FEBRUARY 2026 @ CESM Winter Meeting, AMWG

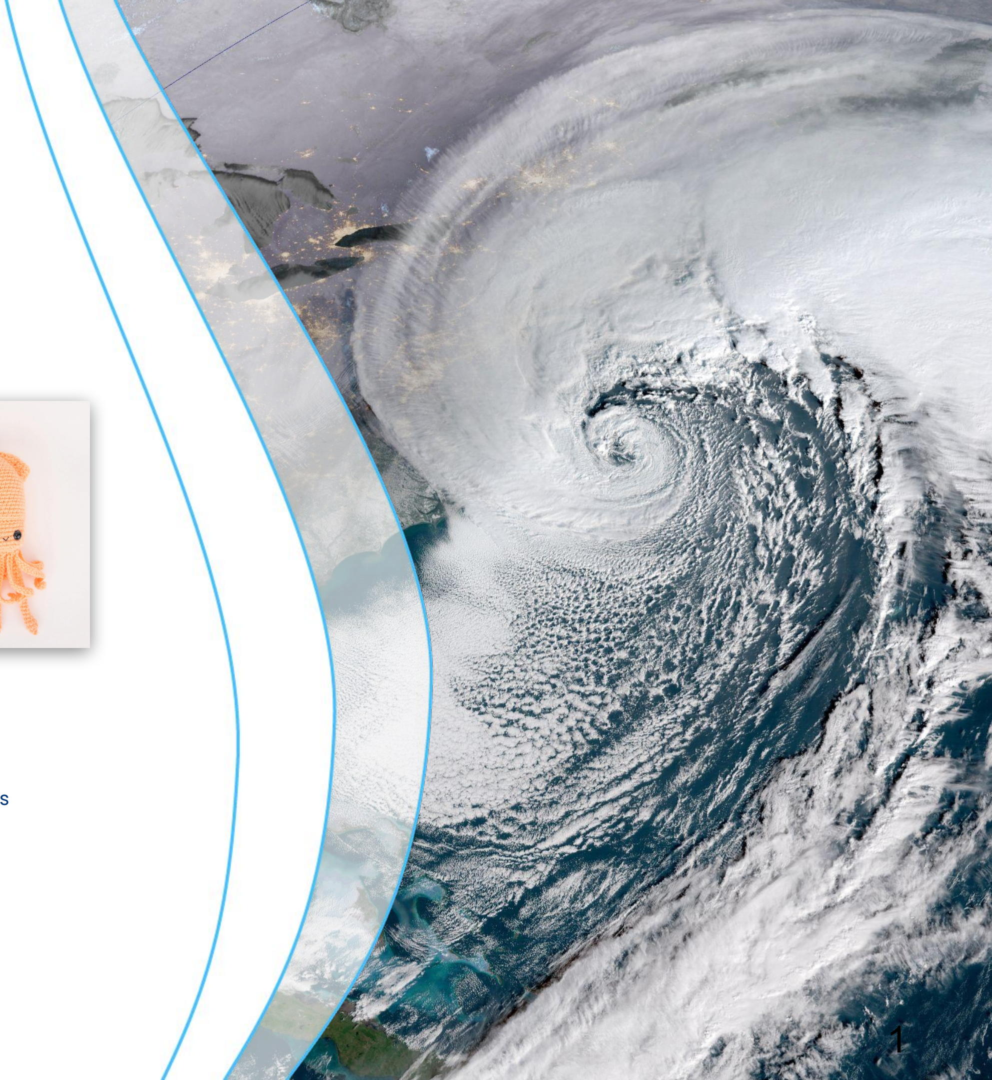
QPC4 in CAM-SIMA

Haipeng Lin

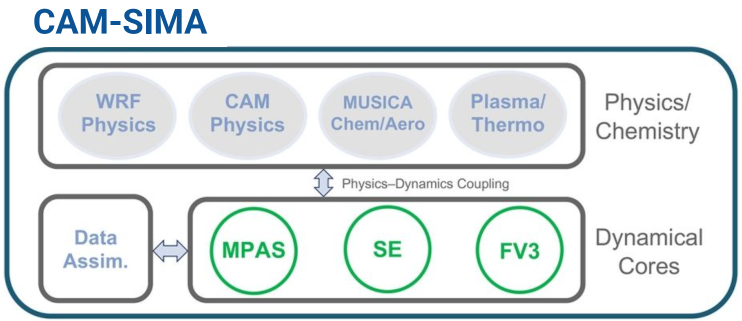
AMP, CGD, NSF NCAR

With contributions from Cheryl Craig, Jesse Nusbaumer, Courtney Peverley, Peter Lauritzen, Adam Herrington, Isla Simpson, Kate Thayer-Calder, Francis Vitt, and others

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



Brief recap/overview of CAM-SIMA: next generation of CAM with CCPP physics and modern software engineering practices



Advantages compared to CAM

- Physics packages refactored and converted to be "CCPP-compliant" – driven automatically with inputs/outputs validated at build time
- Easy reordering of physics schemes via XML file
- "Snapshot"-tested physics schemes
- Automated namelist read
- Python and Fortran-based Unit Tests
- … we'll talk about many of these today!

Haipeng Lin / hplin@ucar.edu

What do you mean by "CCPP-compliant" physics parameterizations (schemes)?

Init/Run/Final Phases

Standardized phases for each scheme (all phases are optional)

CAM-SIMA automatically handles namelist reads based on a namelist.xml file for each scheme - no more writing `_readnl()`!

Metadata

The CCPP framework automatically handles input/output into/from schemes based on this information, **replacing the physics buffer (pbuf)**.

Standard names identify unique physical quantities.

Units are checked and converted as necessary - "built-in documentation"!

```
[ kvm ]
  standard_name =
eddy_momentum_diffusivity_at_interfaces
  units = m2 s-1
  dimensions = (horizontal_loop_extent,
vertical_interface_dimension)
  type = real | kind = kind_phys
  intent = inout
```

No "host model" dependencies* = improved portability

(* best effort)

Avoiding CAM-specific code allows for physics code to be better shared across models

Final assembly: Suite Definition Files (SDFs) build model configurations based on a list of schemes.

