* will be controlled by declaring these variables as constants such that an error is raised if they are changed (const–correctness). They will be further checked by comparing the time when the variable was last changed against the time when it was last expected to change. For more robust variable audits, the field manager can keep a local copy of the entire set of field managed variables and compare with the same variables after each process is called to make sure only those variables that are intended to be changed are changed. This would be a DEBUG flag compile option.

### 2.4.2 Variables handled by the Field Manager

Unlike EAM, which used derived datatypes to pass state and tendency information between physics parameterizations and PBUF to pass other variables, the FM will be used to pass *all* variables. This will eliminate the need to pass multiple structures between atmospheric processes. Prognostic variables traditionally included in the EAM 'state' variable will continue to be treated

differently. They will be given a special designation in FM which ensures they will be advected and prevents them from being designated output variables by parameterizations. Only allowing the AD layer to update these variables ensures that timestepping is handled centrally and important variables aren't updated by accident. If an atmospheric process needs to change a prognostic variable for internal calculations, it have to create a local copy of that variable for this purpose.

### 2.4.3 AD data structure

All fields stored in the FM will share a common data structure for consistency. Given that most of the atmospheric processes in SCREAM use the physics domain the initial implementation of the FM will use the physics data structure on the physics map as the standard universal data structure. This means that atmospheric processes that use a different data or a different domain decomposition will need to map the FM managed variable to a local variable in the appropriate data structure in their interface layer. For example, the dynamics interface will have to map data from the physics structure to the dynamics structure before calling the main dynamics routine and will need to map the output from dynamics back to the physics data structure before returning control to the AD. This is what is currently done in E3SM via the *pd_coupling* and *dp_coupling* routines respectively. The FM will have a set of mapping tools specifically designed to efficiently map between different data structures.

### 2.4.4 Vectorization

Every process which requests a field will include an optional input that specifies the length of vectorization, $N$, the process wants to use for that specific field. If the optional input is not exercised than the assumption will be $N = 1$, no vectorization. The FM will then make sure the allocation for that field can accommodate all the requested vector lengths.

## 2.5 Specifying Runtime Options

Runtime options are a powerful tool from EAM that will be adopted in SCREAM. In EAM, there are two sets of runtime options, those controlled by XML (e.g. env_run.xml) and those controlled by namelist (see user_nl_cam).

Currently, the runtime options controlled by XML are related to CIME and involve global run settings such as run length, timestep, coupling frequency, run type, etc. Model specific runtime options are handled by namelist. The C++ coding language does not have an equivalent to the Fortran namelist parser, so a task of developing the SCREAM–AD is to design a user–interface in the C++ architecture. The SCREAM–AD will adopt a YAML interface for handling runtime options.

The combination of treating all processes as objects of the same class and using YAML to parse runtime options provides the flexibility to have the coupling mechanism between atmospheric processes be a runtime option. Similarly, defining the process order will be straightforward using this interface.

## 2.6 Input/Output

In its initial implementation, SCREAM–AD will adopt PIO2 for input and output. Because PIO2 supports ADIOS, this will allow us to take advantage of ADIOS' ability to write output to many local files rather than enduring the large communication costs needed to write to a single file at very high core counts. Although PIO2 is written in C++, the F90 code layer needed to actually use it is complicated. Thus our initial version of SCREAM will call EAM's F90 I/O routines using C-to-Fortran wrappers. These include the $addfld$, $outfld$, $add\_default$ and $whist$ routines. Future versions of the AD will address rewriting the input/output routines in C++. Because all variables handled by the FM will use the same data structure format, it will be easier to write output for any variable. Like EAM, it will be possible to write out a variable's state at any point within a timestep even though most output will be sampled at the end of a timestep. It will be possible to write restart files at any frequency, for example hourly, daily or monthly. There will be a change to the standard naming convention used for scream based output, the current filename convention of RUNNAME.cam.hX.DATE.nc will be replaced with the more informative RUNNAME.**scream.TYPE**.DATE.nc. Where TYPE will be the frequency of output, e.g. daily, monthly, yearly, etc.

SCREAM–AD will be built using the same external libraries as CIME and as such will have access to all of the standard CIME tools including the *perf_mod* package of performance metrics. Phase 1 of the SCREAM–AD will use a C++ wrapper to call the set of performance Fortran routines. Taking

advantage of the generalized "atmospheric process" object class, the placement of performance timer will be automatic for the initialization, running and finalization of all atmospheric processes. This will be enforced by hard–coding the call to the performance timers inside the atmospheric process class. There will also be the option to include extra timer flags to be called within a specific atmospheric process.

### 2.7 Concluding Remarks

The list of utilities described here is not an exhaustive list, but it represents the set of tasks and goals we believe are required to produce a working model in the three–year timeframe of the SCREAM project. There are very likely other functions and utilities that will arise over the course of the project. As such, the SCREAM–AD development process will be flexible and prepared to tackle any new issues as they come. For example, when the new component coupler becomes available we anticipate devoting a major amount of resources towards incorporating this into the SCREAM–AD interface.

## 3 Nonhydrostatic Spectral Element Dycore

### 3.1 Theory

Put explanation of continuous equations here.

### 3.2 Numerical Methods

Describe the numerical methods used to discretize the equations here.

### 3.3 Computational Implementation

Describe the strategies used (if any) to ensure good computational performance.

### 3.4 Verification

Describe testing strategy

# 4 Simplified Higher-Order Closure (SHOC)

## 4.1 Introduction

Simplified Higher Order Closure (SHOC Bogenschutz and Krueger, 2013) is a parameterization of subgrid-scale (SGS) clouds and turbulence. It is formulated to parameterize SGS shallow cumulus, stratiform cloud, and boundary layer turbulence in models that can either resolve deep convection or has an existing deep convection parameterization. SHOC is an assumed-PDF based parameterization and uses a double Gaussian PDF to diagnose cloud fraction, cloud water, and higher-order turbulence moments. SHOC is only a liquid cloud parameterization, thus it is assumed that any model SHOC is implemented in can treat the ice cloud phase.

Table 1 describes the SHOC prognostic variables and their nomenclature to be used throughout this document.

The liquid water potential temperature, $\theta_l$, is defined as:

$$\theta_l \approx \theta - \frac{L_v}{c_{pd}} q_l \tag{1}$$

where $\theta$ is potential temperature, $L_v$ is the latent heat of vaporization, $c_{pd}$ the specific heat of dry air at constant pressure, $q_l$ the liquid water mixing ratio. The turbulent kinetic energy ($\overline{e}$) is defined as

$$\overline{e} = 0.5(\overline{u'^2} + \overline{v'^2} + \overline{w'^2}), \tag{2}$$

where $\overline{u'^2}$, $\overline{v'^2}$, and $\overline{w'^2}$ represent the SGS zonal, meridional, and vertical wind variances, respectively.

In the SHOC parameterization, all prognostic variables are defined vertically at the mid-point of the grid box.

Table 1: Prognostic Variables in SHOC

| variable | description | units |
|---|---|---|
| $\theta_l$ | Liquid water potential temperature | K |
| $q_t$ | Total water mixing ratio (vapor + cloud liquid) | kg/kg |
| $e$ | Turbulent kinetic energy | $m^2/s^2$ |
| $u$ | Zonal wind component | m/s |
| $v$ | Meridional wind component | m/s |
| $c$ | Tracer constituent | varies |

Table 2 describes key diagnostic variables used throughout the SHOC parameterization, and their respective locations on the vertical grid. Note that many diagnostic variables are defined at the interfaces of the grid box. This is because many diagnostic variables are the result of centered vertical differences of the prognostic variables.

