

Computing New Diagnostic Fields in MPAS-Atmosphere Simulations

As the number of people using MPAS-Atmosphere increased to even a very modest number, it was recognized that we needed some structured way of introducing the computation of new diagnostics

As the number of people using MPAS-Atmosphere increased to even a very modest number, it was recognized that we needed some structured way of introducing the computation of new diagnostics

We wanted a framework that could handle:

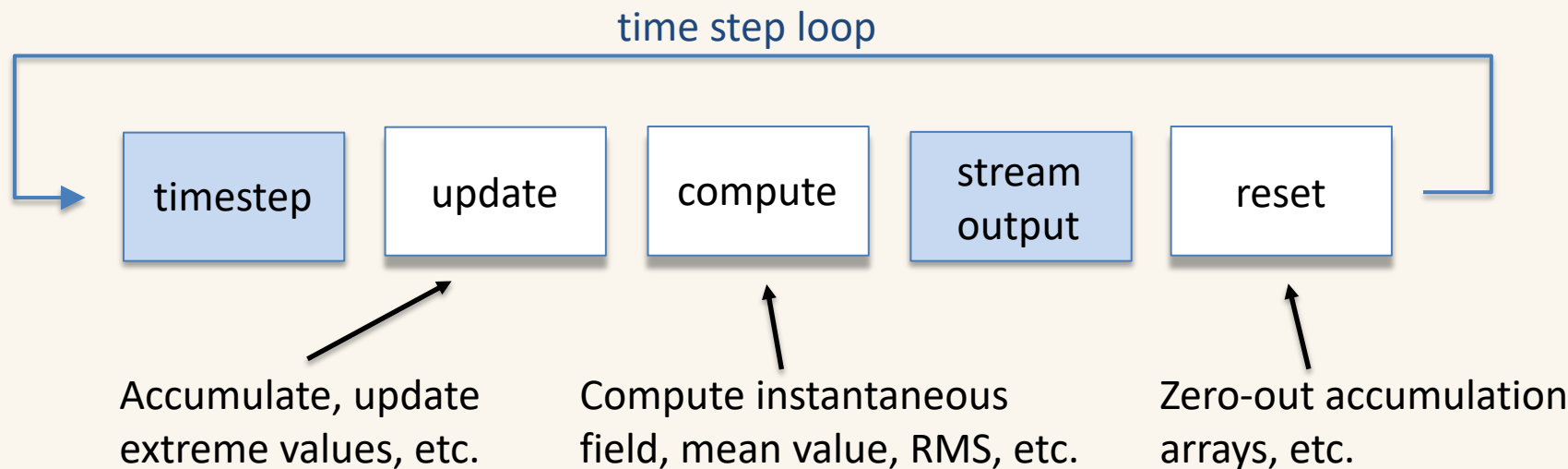
- Instantaneous diagnostics
- Cumulative diagnostics
- Extreme value diagnostics
- *Or anything that looked like the above from a computational standpoint!*

Broadly, there are five stages in the computation of diagnostics like those types listed on the previous slide:

1. **Setup** – Pre-allocate arrays that will be needed by the diagnostic during the simulation, initialize values, etc.
2. **Update** – Accumulate values, update extrema, etc.
3. **Compute** – Divide by the accumulation period to get mean values, compute instantaneous fields, etc.
4. **Reset** – After writing the diagnostic, zero-out accumulation arrays, etc.
5. **Cleanup** – Deallocate any memory that was allocated by the diagnostic

Diagnostics framework

Other than *Setup* and *Cleanup*, how do these phases fit into the sequence of a model timestep?