A XML file **replaces "physpkg.F90"** making it easy to reorder parameterizations and add calls to individual schemes:

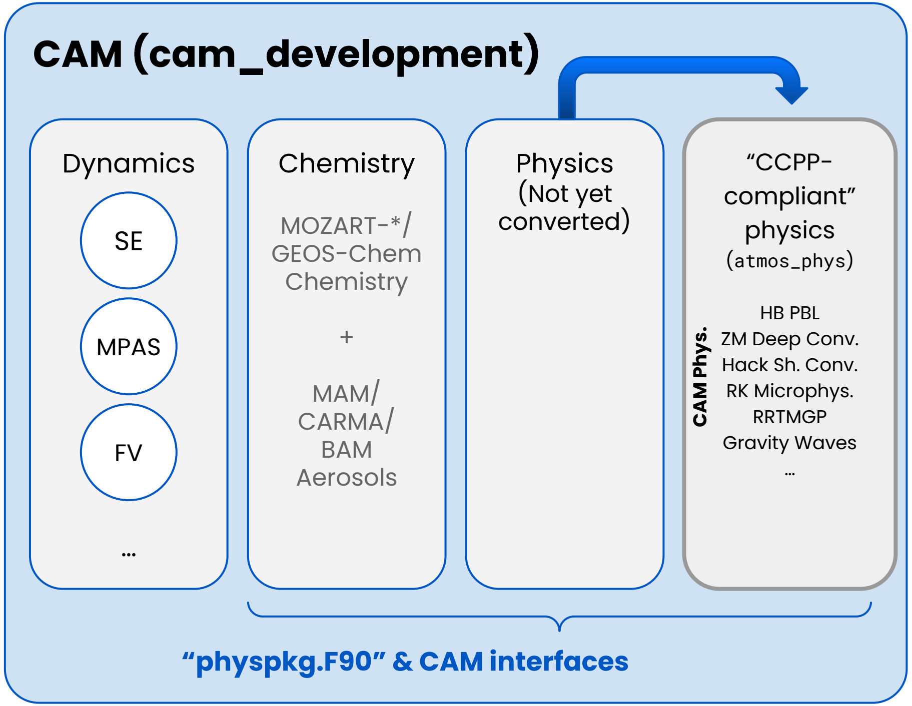
```
<scheme>cloud_particle_sedimentation</scheme>
```

Instead of

```
call cloud_particle_sedimentation( … )
```


Haipeng Lin / hplin@ucar.edu

Conversion of CAM(4) physics to CCPP-compliant was done under the hood

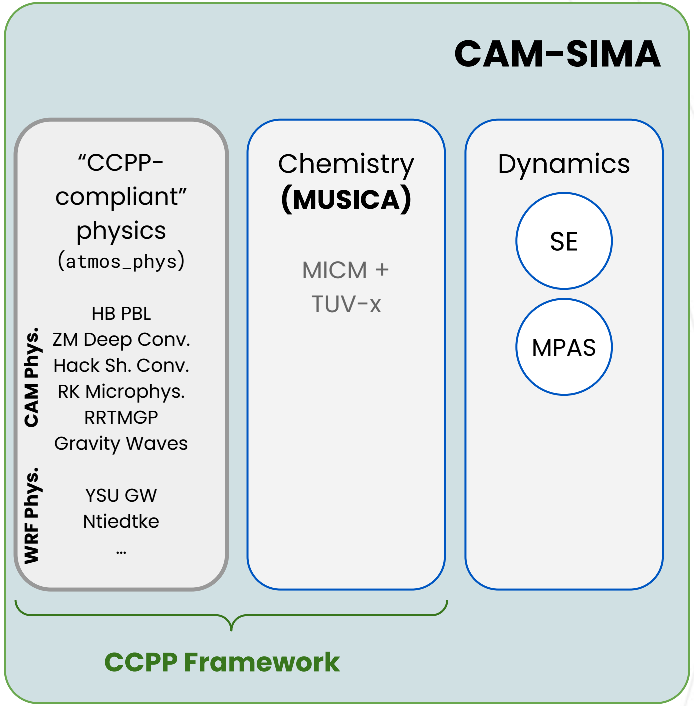


Physics parameterizations from CAM have been converted and moved to the *atmospheric_physics* repository, leaving only minor interface code behind in CAM…