Table 2: Key Diagnostic Variables in SHOC. M in the location column indicates that the variable is located vertically in the mid-point of the grid box, while I indicates that the variable is located at the grid interfaces.

| variable | description | units | location |
|---|---|---|---|
| $L$ | Turbulent Length Scale | m | M |
| $\overline{\theta_l'^2}$ | Temperature variance | $K^2$ | I |
| $\overline{q_t'^2}$ | Moisture variance | $kg^2/kg^2$ | I |
| $\overline{w'^2}$ | Vertical velocity variance | $m^2/s^2$ | M |
| $\overline{w'\theta_l'}$ | Vertical temperature flux | K m/s | I |
| $\overline{w'q_t'}$ | Vertical moisture flux | m/s kg/kg | I |
| $\overline{q_t'\theta_l'}$ | Temperature and moisture covariance | K kg/kg | I |
| $\overline{w'^3}$ | Third moment of vertical velocity | $m^3/s^3$ | I |
| $\overline{w'\theta_v'}$ | Buoyancy flux | K m/s | M |
| $K_m$ | Eddy diffusivity for momentum | $m^2/s$ | M |
| $K_h$ | Eddy diffusivity for heat | $m^2/s$ | M |

The code for SHOC breaks down each process into a separate subroutine. Briefly, the order of operations of SHOC is described below. Each process is then expanded upon with its own section.

SHOC order of operations:

1. **Diagnose Turbulence Length Scale** (section 4.2): The length scale represents the size of unresolved large eddies in a column. This is needed to close the TKE equation and to diagnose several second order moments.

2. **Solve the Turbulence Kinetic Energy Equation** (section 4.3): Advance the TKE equation (due to shear production, buoyant production, and dissipation processes) one time step. Note that advection of TKE is performed by the host model, while turbulent transport of TKE is done by SHOC turbulence diffusion.

3. **Perform Turbulence Diffusion** (section 4.4): Using eddy coefficients derived from TKE, advance $\overline{u}$, $\overline{v}$, $\overline{\theta_l}$, $\overline{q_t}$ , $\overline{e}$, and any tracers ($\overline{c}$) one time step using an implicit diffusion solver.

4. **Diagnose the Second Order Moments** (section 4.5): Diagnose $\overline{q_t'^2}$, $\overline{\theta_l'^2}$, $\overline{q_t'\theta_l'}$, $\overline{w'^2}$, $\overline{w'\theta_l'}$, and $\overline{w'q_t'}$. These are the second order moments needed to close the assumed PDF.

5. **Diagnose the Third Order Moment** (section 4.6): Diagnose the third moment of vertical velocity ($\overline{w'^3}$), needed to parameterize vertical velocity skewness in the assumed PDF

6. **Compute Assumed PDF** (section 4.7): Use the Assumed PDF to compute SGS cloud water, cloud fraction, and the buoyancy flux ($\overline{w'\theta_v'}$).

In SHOC the order of operations is chosen deliberately so that the prognostic variables are updated first and the clouds are diagnosed last. This is to prevent supersaturation from occurring when SHOC is complete, to avoid any potential conflicts with a microphysics scheme which may be called in the host model.

## 4.2 Turbulence Length Scale

The empirical formulation is based on the finding that the turbulent length scale is highly correlated with the distance from the wall, strength of the turbulence, boundary layer depth, and local thermal stability (Bogenschutz et al. 2010). Within the turbulent boundary layer, the length scale definition is set equal to an asymptotic shape, similar to that of Blackadar (1962). However it is weighted more strongly by the strength of the turbulence. This reflects the behavior that as the grid size increases, the SGS TKE increases and so does the mixing length. The effects of thermal stability are also included to reduce the length scale where the local stability is large.

The formulation in (3) is empirically determined from LES data and essentially represents a geometric average between the strength of the SGS TKE (as suggested by Texieria et al. 2004) and an asymptote length scale, with a contribution due to stability effects. The geometric average assures that in close proximity to the surface, the length scale will be small.

$$L = \sqrt{8\left[\frac{1}{\tau\sqrt{e}kz} + \frac{1}{\tau\sqrt{e}L_\infty} + 0.01\delta\frac{N^2}{\overline{e}}\right]^{-1}} \tag{3}$$

Above, $k$ is the von Karman constant. $L_\infty$ is the asymptotic value of the length scale as defined in Blackadar (1962) as

$$L_\infty = 0.1 \frac{\int_0^\infty \overline{e}^{1/2} z dz}{\int_0^\infty \overline{e}^{1/2} dz}. \tag{4}$$

In equation 3 $\delta$ is defined as:

$$\delta = \begin{cases} 1 & \text{if} \quad N^2 > 0 \\ 0 & \text{if} \quad N^2 \leq 0 \end{cases}$$

where $N^2$ is the moist Brunt Vaisala Frequency. In SHOC $N^2$ is defined as:

$$N^2 = \frac{g}{\overline{\theta_v}} \frac{\partial \overline{\theta_v}}{\partial z} \tag{5}$$

where $\theta_v$ is the virtual potential temperature defined as:

$$\theta_v = \theta(1 + 0.61 q_v - q_l), \tag{6}$$

where $q_v$ is the water vapor mixing ratio. Finally, $\tau$ in equation 3 represents the eddy turnover timescale and is defined as

$$\tau = \frac{D_b}{w_*}, \tag{7}$$

where $D_b$ is the boundary layer depth and is computed according to that of Holtslag and Boville (1993). $w_*$ represents the convective velocity scale, integrated from the surface to the height of the boundary layer depth, and is defined as:

$$w_*^3 = 2.5 \frac{g}{\overline{\theta_v}} \int_0^{z_{D_b}} \overline{w' \theta_v'} dz. \tag{8}$$

In the event that $w_*^3 < 0$, which is indicative of a stable boundary layer, then $\tau$ is set to a default value of 100 s.

## 4.3 Turbulent Kinetic Energy Equation

In SHOC, the turbulent kinetic energy (TKE) equation to be solved is given by:

$$\frac{\partial \overline{e}}{\partial t} = \underbrace{-\overline{u_j} \frac{\partial \overline{e}}{\partial x_j}}_{\text{advection}} + \underbrace{\frac{g}{\overline{\theta_v}} \left( \overline{w' \theta_v'} \right)}_{\text{buoyant production}} - \underbrace{P_s}_{\text{shear production}} - \underbrace{\frac{\partial \overline{w' e}}{\partial z}}_{\text{turbulent transport}} - \underbrace{C_{ee} \frac{\overline{e}^{3/2}}{L}}_{\text{dissipation}}. \tag{9}$$

The first term on the RHS of equation 9 is advection, which is performed by the host model (i.e. SCREAM dynamics) and not SHOC.

The second term is the buoyant production of TKE. The buoyancy flux term ($\overline{w'\theta'_v}$) is closed by integrating over the assumed PDF (see section 4.7) using equation 50. Thus, $\overline{w'\theta'_v}$ from the previous SHOC time step is used to close this term.