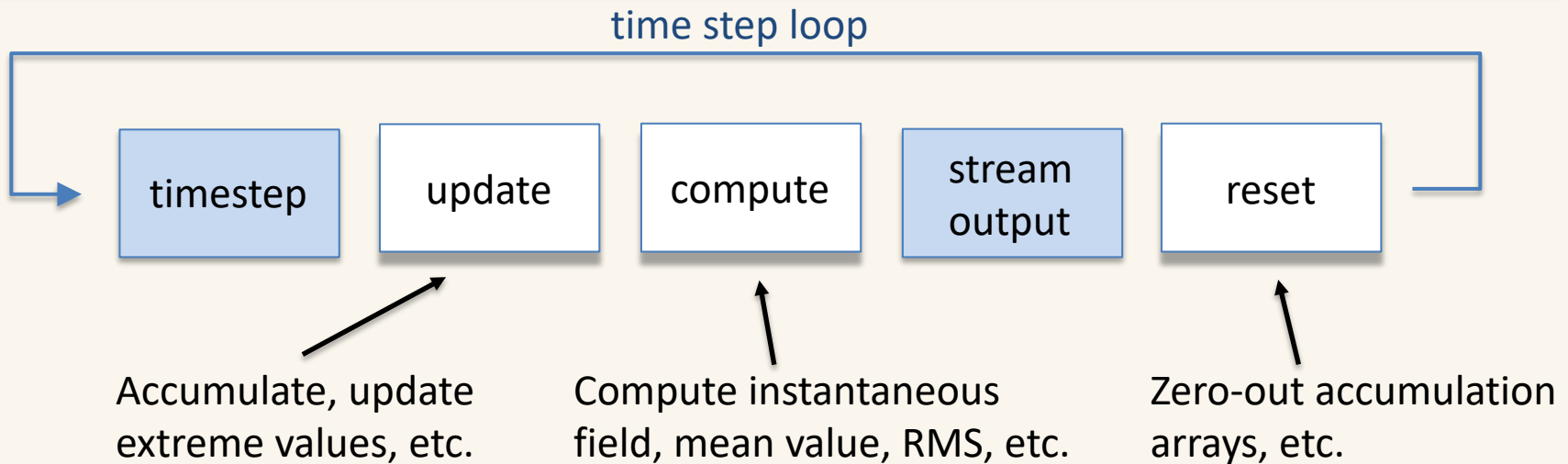


Critically, we need some way for a diagnostic to determine whether it will be written out in a given timestep!

Diagnostics utilities

Along with the diagnostics framework, a utility module has been written that will tell a diagnostic:

- Whether its field will be written in a given timestep
 - Generally called from the *Compute* phase
- How many output streams include its field



Code organization

In terms of code structure, where do we add diagnostics?

- The idea is that all diagnostics should be implemented as self-contained modules
- All diagnostics modules should reside in the same place

```
MPAS-Model/  
  src/  
    core_atmosphere/  
      diagnostics/  
      dynamics/  
      inc/  
      physics/  
      Registry.xml  
      utils/
```

Add new diagnostics in this directory

The main Registry.xml file will include “sub-Registry” sections from the diagnostics directory

How to add a new diagnostics?

1. Define new namelist options and fields in a new `Registry_<your_diagnostic>.xml` file. Add a `#include` statement for this new Registry section in the `Registry_diagnostics.xml` file.
2. Create a new module for the diagnostic.
3. Add calls to your diagnostic's *setup*, *update*, *compute*, *reset*, and *cleanup* routines in the main diagnostic driver.
4. Update the `Makefile` to compile your new diagnostic module.

The `README` file in the `diagnostics/` subdirectory describes the step-by-step process to adding a new diagnostic

Example: Maximum SW downward flux

Working in the `src/core_atmosphere/diagnostics/` directory:

1. Define the maximum SWDOWN field in a new `Registry_swdown.xml` file

```
<var_struct name="diag_physics" time_levs="1">
  <var name="gsw_max" type="real"
    dimensions="nCells Time"
    units="W m^{-2}"
    description="Maximum surface SW radiation flux"/>
</var_struct>
```

2. Include `Registry_swdown.xml` in `Registry_diagnostics.xml`

```
<!-- SW radiation diagnostics -->
#include "Registry_swdown.xml"
```

Example: Maximum SW downward flux

Working in the `src/core_atmosphere/diagnostics/` directory:

3. Write the *setup*, *update*, and *reset* routines for the diagnostic:

```
subroutine swdown_setup(all_pools)

  type (MPAS_pool_type), pointer :: all_pools

  type (MPAS_pool_type), pointer :: diag_physics

  call mpas_pool_get_subpool(all_pools, 'diag_physics', diag_physics)
  call mpas_pool_get_array(diag_physics, 'gsw', gsw)
  call mpas_pool_get_array(diag_physics, 'gsw_max', gsw_max)

  gsw_max(:) = 0.0

end subroutine swdown_setup
```

Not shown in the code above is the declaration of `gsw` and `gsw_max` as module variables...