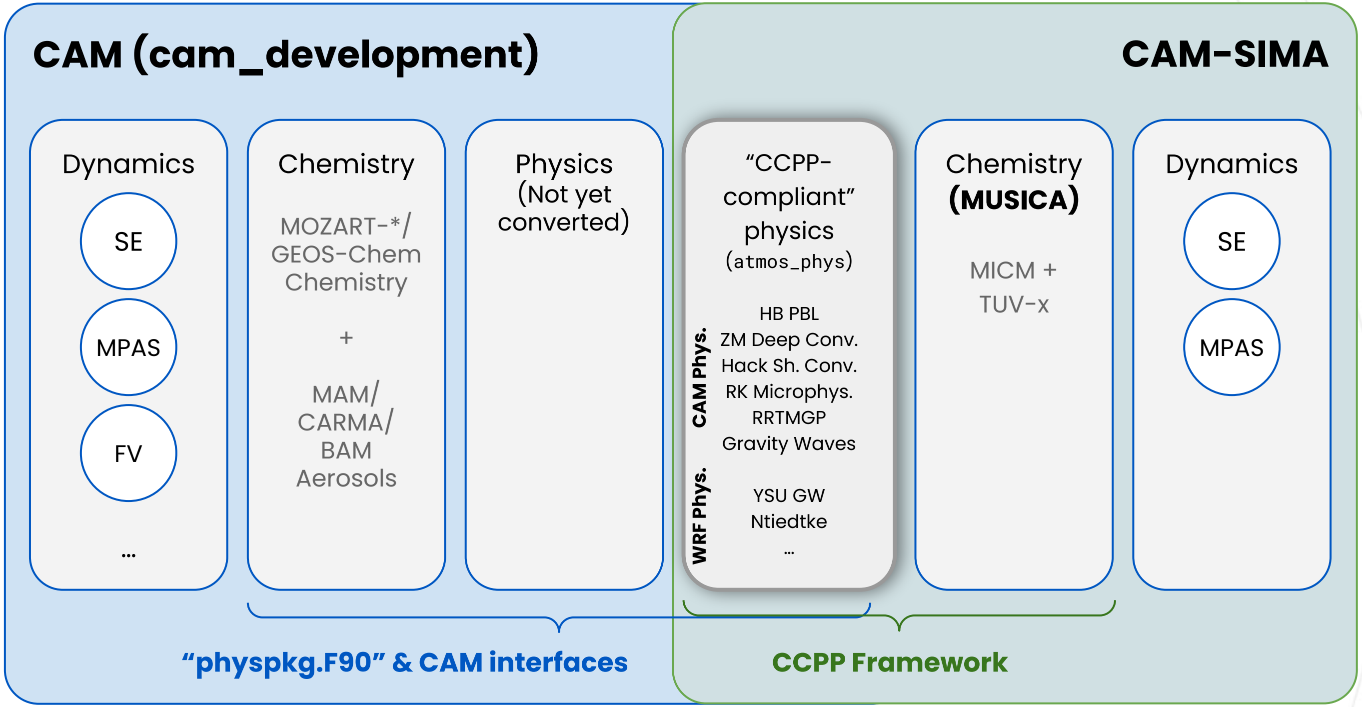
Haipeng Lin / hplin@ucar.edu

Conversion of CAM(4) physics to CCPP-compliant was done under the hood

The converted physics parameterizations are used in CAM-SIMA through the CCPP framework, and can be mixed-and-matched to create complete “suites”, or tested individually against CAM outputs to ensure they’re bit-for-bit…



Conversion of CAM(4) physics to CCPP-compliant was done under the hood



Once converted, the same code is shared by CAM and CAM-SIMA!

Haipeng Lin / hplin@ucar.edu

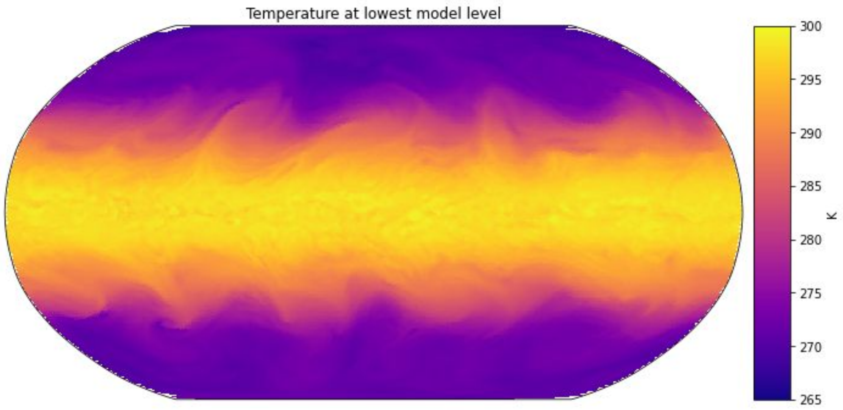
CAM4 physics now converted to CAM-SIMA: QPC4 (CAM4 aquaplanet) config available

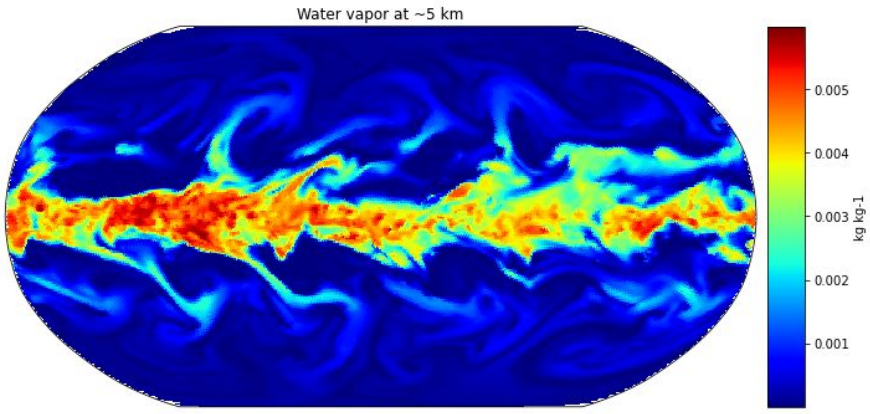
CCPP-compliant / Work in progress

Version (release date) **Coupled Model**	**CAM4** **CCSM4**	**CAM4** ("CAM4-ish") **CAM-SIMA**
PBL scheme	Holtslag-Boville (HB)	
Shallow convection	Hack	
Deep convection	Zhang-McFarlane (ZM)	
Microphysics	Rasch-Kristjánsson (RK)	
Radiation	CAMRT	**RRTMGP**
Gravity wave drag	McFarlane Orographic GW	
Chemistry/Aerosols	None / Bulk Aerosol Model (BAM)	None / BAM
Dynamical cores	FV | EUL | SLD	**SE | MPAS**
Model top, # levels	~42 km 26 lev.	~42 km 26 lev. (**runtime-configurable**)

Haipeng Lin / hplin@ucar.edu

CAM4 physics now converted to CAM-SIMA: QPC4 (CAM4 aquaplanet) now working





Thanks to Jesse Nusbaumer for the plots!
10 year QPC4 in CAM-SIMA, MPAS dycore

Haipeng Lin / hplin@ucar.edu

Converting CAM4 physics: why?
Separating science from “boilerplate”: convect_shallow example

Cleaning up physics code for CCPP

Previously, CAM physics code usually mixed:

- History/diagnostic code (addfld, outfld)
- Host-model specific code (setup pbuf indices, index of Q/cloud liquid, get physics options)
- **Actual science code** (may or not depend on CAM-specific stuff)

```
   !  This field probably should reference the pbuf tpert field but it
doesnt
   tpert(:ncol)         = 0._r8
...
   select case (shallow_scheme)
   case('Hack') ! Hack scheme
      call pbuf_get_field(pbuf, qpert_idx, qpert)
      qpert(:ncol,2:pcnst) = 0._r8

      call cmfmca( lchnk,  ncol, ... )
...
   call pbuf_set_field(pbuf, rprdtot_idx, rprdsh(:ncol,:pver) +
rprddp(:ncol,:pver), start=(/1,1/), kount=(/ncol,pver/))
...
   ftem(:ncol,:pver) = ptend_loc%s(:ncol,:pver)/cpair
   call outfld( 'ICWMRSH ', icwmr, pcols   , lchnk )
   call outfld( 'CMFDT   ', ftem, pcols   , lchnk )
```

Code that is specific vs. non-specific for schemes mixed in one file separated by `SELECT CASE` or `IF`s

CAM-specific “chunking”

Host-model setup code & Diagnostics

Anatomy of convect_shallow.F90 before cleanup: many physics modules look similar.