The shear production term is computed according to Bretherton and Park (2009):

$$-P_s = -\overline{w'u'}\frac{\partial \overline{u}}{\partial z} - \overline{w'v'}\frac{\partial \overline{v}}{\partial z} = K_M S^2, \tag{10}$$

where

$$S^2 = \left(\frac{\partial \overline{u}}{\partial z}\right)^2 + \left(\frac{\partial \overline{v}}{\partial z}\right)^2. \tag{11}$$

Since $\overline{u}$ and $\overline{v}$ are located vertically in the mid-points, $S^2$ is computed on the interface grid, then interpolated onto the mid-point grid. After the shear production term is calculated on the interface grid, it is interpolated to the mid-point grid to be consistent with the location of $\overline{e}$.

The boundary surface value of $K_M S^2$ is set to zero as the boundary fluxes for TKE are applied in the diffusion solver.

The fourth term on the RHS of equation 9 represents the turbulent transport of TKE. This term is computed in the turbulent diffusion (section 4.4) of SHOC.

The last term on the RHS of equation 9 represents the turbulent dissipation of TKE. Here $C_{ee}$ is a turbulent constant, which is defined in Deardorff (1980) as $C_{ee} = C_{e1} + C_{e2}$, where $C_{e1} = C_e/0.133$ and $C_{e1} = C_e/0.357$ and $C_e = C_k^3/C_s^4$. Finally, $C_k = 0.1$ and $C_s = 0.15$.

### 4.3.1 Eddy Diffusivities

**4.3.1.1 Default Formulation** After TKE is updated due to buoyant production, shear production, and dissipation processes, the eddy diffusivity parameters for heat and momentum, to be used in turbulence diffusion, are respectively defined in the TKE module as:

$$K_H = C_{Kh}\tau_v\overline{e} \tag{12}$$

$$K_M = C_{Km}\tau_v\overline{e} \tag{13}$$

where $C_{Kh}$ and $C_{Km}$ are tunable constants. $C_{Kh}$ and $C_{Km}$ could be tuned independently, but as a starting point we set them equal to 0.1. In equations 12 and 13 $\tau_v$ represents a damped return to isotropic timescale where $\tau = 2\overline{e}/\epsilon$ and

$$\tau_v = \tau \left[1 + \lambda_0 N^2 \tau^2\right]^{-1} \tag{14}$$

where $\lambda_0 = 0$ if $N^2 < 0$ and $\epsilon$ is the turbulence dissipation rate (last term of equation 9). If $N^2 > 0$ then $\lambda_0$ is set as a ramp function in terms of the integrated column stability in the lower troposphere ($N^2_\infty$):

$$\lambda_0 = \lambda_{min} + \lambda_{slope} * (\frac{N^2_\infty}{g} - N_{low}), \tag{15}$$

$$N^2_\infty = \int_{1000hPa}^{800hPa} N^2 dz. \tag{16}$$

Where $\lambda_{min} = 0.001$, $\lambda_{slope} = 0.35$, and $N_{low} = 0.037$. Here, $\lambda_{slope}$ is an adjustable parameter. $\lambda_0$ has a minimum threshold of 0.001 and a maximum threshold of 0.04.

**4.3.1.2 Stable Boundary Layer** For the case of a moderate to very stable boundary layer, the formulation of the eddy diffusivities are revised to promote sufficient mixing, as they are based primarily on turbulence shear production, and prevent runaway cooling. We use the dimensionless Obukov length, $z/L$ to determine when to trigger the stable boundary layer eddy diffusivities, where $z$ is the height of the lowest mid-point grid height and L the Monin-Obukhov length defined as

$$L = -\frac{u_*^3 \overline{\theta_v}}{kg\left(\overline{w'\theta'_v}\right)_s}. \tag{17}$$

The stable boundary layer formulation for the eddy diffusivities triggers when $z/L$ is greater than 100, which signifies a moderately or very stable boundary layer. These stable boundary layer diffusivities are applied for the PBL depth within this column and are defined as:

$$K_H = C_{Khs} L^2 S \tag{18}$$

and

$$K_M = C_{Kms} L^2 S, \tag{19}$$

where $C_{Khs}$ and $C_{Kms}$ are the stable boundary eddy coefficients for heat and momentum, respectively. By default these values are set to 1.

## 4.4 Turbulence Diffusion

The prognostic variables for SHOC (table 1) are updated due to turbulence diffusion via:

$$\frac{\partial \overline{\chi}}{\partial t} = -\frac{\partial \overline{w'\chi'}}{\partial z}. \tag{20}$$

Where $\chi$ represents any of SHOC's prognostic variables ($\theta_l$, $q_t$, $u$, $v$, $e$, or $c$). SHOC uses downgradient diffusion to represent the vertical flux of turbulence using:

$$\overline{w'\chi'} = -K_\chi \frac{\partial \chi}{\partial z}, \tag{21}$$

where $K_\chi$ represents either $K_H$ or $K_M$.

To preserve numerical stability equations 20 and 21 are solved using an implicit backward Euler scheme for the diffusion of $\theta_l$, $q_t$, $u$, $v$, $e$, or $c$. Given an input state $\chi^*$ and diffusivity profile:

$$\frac{\chi(t+\Delta t) - \chi^*}{\Delta t} = \frac{\partial}{\partial z}\left( K_\chi(z) \frac{\partial}{\partial z} \chi(t+\Delta t) \right). \tag{22}$$

In SHOC the surface fluxes for heat, moisture, TKE, and tracers are explicitly deposited into the lowest model layer and then implicit diffusion is performed. For TKE the bottom surface flux is defined as

$$u_*^3 = max(\sqrt{(}(\overline{u'w'}_{sfc} + \overline{v'w'}_{sfc})^{0.5}), 0.01). \tag{23}$$

However, the method of explicit surface fluxes results in a numerically unstable solution for momentum since such explicit adding can flip the direction of the lowest model layer wind ($\overline{u}_s^*$, $\overline{v}_s^*$), especially when the lowest model layer is thin. Thus, the surface momentum fluxes ($\tau_x^* = \overline{u'w'}_s$,$\tau_y^* = \overline{v'w'}_s$) in SHOC are added in an implicit way. This is done by computing the total momentum surface stress and applying this as a boundary condition in equation 22:

$$k_{tot} = max[\sqrt{(}(\tau_x^*)^2 + (\tau_y^*)^2)/max(\sqrt{(}(\overline{u}_s^*)^2 + (\overline{u}_s^*)^2), 1), 10^{-4}]. \tag{24}$$

The procedure for the solution of the implicit equation 22 follows that of Richtmyer and Morton (1967) pages 198-200.

## 4.5 Diagnosis of Second Order Moments

In order to close the assumed PDF (section 4.7) we need to diagnose several second order moments. Namely, we need to determine, $\overline{w'\theta_l'}$, $\overline{w'q_t'}$, $\overline{q_w'\theta_l'}$, $\overline{q_t'^2}$, $\overline{\theta_l'^2}$, and $\overline{q_w'\theta_l'}$.

The expression we use to determine $\overline{w'\theta_l'}$ and $\overline{w'q_t'}$ is based on downgradient diffusion as:

$$\overline{w'C'} = -K_H \frac{\partial \overline{C}}{\partial z} \tag{25}$$

where $C$ is interchanged for $\theta_l$ and $q_t$.