Example: Maximum SW downward flux

Working in the `src/core_atmosphere/diagnostics/` directory:

3. Write the *setup*, *update*, and *reset* routines for the diagnostic:

```
subroutine sdown_update()  
  
    gsw_max(:) = max(gsw_max(:), gsw(:))  
  
end subroutine sdown_update
```

Example: Maximum SW downward flux

Working in the `src/core_atmosphere/diagnostics/` directory:

3. Write the *setup*, *update*, and *reset* routines for the diagnostic:

```
subroutine sdown_reset()  
  
    use mpas_atm_diagnostics_utils, only : mpas_field_will_be_written  
  
    if (mpas_field_will_be_written('gsw_max')) then  
        gsw_max(:) = 0.0  
    end if  
  
end subroutine sdown_reset
```

Example: Maximum SW downward flux

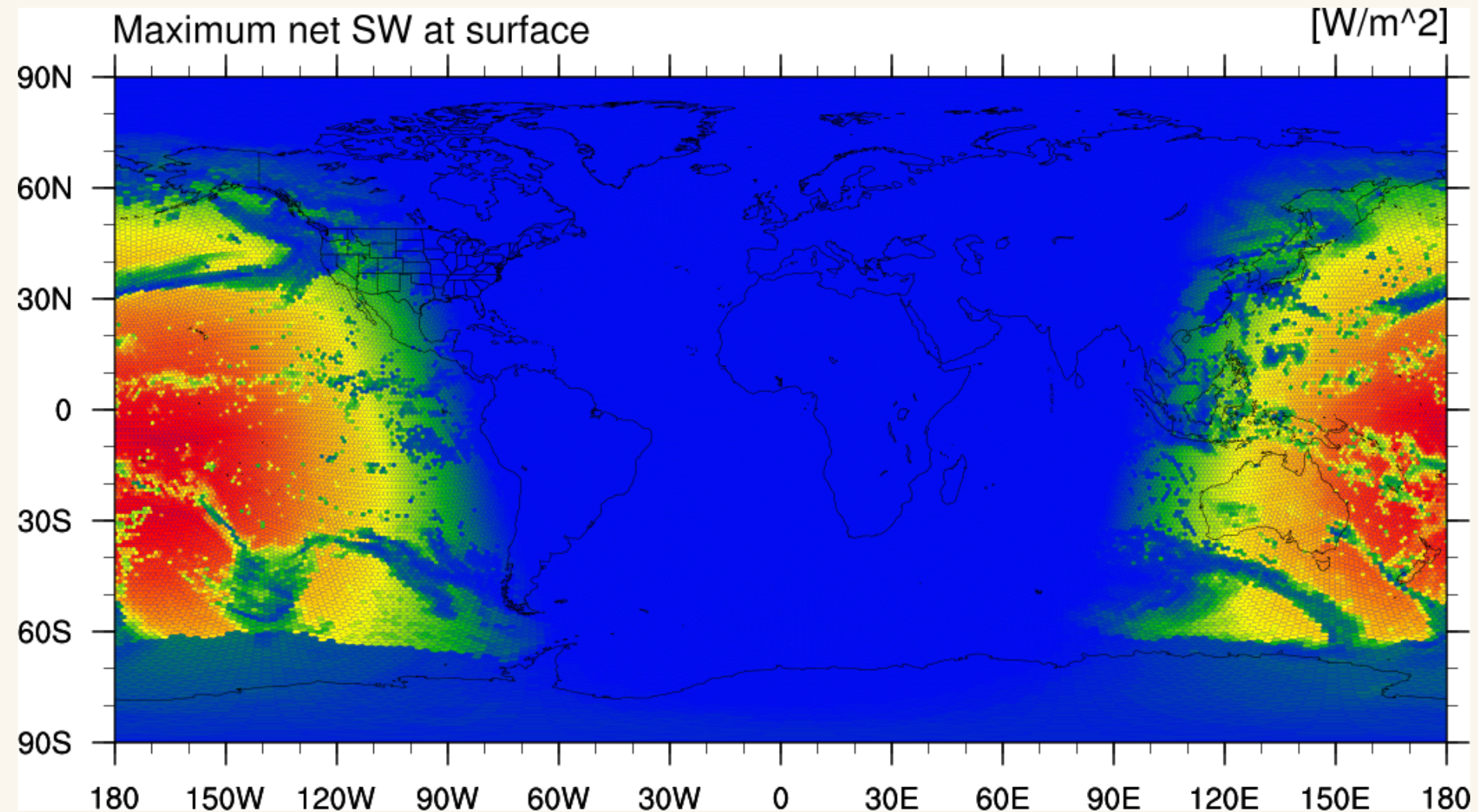
After adding a couple of lines to the Makefile, we're ready to compile MPAS-Atmosphere and try out our new diagnostic

We'll test it by:

- Outputting the gsw_max field every 60 minutes
- Outputting the gsw_max field every 1440 minutes

Our test simulation will start at 2010-10-23 at 00 UTC, and we'll look at the output that we get at 2010-10-24 at 00 UTC...

Example: 60-minute output interval



Example: 1440-minute output interval