Haipeng Lin / hplin@ucar.edu

Converting CAM4 physics: why?
Separating science from “boilerplate”: convect_shallow example

Cleaning up physics code for CCPP

Previously, CAM physics code usually mixed:

- History/diagnostic code (addfld, outfld)
- Host-model specific code (setup pbuf indices, index of Q/cloud liquid, get physics options)
- **Actual science code** (may or not depend on CAM-specific stuff)

```
   !  This field probably should reference the pbuf tpert field but it
doesnt
   tpert(:ncol)         = 0._r8
...
   select case (shallow_scheme)
   case('Hack') ! Hack scheme
      call pbuf_get_field(pbuf, qpert_idx, qpert)
      qpert(:ncol,2:pcnst) = 0._r8

      call cmfmca( lchnk,  ncol, ... )
...
   call pbuf_set_field(pbuf, rprdtot_idx, rprdsh(:ncol,:pver) +
rprddp(:ncol,:pver), start=(/1,1/), kount=(/ncol,pver/))
...
   ftem(:ncol,:pver) = ptend_loc%s(:ncol,:pver)/cpair
   call outfld( 'ICWMRSH ', icwmr, pcols   , lchnk )
   call outfld( 'CMFDT   ', ftem, pcols   , lchnk )
```

“Permanent leftovers”

Code that is specific vs. non-specific for schemes mixed in one file separated by `SELECT CASE` or `IF`s

CAM-specific “chunking”

Host-model setup code & Diagnostics

Anatomy of convect_shallow.F90 before cleanup: many physics modules look similar.

Haipeng Lin / hplin@ucar.edu

Converting CAM4 physics: why?

Separating science from "boilerplate": convect_shallow example

Cleaning up physics code for CCPP

Previously, CAM physics code usually mixed:

- History/diagnostic code (addfld, outfld)
- Host-model specific code (setup pbuf indices, index of Q/cloud liquid, get physics options)
- **Actual science code** (may or not depend on CAM-specific stuff)
- **Hidden surprises** ("long-range effects")

```
   !  This field probably should reference the pbuf tpert field but it
doesnt
   tpert(:ncol)         = 0._r8
...
   select case (shallow_scheme)
   case('Hack') ! Hack scheme
      call pbuf_get_field(pbuf, qpert_idx, qpert)
      qpert(:ncol,2:pcnst) = 0._r8

      call cmfmca( lchnk,  ncol, ... )
...
   call pbuf_set_field(pbuf, rprdtot_idx, rprdsh(:ncol,:pver) +
rprddp(:ncol,:pver), start=(/1,1/), kount=(/ncol,pver/))
...
   ftem(:ncol,:pver) = ptend_loc%s(:ncol,:pver)/cpair
   call outfld( 'ICWMRSH ', icwmr, pcols   , lchnk )
   call outfld( 'CMFDT   ', ftem, pcols   , lchnk )
```

"Permanent leftovers"

Code that is specific vs. non-specific for schemes mixed in one file separated by `SELECT CASE` or `IFs`

CAM-specific "chunking"

Host-model setup code & Diagnostics

Anatomy of convect_shallow.F90 before cleanup: many physics modules look similar.

Haipeng Lin / hplin@ucar.edu

Converting CAM4 physics: why?
Separating science from “boilerplate”: convect_shallow example

Separation of concerns

CAM-SIMA “suite definition file” for Hack shallow:

- **“Science code” in dedicated scheme**
- Diagnostic-specific computations cleanly separated (using outputs from the “science code” scheme)
- Separate tendency appliers
- Reuse common code with ZM convective evaporation

```
    <!-- SHALLOW CONVECTION: HACK SCHEME -->
    <scheme>check_energy_zero_fluxes</scheme>
    <scheme>hack_convect_shallow</scheme>
    <scheme>convect_shallow_diagnostics_after_shallow_scheme</scheme>
    <scheme>apply_heating_rate</scheme>
    <scheme>apply_constituent_tendencies</scheme>
    <scheme>qneg</scheme>
    <scheme>geopotential_temp</scheme>

    <!-- SUBCLOUD EVAPORATION -->
    <scheme>cloud_fraction_fice</scheme>
    <!-- prepare state for zm_conv_evap (rename shallow outputs to generic
inputs) -->
    <!-- also zero out quantities going in/out -->
    <scheme>set_shallow_conv_fluxes_to_general</scheme>
    <scheme>zm_conv_evap</scheme>
    <scheme>set_general_conv_fluxes_to_shallow</scheme>
    <scheme>convect_shallow_diagnostics_after_convective_evaporation</scheme>
    <scheme>apply_heating_rate</scheme> ...
```

Hack shallow convection suite definition file: think of it as a portion of the new “physpkg”

Converting CAM4 physics: why?
Avoiding "side-effect" surprises due to pbuf/"use"ing logicals from other modules

A frustrating scenario

- Where is something modified?
- Does this code behave differently when other packages are active? (e.g., "dycore_is", "pbuf_get_index < 0", "phys_getopts", "if(deep_scheme) == 'ZM'" …)

"Long-range" effects across modules hurts code readability for everyone

```
subroutine macrop_driver_init(pbuf2d)
use convect_shallow, only: convect_shallow_use_shfrc
    if( convect_shallow_use_shfrc() ) then
        use_shfrc = .true.
        shfrc_idx = pbuf_get_index('shfrc')
    else
        use_shfrc = .false.
    endif
```

Defined as shallow_scheme = 'UW' in convect_shallow …

… defined as when PBL="UW" in namelist_defaults.xml!!!