For the scalar variances and covariances, SHOC diagnoses these terms as:

$$\overline{q_t'^2} = C_{q_t} S_m \left( \frac{\partial \overline{q_t}}{\partial z} \right)^2 \tag{26}$$

$$\overline{\theta_l'^2} = C_{\theta_l} S_m \left( \frac{\partial \overline{\theta_l}}{\partial z} \right)^2 \tag{27}$$

$$\overline{q_t'\theta_l'} = C_{q_t\theta_l} S_m \frac{\partial \overline{q_t}}{\partial z} \frac{\partial \overline{\theta_l}}{\partial z}, \tag{28}$$

where $S_m = \tau_v K_H$. $C_{q_t}$, $C_{\theta_l}$, and $C_{q_t\theta_l}$ are tunable coefficients to adjust the strength of diagnosed variances and covariances. Default setting for these coefficients are $C_{q_t}$, $C_{\theta_l}$, and $C_{q_t\theta_l} = 1.0$.

Note that $\overline{w'\theta_l'}$, $\overline{w'q_t'}$, $\overline{q_w'\theta_l'}$, $\overline{q_t'^2}$, $\overline{\theta_l'^2}$, and $\overline{q_t'\theta_l'}$ are all computed on the interface grid. Thus, before the computation of these terms $K_H$ and $\tau_v$ are linearly interpolated to the interface grid.

The expression for $\overline{w'^2}$ is:

$$\overline{w'^2} = \frac{2}{3}\overline{e} \tag{29}$$

## 4.6 Third Moment of Vertical Velocity

The final term needed to close the assumed PDF is the third order moment of vertical velocity ($\overline{w'^3}$), which is parameterized following that of (Canuto et al., 2001). Canuto et al. (2001) provides expressions for several third-order moments, but we are only interested in $\overline{w'^3}$.

The expressions provided by Canuto et al. (2001) were originally derived for the dry convective boundary layer and we simply replace potential temperature with liquid water potential temperature ($\overline{\theta_l}$) to make the expressions valid in moist convection. The original dynamic equations for the third order moment can be found in Canuto (1992) and these equations entail fourth-order moments that can be written as

$$\overline{a'b'c'd'} = \left(\overline{a'b'}\,\overline{c'd'} + \overline{a'c'}\,\overline{b'd'} + \overline{a'd'}\,\overline{b'c'}\right) F. \tag{30}$$

If function $F$ is taken to be unity then the above expression reduces to the quasi-normal approximation. This was done in Canuto et al. (1994) but the results of some of their third-order moments were not satisfactory when compared to LES data.

The expression for $\overline{w'^3}$ is as follows:

$$\overline{w'^3} = \left(\Omega_1 - 1.2X_1 - \frac{3}{2}f_5\right)\left(c - 1.2X_0 + \Omega_0\right)^{-1}, \tag{31}$$

with the functions $X$ and $\Omega_0$ defined as

$$\begin{aligned} X_0 &= \gamma_2\tilde{N}^2\left(1-\gamma_3\tilde{N}^2\right)\left[1-\left(\gamma_1+\gamma_3\right)\tilde{N}^2\right]^{-1} \\ X_1 &= \left[\gamma_0 f_0 + \gamma_1 f_1 + \gamma_2\left(1-\gamma_3\tilde{N}^2\right)f_2\right]\left[1-\left(\gamma_1+\gamma_3\right)\tilde{N}^2\right]^{-1} \\ \Omega_0 &= \omega_0 X_0 + \omega_1 Y_0 \\ \Omega_1 &= \omega_o X_1 + \omega_1 Y_1 + \omega_2. \end{aligned} \tag{32}$$

The $\omega$ function's are given by

$$\begin{aligned} \omega_0 &= \gamma_4\left(1-\gamma_5\tilde{N}^2\right)^{-1} \\ \omega_1 &= (2c)^{-1}\,\omega_0 \\ \omega_2 &= \omega_1 f_3 + \frac{5}{4}\omega_0 f_4. \end{aligned} \tag{33}$$

The $\gamma$'s are constants which depend on the adjustable parameter $c$. Canuto et al. (2001) and previous work found that $c = 7$, although small variations

are allowed. The $\gamma$ constants are given by:

$$\begin{aligned}
\gamma_0 &= 0.52c^{-2}\left(c-2\right)^{-1} \\
\gamma_1 &= 0.87c^{-2} \\
\gamma_2 &= 0.5c^{-1} \\
\gamma_3 &= 0.60c^{-1}\left(c-2\right)^{-1} \\
\gamma_4 &= 2.4\left(3c+5\right)^{-1} \\
\gamma_5 &= 0.6c^{-1}\left(3c+5\right)^{-1}.
\end{aligned} \tag{34}$$

Finally, the functions are introduced which incorporate the second-order moments of $\overline{w'^2}$, $\overline{w'\theta'_l}$, $\overline{\theta'_l}$, and $\overline{e}$. These are defined as follows:

$$\begin{aligned}
f_0 &= \left(g\alpha\right)^3 \tau_v^4 \overline{w'\theta'_l} \frac{\partial \overline{\theta'^2_l}}{\partial z} \\
f_1 &= \left(g\alpha\right)^2 \tau_v^3 \left( \overline{w'\theta'_l} \frac{\partial \overline{w'\theta'_l}}{\partial z} + \frac{1}{2} \overline{w'^2} \frac{\partial \overline{\theta'_l}}{\partial z} \right) \\
f_2 &= g\alpha\tau_v^2 \overline{w'\theta'_l} \frac{\partial \overline{w'^2}}{\partial z} + 2g\alpha\tau_v^2 \overline{w'^2} \frac{\partial \overline{w'\theta'_l}}{\partial z} \\
f_3 &= g\alpha\tau_v^2 \left( \overline{w'^2} \frac{\partial \overline{w'\theta'_l}}{\partial z} + \overline{w'\theta'_l} \frac{\partial \overline{e}}{\partial z} \right) \\
f_4 &= \tau_v \overline{w'^2} \left( \frac{\partial \overline{w'^2}}{\partial z} + \frac{\partial \overline{e}}{\partial z} \right) \\
f_5 &= \tau_v \overline{w'^2} \frac{\partial \overline{w'^2}}{\partial z}.
\end{aligned} \tag{35}$$

All of the above $f$ functions have the dimensions of velocity cubed. In addition, we define $\tilde{N}^2 = \tau_v^2 N^2$.

Once $\overline{w'^3}$ is determined we perform clipping to ensure that this calculation does not produce unrealistically large values. $\mid \overline{w'^3} \mid$ is constrained by $w3_{clip}\sqrt{2.0\overline{w'^2}}$. Where $w3_{clip}$ is an adjustable parameter with a default value of 1.2.

## 4.7 Assumed PDF

Here details of the Analytic Double Gaussian (ADG) 1 PDF (as referred to in Larson et al. (2002), which is the PDF used in SHOC, are presented. The

input moments for this PDF are $\overline{\theta_l}$, $\overline{q_t}$, $\overline{w'^2}$, $\overline{w'\theta_l'}$, $\overline{w'q_t'}$, $\overline{q_w'\theta_l'}$, $\overline{q_t'^2}$, $\overline{\theta_l'^2}$, $\overline{q_w'\theta_l'}$, and $\overline{w'^3}$. Note that at the beginning of this module $\overline{w'\theta_l'}$, $\overline{w'q_t'}$, $\overline{q_w'\theta_l'}$, $\overline{q_t'^2}$, $\overline{\theta_l'^2}$, $\overline{w'^3}$ are interpolated to the mid-point grid.

