Introduction

While CAM provides a grid for use by physics parameterizations, there are phenomena for which a smaller scale is desirable. In other cases, an ensemble approach within a single grid column can improve parameterization performance. The CAM subcolumn infrastructure provides the ability to create fields with several elements within a single grid column. It supports a variable number of subcolumns per grid column. While some physics parameterizations are currently run on a sub-grid scale (e.g., radiation, SPCAM), the new infrastructure provides a common subcolumn infrastructure for use by multiple parameterizations.

Please note that these changes are transparent to existing parameterizations. No code changes are required in order to work within the new infrastructure.

This poster provides an introduction to infrastructure changes made to support subcolumns, examples of subcolumn usage in CAM physics, and a brief description of how subcolumns are created and managed.

**IF YOU PLAN ON IMPLEMENTING SUBCOLUMNS: CONTACT US – THIS IS A WORK IN PROGRESS**

Infrastructure changes on CAM development trunk

- **Dynamic state/tend/ptend changes**
  - `pcols` is still maximum number of grid columns / chunk `state%ncol` is still the number of columns to loop over, but may be larger than `pcols` if using subcolumns `psubcols` is the maximum number of subcolumns = `1` for grid `state%psubcols` is the maximum number of total columns whether using grid or subcolumns = `pcols*psubcols` (replaces `pcols` in a lot of places)

- **New Fields to support subcolumns**
  - `pbuf` structure:
    - `grid_flag_utils.F90`:
      - `col_type_flag`
      - `grid_types_flag`
      - Only if `pbuf` structure:
        - `pbuf_set_field`
        - `pbuf_get_field`
        - `pbuf_add_field`
        - Generates flags for use with `subcolumn` schemes API

- **Physics Buffer (pbuf) changes**
  - The physics buffer holds fields for use by the physics package.

- **New Parameters to Identify grid/subcolumns**
  - `col_type` int 0=grid, 1=subcolumns
  - `grid_type` int[bit_field_kind] each bit is turned on/off to indicate which field(s) are required

- **pbuf_add_field**: Adds grid and/or subcolumn fields to `pbuf` based on optional `grid_types_flag` – If not present adds grid field only; if present adds field(s) specified by `grid_types_flag`

- **pbuf_set_field**: Sets grid and/or subcolumn fields in `pbuf` based on optional `grid_types_flag` – If not present sets grid field only; if present sets the field(s) specified by the `grid_types_flag`

- **pbuf_get_field**: Gets grid or subcolumn field based on optional `col_type_flag` – If not present then grid field is returned.

- **grid_flag_utils.F90**: new module which contains routines to manipulate grid_type variables

Microphysics and Subcolumns

**Code snippets from tphysbc**

(new code colorized, * indicates function description at right)

if (use_subcol_microp) then
  ! Declare subcolumn structures
  state_sc%psubcols = state%psubcols * psubcols
  call physics_state_alloc(state_sc, lchnk, state_sc%psubcols)
  call physics_tend_alloc(tend_sc, state_sc%psubcols)
  ! Generate subcolumns using the requested scheme
  call subcol_gen(state, tend, state_sc, tend_sc, pbuf) !*
  ! Average the subcolumn ptend for use in gridded update
  call subcol_ptend_avg(ptend_sc, state_sc, tend_sc, pbuf) !*
  call check_energy_chng(state, tend_sc, microp_tend, nstep, &
    ztolt, zero_sc, prev_str, snow_str, zero_sc) !*
  call physics_state_dealloc(state, tend_sc)
  call physics_tend_dealloc(tend_sc)
  else
    ! Subcolumn driver_tend (use_subcol_microp, state%psubcols, &
    ! state_sc, ptend_sc, ztolt, pbuf, cmelqj)
    ! Generate fields for use with physics_state
tend_sc
    ! Average the subcolumn ptend for use in gridded update
    call subcol_ptend_avg(ptend_sc, state_sc, tend_sc, pbuf) !*
  ! Generate subcolumn structures
  call subcol_register(name_list, namelist, subcol_scheme)
  ! Interface to physics state and the physics tend structures with
  ! subcolumn registers/parameter names_variables
  ! Code fragments which implement a
  ! physics scheme API still undergoing development. The
  ! 'subcol_scheme' namelist variable.
  ! The subcolumn scheme API is still undergoing development. The
  ! interfaces below are subject to change.
  ! Public interface functions which implement a subcolumn scheme
  ! subcol_register: Read the scheme name from namelist, initialize any
  ! scheme-global variables, and add scheme fields as necessary (typically
  ! using pbuf_add_field).
  ! subcol_init: Initialize any variables or fields specific to the active scheme
  ! (typically by calling pbuf_set_field).
  ! subcol_gen: Generate subfields from grid fields. Create copies of the
  ! physics state and the physics tend structures with subcolumns.
  ! subcol_field_avg: Average a field defined on subcolumns back into an
  ! averaged field for the grid cell. Averaging may be weighted based on
  ! scheme variables.
  ! subcol_ptend_avg: Average a physics tendency (ptend) defined on
  ! subcolumns to an averaged ptend for the grid cell.
  ! subcol_outfld: Unpack a subcolumn field before calling outfld.
  ! Other physics parameter packages using subcolumns
  ! NB: These functions work on structures which all have subcolumns or which all have no subcolumns. Physics parameterizations using
  ! physics_update do not have to make any changes.
  ! physics_update: Update the state and or tendency structure with the
  ! parameterization tendencies.
  ! check_energy_chng: Check that the energy and water change matches
  ! the boundary fluxes.

Summary

A subcolumn infrastructure has been developed for use by CAM physics parameterizations. This infrastructure:
- is flexible and extensible and supports a variable number of subcolumns
- introduces standard methods for building and working with subcolumns
- dynamically allocates fields storage to optimize memory use
- can be used to support either sub-grid scale simulation or column ensembles
- does not require any code changes to existing parameterizations
- allows parameterizations to use the same subcolumn field data
- supports multiple ‘schemes’ for subcolumn generation and data averaging
- supports output of subcolumn data to CAM history files

While development is currently underway to implement subcolumn usage in CAM microphysics, the infrastructure has been committed to the CAM development trunk as of cam5_3_03. Note that this infrastructure is not in CESM 1.2 (except for the dynamic allocation of state/tend/ptend which was committed in cam5_2_09).

Acknowledgements

We gratefully acknowledge useful discussions with William Collins, Steve Ghan, Phil Rasch, Vincent Larson, Cecile Hannay, Kate Thayer-Calder, Charles Bardeen, Chris Fischer, Joe McInerney, Sean Santos, John Truesdale, Jim Edwards, Mariana Vertenstein, and Francis Vitt.

This work has been supported by the National Science Foundation and by the US DOE Advanced Scientific Computing Research and Biological and Environmental Research Programs.