"Long-range" use statement from shallow convection affects behavior of macrophysics - *purpose of this switch is opaque and is defined three levels deep*

```
subroutine convect_deep_register
use phys_control, only: phys_getopts, use_gw_convect_dp
  ! If gravity waves from deep convection are on, output this field.
  if (use_gw_convect_dp .and. deep_scheme == 'ZM') then
     call pbuf_add_field('TTEND_DP','physpkg',dtype_r8,(/pcols,pver/),ttend_dp_idx)
  end if
```

Deep convection adds a field to the physics buffer depending on whether a gravity wave scheme is active - *what if someone else tries to use it but use_gw_convect_dp is false?*

Haipeng Lin / hplin@ucar.edu

Converting CAM4 physics: how?
vertical_diffusion_tend / HB PBL scheme example

Subroutines | Interspersed logic **("a few lines of load-bearing code hidden in hundreds")**

Before: CCPP schemes

vertical_diffusion_tend:

- Orographic Form Drag
- Vertical profile interpolation
- Boundary Layer Scheme
- Sponge Layer handling (updates diffusivities directly)
- Surface Stress handling (dependent on PBL scheme)
- Diffusion solver (monolithic handling of horizontal momentum, dry static energy, tracer diffusion)
- Diagnostic calculations

Interspersed logic in big, monolithic subroutines make code reuse and debug difficult.

Haipeng Lin / hplin@ucar.edu

Converting CAM4 physics: how?
vertical_diffusion_tend / HB PBL scheme example

Subroutines — Interspersed logic (**“a few lines of load-bearing code hidden in hundreds”**)

CCPP schemes

Before:

vertical_diffusion_tend:

- Orographic Form Drag
- Vertical profile interpolation
- Boundary Layer Scheme
- Sponge Layer handling (updates diffusivities directly)
- Surface Stress handling (dependent on PBL scheme)
- Diffusion solver (monolithic handling of horizontal momentum, dry static energy, tracer diffusion)
- Diagnostic calculations

Interspersed logic in big, monolithic subroutines make code reuse and debug difficult.

Smaller CCPPized subroutines grouped by purpose clarifies dependencies, and allows mix-and-matching schemes as needed enabled by CCPP

After (in CAM-SIMA and CAM):

suite_vdiff_holtslag_boville:

- interpolate_to_interfaces
- hb_pbl_independent_coefficients
- hb_pbl_dependent_coefficients
- hb_diff_exchange_coefficients
- vertical_diffusion_sponge_layer
- surface_stress_add_drag_coefficient
vertical_diffusion_wind_damping_rate
- diffuse_horizontal_momentum
diffuse_dry_static_energy
diffuse_tracers
vertical_diffusion_tendencies
- vertical_diffusion_diagnostics

e.g.: CLUBB reuses this scheme for its PBL diagnostics.

Haipeng Lin / hplin@ucar.edu

Converting CAM4 physics: is it worth it?
vertical_diffusion_tend and supporting CAM5 UW PBL

CCPP schemes

CAM5 UW PBL:

suite_vdiff_bretherton_park:
- interpolate_to_interfaces
- bretherton_park_diff
- eddy_diffusivity_adjustment_above_pbl
- vertical_diffusion_sponge_layer
- surface_stress_add_drag_coefficient
  vertical_diffusion_wind_damping_rate
- diffuse_horizontal_momentum
  diffuse_dry_static_energy
  diffuse_tracers
  vertical_diffusion_tendencies
- vertical_diffusion_diagnostics

Simply swap in the UW PBL scheme!

CAM4 HB PBL:

suite_vdiff_holtslag_boville:
- interpolate_to_interfaces
- hb_pbl_independent_coefficients
- hb_pbl_dependent_coefficients
- hb_diff_exchange_coefficients
- vertical_diffusion_sponge_layer
- surface_stress_add_drag_coefficient
  vertical_diffusion_wind_damping_rate
- diffuse_horizontal_momentum
  diffuse_dry_static_energy
  diffuse_tracers
  vertical_diffusion_tendencies
- vertical_diffusion_diagnostics

All “diffusion solver” code remains the same

If it builds, all physical quantities are present

Looking to the future: CAM5/6/7, chemistry and aerosols

Further progress on CAM-SIMA continues...

- CAM5 & CAM7 physics underway
- Collaboration with ACOM on chemistry and aerosols:
  - Prescribed ozone and aerosols for radiation in final phases
  - ACOM's **Mu**lti-**s**cale **I**nfrastructure for **C**hemistry and **A**erosols (**MUSICA**) library (gas-phase chemical solver + photolysis) coupled to CAM-SIMA via CCPP
  - Abstract aerosol interface for interfacing bulk/modal/sectional aerosols to physics functional in current CAM; will be ported to CAM-SIMA, along with the **B**ulk **A**erosol **M**odel (**BAM**) as a proof-of-concept
- Even if you are not using CAM-SIMA today, CCPP conversions to physics code in CAM will make code much easier to work with
  - Please do reach out if you would like to work with CAM-SIMA, and
  - Consider making any new physics packages compliant with CCPP conventions so they are ready for the future!

Thanks to all those who contributed to CAM & CAM-SIMA!

Stay tuned for more updates…

FEBRUARY 2026 @ CESM Winter Meeting, AMWG

Non-QPC4 CAM-SIMA Updates

Jesse Nusbaumer
AMP, CGD, NSF NCAR

With contributions from Cheryl Craig, Haipeng Lin, Courtney Peverley, Kuan-Chih Wang, Michael Duda, Peter Lauritzen, Adam Herrington, Jordan Powers, Domi Colegrove, Jimmy Dudhia, and others.

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

Non-CAM configurations

CAM-SIMA is designed to provide scientific configurations beyond CAM, including:

- **NCAR MMM configurations**
- **WACCM configurations**
- **WACCM-X configurations**
- **NOAA UFS configurations**

Specific focus has been on bringing in MMM global weather configurations

MMM configurations

The MPAS-A dycore (v8.3.1) has been fully ported and run with idealized and CAM4 physics suites.