This PDF, as the name suggests, is based on the double Gaussian form as

$$P_{adg1}(w', \theta_l', q_t') = aG_1(w', \theta_l', q_t') + (1-a)G_2(w', \theta_l', q_t'). \tag{36}$$

Here $G_1$ and $G_2$ are the individual Gaussians and the parameters for the ADG 1 can be found analytically. To do this, some assumptions have to be made. The first assumption is that the subplume variations in $w$ are uncorrelated with those in $q_t$ and $\theta_l$. Letting $i = 1$ or 2, the individual Gaussians in equation 36 are then given by

$$\begin{aligned} G_i(w', \theta_l', q_t') = \frac{1}{(2\pi)^{3/2}\sigma_{wi}\sigma_{q_t i}\sigma_{\theta_l i}(1-r^2_{q_t\theta_l i})^{1/2}} \exp\left[-\frac{1}{2}\left(\frac{w' - (w_i - \overline{w})}{\sigma_{wi}}\right)^2\right] \\ \times \exp\left(-\frac{1}{2(1-r^2_{q_t\theta_l i})}\left\{\left[\frac{q_t' - (q_{ti} - \overline{q_t})}{\sigma_{q_t i}}\right]^2 + \left[\frac{\theta_l' - (\theta_{li} - \overline{\theta_l})}{\sigma_{\theta_l i}}\right]^2\right.\right. \\ \left.\left. - 2r_{q_t\theta_l i}\left[\frac{q_t' - (q_{ti} - \overline{q_t})}{\sigma_{q_t i}}\right]\left[\frac{\theta_l' - (\theta_{li} - \overline{\theta_l})}{\sigma_{\theta_l i}}\right]\right\}\right). \end{aligned} \tag{37}$$

Now we must define the PDF parameters. The PDF parameters are based on the equations of Lewellen and Yoh (1993) and are found by integrating over the 12 relevant input moments over the double Gaussian PDF. Four of these equations are (the rest are analogous):

$$\begin{aligned} \overline{w} &= aw_1 + (1-a)w_2 \\ \overline{w'^2} &= a[(w_1 - \overline{w})^2 + \sigma_{w1}^2] + (1-a)[(w_2 - \overline{w})^2 + \sigma_{w2}^2] \\ \overline{w'^3} &= a[(w_1 - \overline{w})^3 + 3(w_1 - \overline{w})\sigma_{w1}^2] + (1-a)[(w_2 - \overline{w})^3 + 3(w_2 - \overline{w})\sigma_{w2}^2] \\ \overline{w'q_t'} &= a[(w_1 - \overline{w})(q_{t1} - \overline{q_t}) + r_{wq_t1}\sigma_{w1}\sigma_{q_{t1}}] + (1-a)[(w_2 - \overline{w})(q_{t2} - \overline{q_t}) + r_{wq_t2}\sigma_{w2}\sigma_{q_{t2}}]. \end{aligned} \tag{38}$$

with the relative amplitude of the Gaussian $a$ is defined as

$$a = \frac{1}{2}\left\{1 - Sk_w\left[\frac{1}{4(1-\tilde{\sigma}_w^2)^3 + Sk_w^2}\right]^{1/2}\right\}. \tag{39}$$

This is obtained by assuming that the standard deviations of the two Gaussians are equal in $w$ and integrating over the PDF. Here $Sk_w \equiv \overline{w'^3}/(\overline{w'^2}^{3/2})$,

represents the skewness of vertical velocity. In the case of $\overline{w'^2}$=0 it is assumed that the PDF reduces to a single delta function. The parameters for $w_1$ and $w_2$ are given by:

$$\tilde{w}_1 \equiv \frac{w_1 - \overline{w}}{\sqrt{\overline{w'^2}}} = \left(\frac{1-a}{a}\right)^{1/2} (1 - \tilde{\sigma}_w^2)^{1/2} \tag{40}$$

and

$$\tilde{w}_2 \equiv \frac{w_2 - \overline{w}}{\sqrt{\overline{w'^2}}} = \left(\frac{a}{1-a}\right)^{1/2} (1 - \tilde{\sigma}_w^2)^{1/2}. \tag{41}$$

To avoid numerical instabilities in the model a threshold for $a$ must be defined as $0.01 \leq a \leq 0.99$. We also have the definitions of $\tilde{\sigma}_w \equiv \sigma_{w1}/\sqrt{\overline{w'^2}} = \sigma_{w2}/\sqrt{\overline{w'^2}}$ and $\tilde{\sigma}_w^2 = 0.4$.

Now to define terms for $\theta_{l1}$ and $\theta_{l2}$ we get:

$$\tilde{\theta}_{l1} \equiv \frac{\theta_{l1} - \overline{\theta_l}}{\sqrt{\overline{\theta_l'^2}}} = -\frac{\overline{w'\theta_l'}/(\sqrt{\overline{w'^2}}\sqrt{\overline{\theta_l'^2}})}{\tilde{w}_2} \tag{42}$$

and

$$\tilde{\theta}_{l2} \equiv \frac{\theta_{l2} - \overline{\theta_l}}{\sqrt{\overline{\theta_l'^2}}} = -\frac{\overline{w'\theta_l'}/(\sqrt{\overline{w'^2}}\sqrt{\overline{\theta_l'^2}})}{\tilde{w}_1}. \tag{43}$$

Should there be no variability in $\theta_l$ then the means of the Gaussians are set equal so that $\theta_{l1} = \theta_{l2} = \overline{\theta_l}$ and the widths of the Gaussians in the $\theta_l$ direction are set to zero.

Unlike vertical velocity, the widths in the $\theta_l$ direction are allowed to differ. These are found by integrating over the PDF and defined as:

$$\frac{\sigma_{\theta_l 1}^2}{\overline{\theta_l'^2}} = \frac{3\tilde{\theta}_{l2}[1 - a\tilde{\theta}_{l1}^2 - (1-a)\tilde{\theta}_{l2}^2] - [Sk_{\theta_l} - a\tilde{\theta}_{l1}^3 - (1-a)\tilde{\theta}_{l2}^3]}{3a(\tilde{\theta}_{l2} - \tilde{\theta}_{l1})} \tag{44}$$

and

$$\frac{\sigma_{\theta_l 2}^2}{\overline{\theta_l'^2}} = \frac{3\tilde{\theta}_{l1}[1 - a\tilde{\theta}_{l1}^2 - (1-a)\tilde{\theta}_{l2}^2] - [Sk_{\theta_l} - a\tilde{\theta}_{l1}^3 - (1-a)\tilde{\theta}_{l2}^3]}{3(1-a)(\tilde{\theta}_{l2} - \tilde{\theta}_{l1})}. \tag{45}$$

To prevent unrealistic solutions the following condition is set

$$0 \leq \frac{\sigma^2_{\theta_l 1,2}}{\overline{\theta_l'^2}} \leq 100. \tag{46}$$

Analogous equations are used to find $\tilde{q}_{t1,2}$ and $\sigma^2_{qt1,2}$.

The equations above make clear that SHOC is dependent on the skewness of $\theta_l$ and $q_t$. For the ADG 1 PDF, neither $\overline{\theta_l'^3}$ and $\overline{q_t'^3}$ are input moments, therefore diagnostic assumptions must be made. $Sk_{\theta_l}$ is simply set to zero for the ADG 1 PDF as it is found that this value prevents numerical instabilities from being introduced. To represent skewness in cumulus layers the following conditions are set for $Sk_{q_t}$: When $|\tilde{q}_{t2} - \tilde{q}_{t1}| > 0.4$ we set $Sk_{q_t} = 1.2Sk_w$. When $|\tilde{q}_{t2} - \tilde{q}_{t1}| \leq 0.2$ we set $Sk_{q_t} = 0$. Between these two extremes $Sk_{q_t}$ is linearly interpolated.

The within-plume correlations are computed by setting $r_{q_t\theta_l 1} = r_{q_t\theta_l 2}$ and integrating over the PDF to obtain an equation for $\overline{q_t'\theta_l'}$ and hence:

$$r_{q_t\theta_l 1,2} = \frac{\overline{q_t'\theta_l'} - a(q_{t1} - \overline{q_t})(\theta_{l1} - \overline{\theta_l}) - (1-a)(q_{t2} - \overline{q_t})(\theta_{l2} - \overline{\theta_l})}{a\sigma_{q_t 1}\sigma_{\theta_l 1} + (1-a)\sigma_{q_t 2}\sigma_{\theta_{l2}}} \tag{47}$$

with the condition that

$$-1 \leq r_{q_t\theta_l 1,2} \leq 1 \tag{48}$$

because correlations must lie between -1 and 1.

Now that we have defined the PDF parameters, we can now diagnose SGS cloud and turbulence terms. Cloud fraction, liquid water content, and liquid water flux are all given by:

$$\begin{aligned} C &= a(C)_1 + (1-a)(C)_2 \\ \overline{q_l} &= a(\overline{q_l})_1 + (1-a)(\overline{q_l})_2 \\ \overline{w'q_l'} &= a[(w_1 - \overline{w})(\overline{q_l}) + (\overline{w'q_l'})_1] + (1-a)[(w_2 - \overline{w})(\overline{q_l})_2 + (\overline{w'q_l'})_2]. \end{aligned} \tag{49}$$

In addition, the buoyancy flux can be closed using the expression:

$$\overline{w'\theta_v'} = \overline{w'\theta_l'} + \frac{1-\epsilon_o}{\epsilon_o}\theta_o \overline{w'q_t'} + \left[\frac{L_v}{c_p}\left(\frac{p_o}{p}\right)^{R_d/c_p} - \frac{1}{\epsilon_o}\theta_o\right]\overline{w'q_l'} \tag{50}$$

The individual cloud fraction $C$ and mean specific liquid water content $\overline{q_l}$ are calculated by linearizing the variability in $\theta_l$ and $q_t$ (with analogous expressions for the Gaussian 1 and 2 for equations 51 though 58):

$$C = \frac{1}{2}\left[1 + \text{erf}\left(\frac{s}{\sqrt{2}\sigma_s}\right)\right] \tag{51}$$

and

$$\overline{q_l} = sC + \frac{\sigma_s}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{s}{\sigma_s}\right)^2\right]. \tag{52}$$

Here erf is the error function and $\sigma_s$ is the standard deviation of $s$, which is equal to the liquid water content when $s$ is greater than zero, but can also be negative and is conserved under condensation. These two terms are defined as (Lewellen and Yoh, 1993):

$$\begin{aligned} s &= q_t - q_s(T_l, p)\frac{(1+\beta q_t)}{[1+\beta q_s(T_l,p)]} \\ \sigma_s^2 &= c_{\theta_l}^2\sigma_{\theta_l}^2 + c_{q_t}^2\sigma_{q_t}^2 - 2c_{\theta_l}\sigma_{\theta_l}c_{q_t}\sigma_{q_t}r_{q_t\theta_l} \end{aligned} \tag{53}$$

where $q_s$ is the saturation mixing ratio with respect to either water or ice or a hybrid of the two depending on the temperature, and $\beta$ is defined as:

$$\beta = \beta(T_l) = \frac{R_d}{R_v}\left(\frac{L_v}{R_d T_l}\right)\left(\frac{L_v}{c_p T_l}\right). \tag{54}$$

Also defined are the following terms:

$$c_{q_t} = \frac{1}{1+\beta(T_l)q_s(\overline{T_l}, p)} \tag{55}$$

and

$$c_{\theta_l} = \frac{1+\beta(\overline{T_l})\overline{q_t}}{[1+\beta(\overline{T_l})q_s(\overline{T_l},p)]^2}\frac{c_p}{L_v}\beta(\overline{T_l})q_s(\overline{T_l},p)\left(\frac{p}{p_o}\right)^{R_d/C_p} \tag{56}$$

Finally, the flux of liquid water is given by:

$$\overline{w' q_l'} = C\overline{w' s'} \tag{57}$$

where

$$\overline{w' s'} = c_{q_t}\sigma_w\sigma_{q_t}r_{wq_t} - c_{\theta_l}\sigma_w\sigma_{\theta_l}r_{w\theta_l}. \tag{58}$$

In the above expressions $q_s$ is defined as:

$$q_s(T_l, P) = \frac{R_d}{R_v}\frac{e_s(T_l)}{p - [1-(R_d/R_v)]e_s(T_l)}. \tag{59}$$

Here $q_s$ is the saturation specific humidity, $e_s$ is the saturation vapor pressure over liquid, $p$ is pressure, $c_p$ is the specific heat at constant pressure, and $R_d$

and $R_v$ are the gas constants for dry air and water vapor. In addition, we define $T_l$ as the liquid water temperature:

$$T_l = T - \frac{L_v}{c_p} q_l \tag{60}$$

where $T$ is temperature. In SHOC, $e_s$ is computed based on Flatau et al. (1992).

# 5 Predicted Particle Properties (P3)

Describe scheme in general (copy/paste Hassan's existing doc)

## 5.1 Autoconversion

Say what autoconversion does

### 5.1.1 Theory

Put explanation of continuous equations here.

### 5.1.2 Numerical Methods

Describe the numerical methods used to discretize the equations here.

### 5.1.3 Computational Implementation

Describe the strategies used (if any) to ensure good computational performance.

### 5.1.4 Verification

Describe testing strategy

## 5.2 Accretion

Say what accretion does

#### 5.2.1 Theory

Put explanation of continuous equations here.

#### 5.2.2 Numerical Methods

Describe the numerical methods used to discretize the equations here.

#### 5.2.3 Computational Implementation

Describe the strategies used (if any) to ensure good computational performance.

#### 5.2.4 Verification

Describe testing strategy

# 6 Rapid Radiative Transfer for Global models in Parallel (RRTMGP)

## 6.1 Theory

Put explanation of continuous equations here.

## 6.2 Numerical Methods

Describe the numerical methods used to discretize the equations here.

## 6.3 Computational Implementation

Describe the strategies used (if any) to ensure good computational performance.