The dynamics-physics coupling layer has been significantly refactored by Kuan-Chih Wang to allow for cleaner separation between the dycore and physics, along with more modular (and unit-tested) routines.

MPAS
Model for Prediction Across Scales

CAM coupling layer

```
do k = 1, pver                                    ! vertical index in physics chunk
   kk = pver - k + 1                              ! vertical index in dynamics block

   phys_state(lchnk)%t(icol_p,k)        = theta_m(kk,i) / (1.0_r8 + &
                                          Rv_over_Rd * tracers(index_qv,kk,i)) * exner(kk,i)
   phys_state(lchnk)%u(icol_p,k)        = ux(kk,i)
   phys_state(lchnk)%v(icol_p,k)        = uy(kk,i)
   phys_state(lchnk)%omega(icol_p,k)    = -rho_zz(kk,i)*zz(kk,i)*gravit*0.5_r8*(w(kk,i)+w(kk+1,i))   ! omega
   phys_state(lchnk)%pmiddry(icol_p,k) = pmiddry(kk,i)
   phys_state(lchnk)%pmid(icol_p,k)     = pmid(kk,i)

   if (use_gw_front .or. use_gw_front_igw) then
      frontgf_phys(icol_p, k, lchnk) = frontogenesisFunction(kk, i)
      frontga_phys(icol_p, k, lchnk) = frontogenesisAngle(kk, i)
   end if

   if (use_gw_movmtn_pbl) then
      vort4gw_phys(icol_p, k, lchnk) = vort4gw(kk, i)
   end if
end do
```

CAM-SIMA coupling layer

```
call dyn_debug_print(debugout_debug, subname // ' entered')

call init_shared_variables()

call dyn_exchange_constituent_states(direction='i', exchange=.true., conversion=.false.)

call dyn_debug_print(debugout_info, 'Setting physics state variables column by column')

! Set variables in the `physics_state` derived type column by column.
! This way, peak memory usage can be reduced.
do column_index = 1, ncells_solve
   call update_shared_variables(column_index)
   call set_physics_state_column(column_index)
end do

call set_physics_state_external()

call final_shared_variables()

call dyn_debug_print(debugout_debug, subname // ' completed')
```

MMM configurations

Along with the dycore, there are also plans to port over MMM physics schemes, with the hope to eventually have an MMM-derived physics suite that is applicable for global NWP and high-resolution simulations.

MMM configurations: MPAS physics suites

Parameterization	Scheme
Convection	New Tiedtke
Microphysics	WSM6
Land Surface	Noah
Boundary Layer	YSU
Surface Layer	Monin-Obukhov
Radiation (Long-wave)	RRTMG
Radiation (Short-wave)	RRTMG
Cloud Fraction for Radiation	Xu-Randall
Gravity Wave Drag by Orography	YSU

Parameterization	Scheme
Convection	Grell-Freitas
Microphysics	Thompson (non-aerosol aware)
Land Surface	Noah
Boundary Layer	MYNN
Surface Layer	MYNN
Radiation (Long-wave)	RRTMG
Radiation (Short-wave)	RRTMG
Cloud Fraction for Radiation	Xu-Randall
Gravity Wave Drag by Orography	YSU

MMM configurations CAM-SIMA suite

Parameterization	Scheme
Convection	New Tiedtke
Microphysics	TEMPO*
Land Surface	CTSM*
Boundary Layer	MYNN
Surface Layer	MYNN
Radiation (Long-wave)	RRTMG-P*
Radiation (Short-wave)	RRTMG-P*
Cloud Fraction for Radiation	Xu-Randall
Gravity Wave Drag by Orography	YSU

red = ported

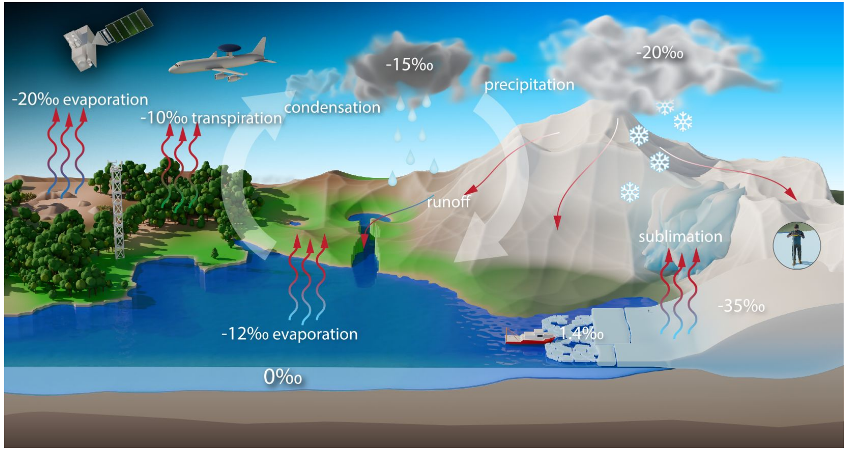
* = new scheme

Parameterization	Scheme
Convection	Grell-Freitas
Microphysics	Thompson (non-aerosol aware)
Land Surface	Noah
Boundary Layer	
Surface Layer	
Radiation (Long-wave)	
Radiation (Short-wave)	RRTMG
Cloud Fraction for Radiation	Xu-Randall
Gravity Wave Drag by Orography	YSU

Water tracer capabilities

The SCI-SWIM project is a major new effort to bring water-tracing capabilities into CAM-SIMA, for eventual use in a new water isotope-enabled version of CESM3.

This will set the stage for not only general hydrologic and paleoclimate research, but also for more accurate thermodynamics (e.g. knowing the temperature of rain as it falls into the ocean).



User survey can be found here (~15 min):

https://docs.google.com/forms/d/e/1FAIpQLSe0G82Ne7wTMqCMJ7dSS6wBhmDni6xYsegaRvq9HGKL2OPnqw/viewform

Software engineering updates - runtime config.

There are several software engineering improvements in CAM-SIMA relative to CAM that could have a major impact on users. One major shift is the push to more runtime configuration, such as:

1. Setting the number of tasks
2. Setting the number of vertical levels
3. Setting the number of constituents (including advected species)
4. Defining the set of known chemical reactions (provided by MUSICA)
5. Setting physics parameters that used to be hard-coded.