## 6.4 Verification

Describe testing strategy

# 7 Calculation of Sea Level Pressure

Over land, surface pressure is dominated by the land surface elevation rather than the position of low pressure centers, fronts, etc. In order to see these features, surface pressure must be converted to sea level pressure, also known as the pressure at sea level (PSL, denoted here by $p_s$), which is the surface pressure we would have if the landmass was not there. There's no single accepted set of assumptions for how this "subterranean atmosphere" should behave. What is its lapse rate? How much moisture does it contain? Because PSL is just a diagnostic output for EAMxx, our current implementation just copies what EAMf90 did, which was inherited from an early version of the Community Atmosphere Model. This approach is outlined in Section 3.1B of Trenberth et al. (1993) but that document isn't very clear so we describe the approach below.

## 7.1 Formulation

The basic idea is to assume the subterranean atmosphere is free of moisture and has a lapse rate $\Gamma = 6.5$ K km$^{-1}$ unless the atmosphere is too warm or too cold, in which case the lapse rate and the ground temperature are empirically modified to provide more reasonable results (as described later). The hydrostatic equation $\partial p/\partial z = -\rho_{\text{air}} g$ is integrated from a height of zero (sea level) to the height of the ground to compute sea level pressure. Because the model stores geopotential $\Phi$ rather than geopotential height $z$, we rewrite the hydrostatic equation using $\Phi = gz$ (where $g$ is the gravitational constant). We also use the ideal gas law $\rho_{\text{air}} = p/(R_d T)$ (where $R_d$ is the gas constant for dry air) to write density in terms of pressure and temperature. Thus our hydrostatic equation is

$$\frac{\partial p}{\partial \Phi} = \frac{-p}{R_d T} \tag{61}$$

Because lapse rate is assumed linear, the temperature at any $\Phi$ is

$$T = T_{\text{g}} + \frac{\Gamma}{g}\left(\Phi_g - \Phi\right) \tag{62}$$

where the subscript $g$ refers to the value of the atmosphere at ground level (i.e. the bottom interface of the lowest atmosphere grid cell). Note that $\Gamma$ is positive in this formulation. For our subterranean atmosphere, $\Phi$ varies between 0 (sea level) and $\Phi_g$, which should be positive except in a few rare

depressions like Death Valley. As a result, we expect $T$ to be warmer than $T_g$ most but not all of the time.

Integrating (61) over the vertical after substituting (62) yields:

$$\int_{p_s}^{p_g} \frac{\partial p}{p} = \int_0^{\Phi_g} \frac{-\partial \Phi}{R_d \left[ T_{\rm g} + \frac{\Gamma}{g} \left( \Phi_g - \Phi \right) \right]}. \tag{63}$$

For $\Gamma \neq 0$, this equation can be solved exactly by making the substitution $x = R_d(T_{\rm g} + \Gamma/g\,(\Phi_g - \Phi))$, which yields

$$p_s = p_g \left[ 1 + \frac{\Gamma \Phi_g}{g T_g} \right]^{g/(R_d \Gamma)}. \tag{64}$$

If $\Gamma = 0$, the exact solution is

$$p_s = p_g \exp\left\{ \frac{\Phi_g}{R_d T_g} \right\}. \tag{65}$$

Because we will set $\Gamma$ to zero when $_g$ gets too warm, using exact solutions would require conditional execution which is inefficient on GPUs. Also, computing non-integer exponents is expensive. Thus we note that the integrand on the right-hand side of (63) can be written as $A \cdot 1/(1-y) = A \sum_{k=0}^{\infty} y^k$ for $A = 1/(R_d T_g)$ and $y = \Gamma/(gT_g)(\Phi - \Phi_g)$. Because $y$ is always much less than 1, accurate approximations can be made from just the first 3 summands. This leads to

$$p_s \approx p_g \exp\left\{ \beta \left[ 1 - \frac{\alpha\beta}{2} + \frac{(\alpha\beta)^2}{3} \right] \right\} \tag{66}$$

for $\alpha = \Gamma R_d/g$ and $\beta = \Phi_g/(R_d T_g)$.

If $T_g < 255$ K, (66) is applied with $T_g$ modified according to $\widetilde{T}_g = (255 - T_g)/2$. If $T_g > 290.5$ K, a trial sea-level temperature $\widetilde{T}_s$ is computed using (62) with $\Phi = 0$ and $\Gamma = 6.5$ K km$^{-1}$. If $\widetilde{T}_s$ is also greater than 290.5 K, $\Gamma$ is set to zero (to prevent temperatures near sea levels from getting even warmer) and $\widetilde{T}_g = (290.5 - T_g)/2$ is used in (66). If $T_g > 290.5$ K but $\widetilde{T}_s < 290.5$ K, excessive temperatures are prevented by choosing $\Gamma$ such that sea-level temperature just reaches 290.5 K. Note that these crude hacks accomplish their intended goal of avoiding extreme temperatures in a smooth way when $\Phi_g > 0$, but may not be appropriate when $\Phi_g < 0$. We didn't bother to improve this treatment because accurate $p_s$ isn't a major goal for us and the current treatment survived without complaint for the last 28 yrs.

## 7.2 Validation

Fig. 1 compares the exact solutions from (64) and (65) against the 2nd order Taylor series we use in our calculations and the cruder 1st order approximation. Both approximations agree very well with the exact solutions. In fact, the first-order approximation is probably sufficient for our uses - it is at most 1 mb worse than the 2nd order result in even the most extreme cases. For the $\Gamma = 0$ case, both approximations are off by as much as 6 mb, but the 1st order approach isn't any worse than the 2nd order version.

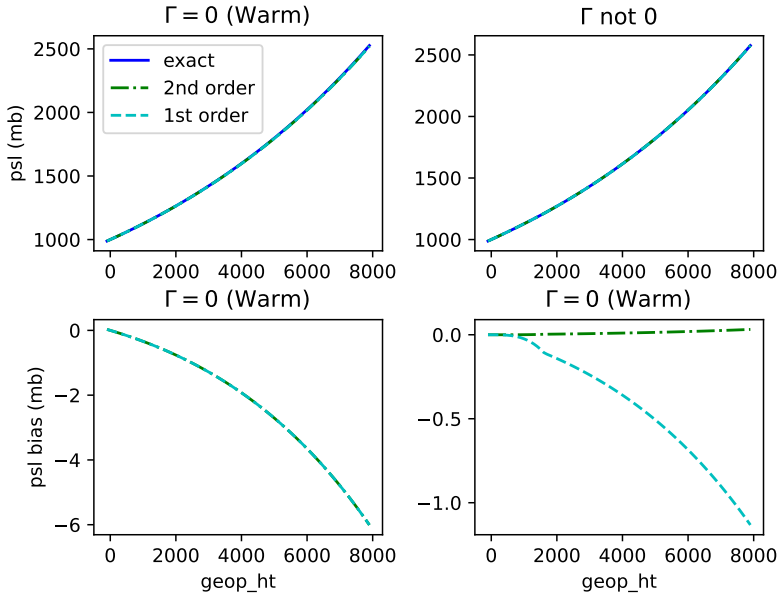


Figure 1: Tests of exact versus approximate $p_s$ calculations. Left panels are warm enough that $\Gamma = 0$ ($T_g = 292$ K) while right panels use $\Gamma = 6.5$ K km$^{-1}$ ($T_g = 280$ K). Top panels show actual $p_s$ and bottom panels show bias. Sea level pressures are much too high because $p_g = 1000$ mb, but this should be fine for testing purposes. Note that the lowest elevation sampled is -100 m to test behavior when terrain is below sea level.