We hope to continue moving in this direction, with the goal to reduce the need to build the model to only when core source code modifications are made.

Software engineering updates – OOP and FP

CAM-SIMA is also moving to programming paradigms beyond just the regular procedural programming methods used in CAM. In particular we are trying to use an object-oriented approach with infrastructure, and a functional approach for the actual science calculations.

File reading using an object

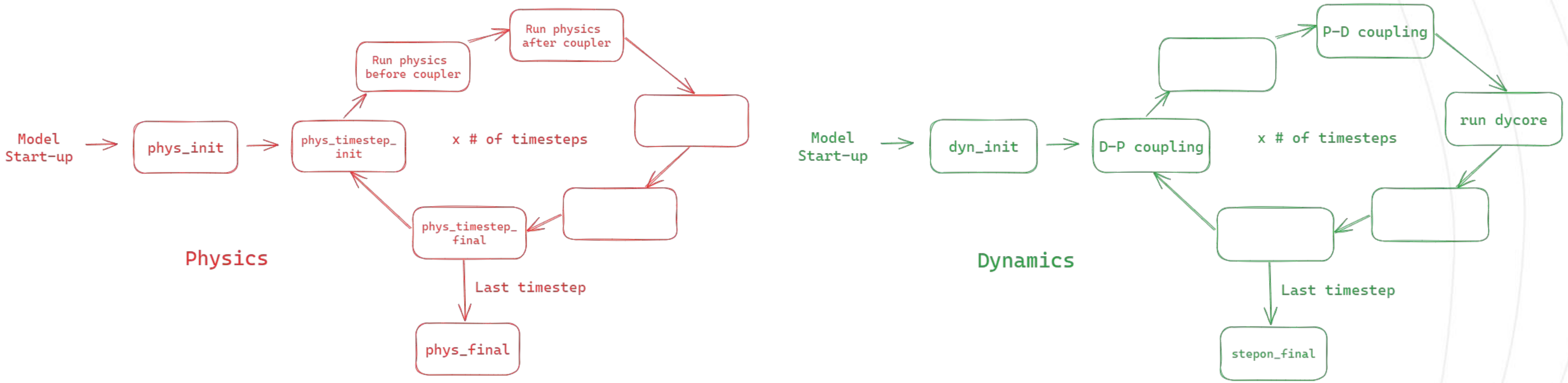
```
file_reader => create_netcdf_reader_t()

! Open the solar irradiance data file
call file_reader%open_file(irrad_file_path, errmsg, errflg)
if (errflg /= 0) then
   errmsg = subname // errmsg
   return
end if

! Read the wavelengths variable
call file_reader%get_var('wavelength', lambda, errmsg, errflg)
```

Science calculation using a pure function

```
pure elemental function calc_obukhov_length(thvs, ustar, g, karman, kbfs) result(obukhov_length)
    ! Stull, Roland B. An Introduction to Boundary Layer Meteorology. Springer Kluwer Academic Publishers, 1988. Print.
    ! DOI: https://doi.org/10.1007/978-94-009-3027-8
    ! Equation 5.7c, page 181
    ! \frac{-\theta*u_*^3}{g*k*\overline{(w' \theta_v')}_s} = frac{-\theta*u_*^3}{g*k*kbfs}

    real(kind_phys), intent(in)  :: thvs            ! virtual potential temperature at surface [ K       ]
    real(kind_phys), intent(in)  :: ustar           ! Surface friction velocity                [ m s-1   ]
    real(kind_phys), intent(in)  :: g               ! acceleration of gravity                  [ m s-2   ]
    real(kind_phys), intent(in)  :: karman          ! Von Karman's constant (unitless)
    real(kind_phys), intent(in)  :: kbfs            ! surface kinematic buoyancy flux          [ m K s-1 ]

    real(kind_phys)              :: obukhov_length  ! Obukhov length                           [ m       ]

    ! Added sign(...) term to prevent division by 0 and using the fact that
    ! `kbfs = \overline{(w' \theta_v')}_s`
    obukhov_length = -thvs * ustar**3 /                              &
                     (g*karman*(kbfs + sign(1.e-10_kind_phys,kbfs)))
end function calc_obukhov_length
```

Software engineering updates - Design docs

One type of documentation that is lacking in CAM is design documentation, e.g. workflow diagrams of CAM itself. We have been trying to develop this documentation for CAM-SIMA, which we have found helpful for new developers or SEs working with the model.



https://escomp.github.io/CAM-SIMA-docs/

Software engineering updates – reduced Fortran

Another development shift we are trying to do is to move certain model system actions, such as the defining of variables, away from Fortran and to other languages, including XML. This will hopefully help provide future flexibility in CAM-SIMA, and focus the Fortran on the core calculations (e.g. the dycore and physics).