Fig. 2 compares $p_s$ from an old F90 simulation against values calculated using our approximate formula. As expected, results look very similar despite using midpoint $T$ as an approximation for $T_g$. Using a first-order rather than 2nd order Taylor series once again has an insignificant effect.

### 7.3 Unit Tests

1. Is C++ value close to exact solution for very warm conditions where $\Gamma$ should be zero?

2. Is C++ value close to exact solution for normal conditions where $\Gamma$ should be 6.5 K km$^{-1}$?

3. Is $p_s < p_g$ when $\Phi_g round > 0$ (for very cold, moderate, and very warm $T_g$ and a variety of $p_g$ values)?

4. Is $p_s > p_g$ when $\Phi_g round < 0$ (for very cold, moderate, and very warm $T_g$ and a variety of $p_g$ values)?

5. Test that modified $T_g$ values for very cold and very warm conditions are applied correctly. Not sure how to do this - should computation of $\widetilde{T}_g$ be its own unit? Or should modified $T_g$ be computed in the unit test itself and stuffed into the exact solution to provide an independent calculation of the expected solution? Or is this too dumb to test?

# 8 Bibliography

# References


Blackadar, A. K., 1962: The vertical distribution of wind and turbulent exchange in a neutral atmosphere. *Journal of Geophysical Research (1896-1977)*, **67**, 3095–3102.
URL `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/JZ067i008p03095`

Bogenschutz, P. A. and S. K. Krueger, 2013: A simplified pdf parameterization of subgrid-scale clouds and turbulence for cloud-resolving models. *Journal of Advances in Modeling Earth Systems*, **5**, 195–211.
URL `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/jame.20018`

Bretherton, C. S. and S. Park, 2009: A new moist turbulence parameterization in the community atmosphere model. *Journal of Climate*, **22**, 3422–3448.
URL `http://dx.doi.org/10.1175/2008JCLI2556.1`

Canuto, V., Y. Cheng, and A. Howard, 2001: New third-order moments for the convective boundary layer. *Journal of the Atmospheric Sciences*, **58**, 1169–1172.

Canuto, V., F. Minotti, C. Ronchi, R. Ypma, and O. Zeman, 1994: Second-order closure pbl model with new third-order moments: Comparison with les data. *Journal of the Atmospheric Sciences*, **51**.

Canuto, V. M., 1992: Turbulent convection with overshooting: Reynolds stress approach. *Astrophys. J.*, **392**, 218–232.

Deardorff, J. W., 1980: Stratocumulus-capped mixed layers derived from a three-dimensional model. *Boundary-Layer Meteorology*, **18**, 495–527.

Donahue, A. S. and P. M. Caldwell, 2018: Impact of physics parameterization ordering in a global atmosphere model. *Journal of Advances in Modeling Earth Systems*, **10**, 481–499.
URL `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2017MS001067`

Flatau, P. J., R. L. Walko, and W. R. Cotton, 1992: Polynomial Fits to Saturation Vapor Pressure. *Journal of Applied Meteorology*, **31**, 1507–1513.

Herrington, A. R., P. H. Lauritzen, M. A. Taylor, S. Goldhaber, B. E. Eaton, J. T. Bacmeister, K. A. Reed, and P. A. Ullrich, 2019: Physics–dynamics coupling with element-based high-order galerkin methods: Quasi-equal-area physics grid. *Monthly Weather Review*, **147**, 69–84.
URL `https://doi.org/10.1175/MWR-D-18-0136.1`

Larson, V., J.-C. Golaz, and W. Cotton, 2002: Small-scale and mesoscale variability in cloudy boundary layers: Joint probability density functions. *Journal of Atmospheric Sciences*, **59**, 3519–3539.

Lewellen, W. S. and S. Yoh, 1993: Binormal model of ensemble partial cloudiness. *Journal of the Atmospheric Sciences*, **50**, 1228–1237.

Richtmyer, R. and K. Morton, 1967: *Difference methods for initial value problems*. Interscience, New York, NY, 405 pp. pp.

Trenberth, K. E., J. C. J. C. Berry, and L. E. Buja, 1993: Vertical interpolation and truncation of model-coordinate data. NCAR Technical Note TN-396+STR, NCAR, Boulder CO, USA.

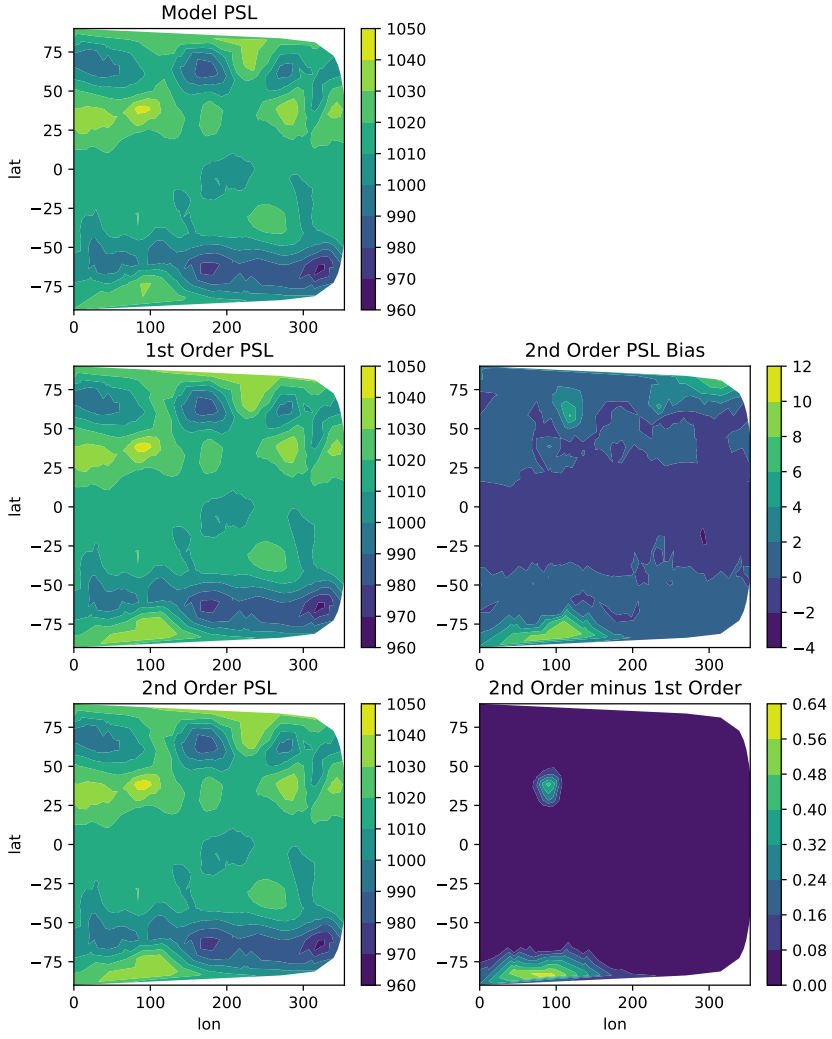


Figure 2: Comparison of $p_s$ from an old ne4 F90 model simulation (top left) versus calculation of the same quantity using (66). Panels on the left show $p_s$ and right-hand panels show differences between simulations. Calculated $p_s$ used midpoint rather than surface temperature because the latter wasn't available; this likely accounts for most of the F90 vs new calculation differences we see; despite this imperfection, biases are still acceptably low.