CAM physics_types.F90

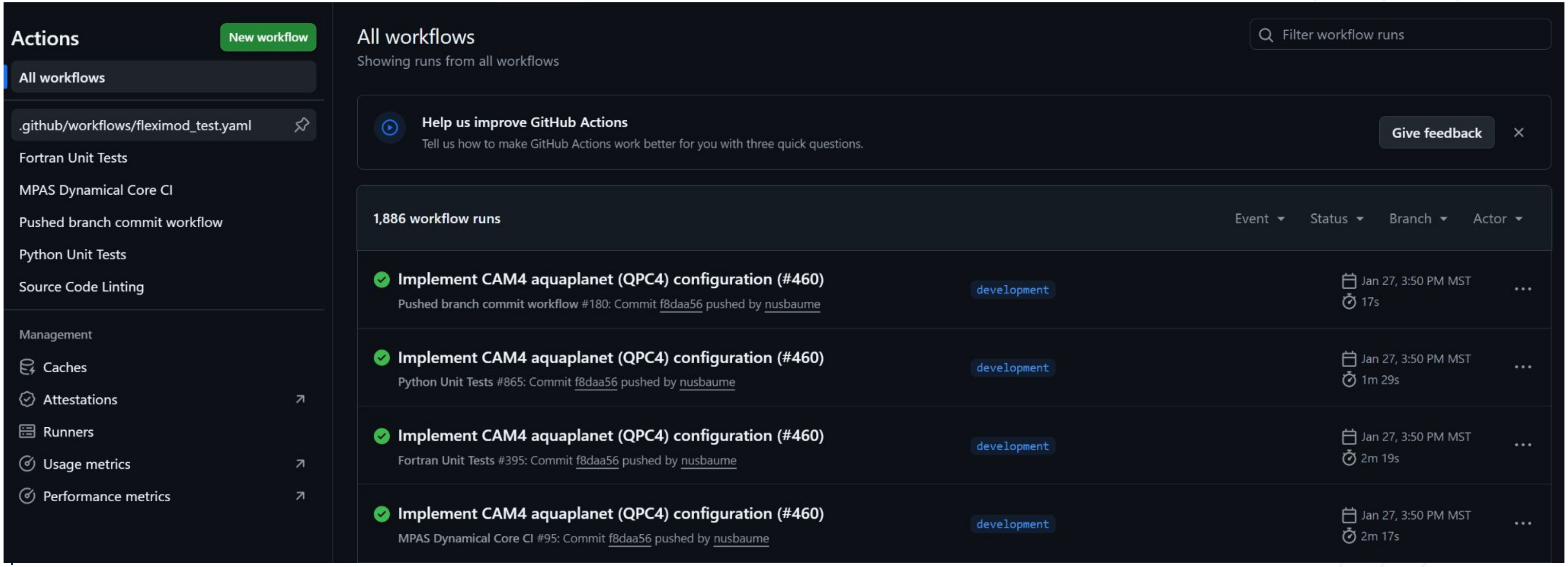
```
real(r8), dimension(:,:),allocatable   :: &
     s,                    &! heating rate (J/kg/s)
     u,                    &! u momentum tendency (m/s/s)
     v                      ! v momentum tendency (m/s/s)
```

CAM-SIMA registry.xml

```
<!-- State tendencies -->
<variable local_name="dTdt_total"
          standard_name="tendency_of_air_temperature_due_to_model_physics"
          units="K s-1" type="real" kind="kind_phys"
          allocatable="pointer">
  <long_name>Change in temperature from a parameterization</long_name>
  <dimensions>horizontal_dimension vertical_layer_dimension</dimensions>
  <ic_file_input_names>dTdt tend_dtdt</ic_file_input_names>
</variable>
```

Software engineering updates

Finally, Automated testing has been implemented in the CAM-SIMA software ecosystem using Github Actions. This includes Python and Fortran unit testing and static code analysis (pylint and fortitude).



Software engineering updates - Future work

Possible future work that has come up with regards to CAM-SIMA include:

1. **Mixed-precision capabilities**

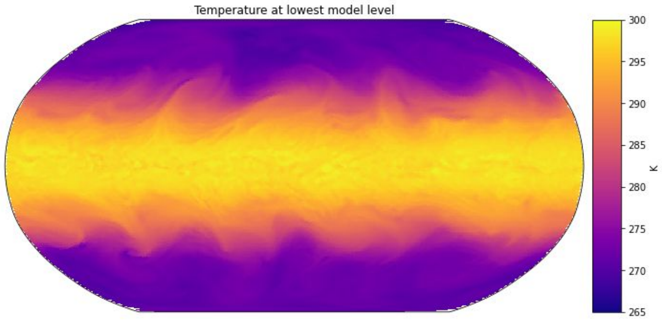
2. **Continued GPU-ization efforts**

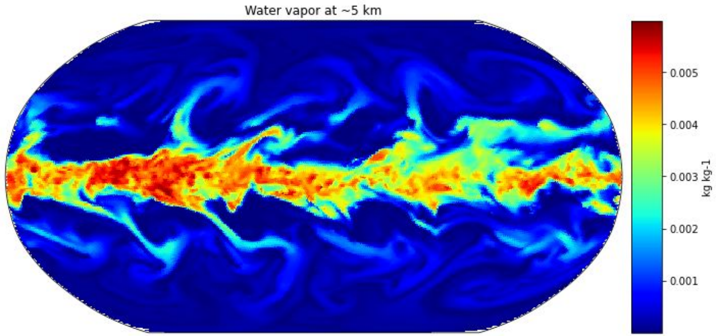
3. **The addition of JAX-accelerated python**
   a. Including Fortran codes translated by AI.

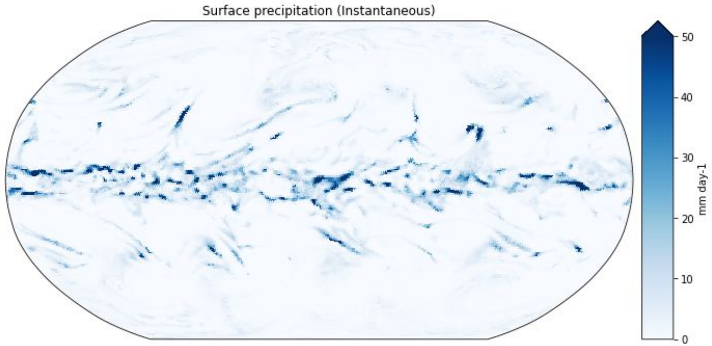
Thanks for listening!

Any questions?

Some CAM-SIMA-QPC4 Plots!







Converting CAM4 physics: why?

Separating science from “boilerplate”: convect_shallow example

Cleaning up physics code for CCPP

Subroutine signature for hack_convect_shallow scheme

```
!> \section arg_table_hack_convect_shallow_run Argument Table
!! \htmlinclude hack_convect_shallow_run.html
  subroutine hack_convect_shallow_run( &
    ncol, pver, pcnst, &
    iulog, const_props, &
    ztodt, &
    pmid, pmiddry, &
    pdel, pdeldry, rpdel, rpdeldry, zm, &
    qpert_in, &
    phis, pblh, &
    t, &
    q, & ! ... below are output arguments:
    dq, &
    qc_sh, &
    cmfdt, cmfmc_sh, cmfdqr, cmfsl, cmflq, &
    precc, cnt_sh, cnb_sh, &
    icwmr, &
    rliq_sh, &
    scheme_name, flx_cnd, &
    errmsg, errflg)
```