Adding Passives Tracers to MPAS-Atmosphere Simulations

Michael G. Duda
NCAR/MMM

This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the National Science Foundation under Cooperative Agreement No. 1852977.

MPAS
Model for Prediction Across Scales

Introduction

With the background about MPAS software from the "MPAS Software" talk, adding a passive tracer to an MPAS-Atmosphere simulation is *relatively* easy!

There are three steps to accomplish this:

1. Define initial conditions for the tracer when running the `init_atmosphere_model` program
2. Define the tracer in the model itself
3. If you'd like, add sources/sinks

Setting initial conditions

When setting initial conditions for the passive tracer (let's call it radon), we need to first define this field in the `Registry.xml` file of the *init_atmosphere* core

- All scalars (whether moisture or passive) are defined in an array of variables called "scalars"

```
<var_array name="scalars" type="real"
           dimensions="nVertLevels nCells Time">
      <var name="qv"    array_group="moist" units="kg kg^{-1}"/>
      <var name="qc"    array_group="moist" units="kg kg^{-1}"/>
      <var name="qr"    array_group="moist" units="kg kg^{-1}"/>
      <var name="radon" array_group="passive" units="kg kg^{-1}"/>
</var_array>
```

In the model, the `array_group` is used to identify, e.g., moisture species that contribute air density; instead of "passive", we could use any name other than "moist"

MPAS
Model for Prediction Across Scales

Setting initial conditions

Inside the *init_atmosphere* core, the `mpas_init_atm_cases.F` code is responsible for defining initial conditions

- We can add a new subroutine here to initialize “radon”

```
! Subroutine argument
type (mpas_pool_type), intent(inout) :: state

! Local variables
integer, pointer :: index_radon
real (kind=RKIND), dimension(:,:,:), pointer :: scalars

call mpas_pool_get_array(state, ‘scalars’, scalars)
call mpas_pool_get_dimension(state, 'index_radon', index_radon)
```

Using pools, we can access the 3-d “scalars” array that was defined in the `Registry.xml` file, and we can figure out which index in this array contains “radon” (rather than, e.g., “qv” or “qc”)

Setting initial conditions

Now, using whatever clever logic we would like, we can set the "radon" component of the "scalars" array:

```
scalars(index_radon, :, :) = 0.12345
```

Note: scalars is dimensioned (1:num_scalars, 1:nVertLevels, 1:nCells).

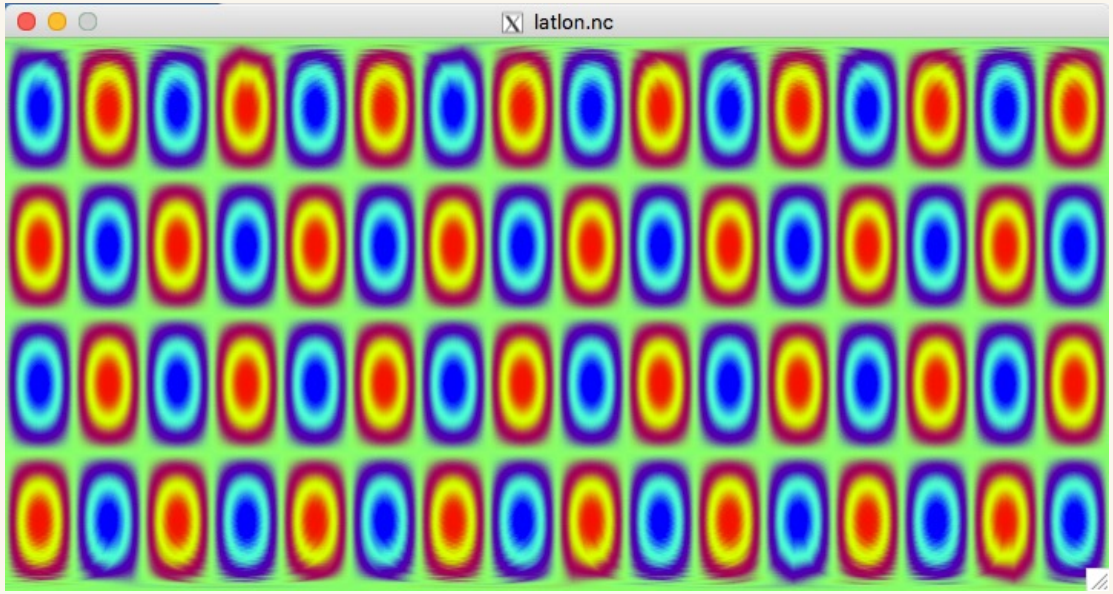
In general, we can initialize the tracer in one of many ways:

- Horizontally interpolate from some other dataset
- Set the tracer based on geographic location, (lat, lon)
- Set the tracer based on terrain height or land use category
- Etc.

Setting initial conditions

Once we've added code to set the values in `scalars(index_radon,:,:)` we will need to re-compile the *init_atmosphere* core

Creating initial conditions as usual, we should now have "radon" in our initial conditions file in addition to "qv", "qc", and "qr"!



Left: A simple sinusoidal pattern is a nice way to test a passive tracer

Adding the tracer to the model

We have produced initial conditions for the "radon" tracer, but without changes in the `atmosphere_model` program, the model will simply ignore "radon" in the initial conditions file

- We need to edit the `Registry.xml` file for the *atmosphere* core, too!

```
<var_array name="scalars" type="real"
           dimensions="nVertLevels nCells Time">

    <var name="qv" array_group="moist" units="kg kg^{-1}"
         description="Water vapor mixing ratio"/>

    <var name="qc" array_group="moist" units="kg kg^{-1}"
         description="Cloud water mixing ratio"
         packages="bl_mynn_in;bl_ysu_in;cu_tiedtke_in;mp_kessler_in"/>

    <var name="radon" array_group="passive" units="kg kg^{-1}"
         description="Radon mixing ratio"/>
</var_array>
```

Above: A subset of the "scalars" variable array in the atmosphere core Registry.xml file. The model definition of "scalars" is a little more complicated…

Adding the tracer to the model

Note several new things:

- Individual scalar constituents have *packages*
- The units of most scalars are *kg kg^{-1}*

```
<var_array name="scalars" type="real"
           dimensions="nVertLevels nCells Time">

    <var name="qv" array_group="moist" units="kg kg^{-1}"
         description="Water vapor mixing ratio"/>

    <var name="qc" array_group="moist" units="kg kg^{-1}"
         description="Cloud water mixing ratio"
         packages="bl_mynn_in;bl_ysu_in;cu_tiedtke_in;mp_kessler_in"/>

    <var name=”radon" array_group="passive" units="kg kg^{-1}"
         description=”Radon mixing ratio"/>
</var_array>
```

Above: A subset of the “scalars” variable array in the atmosphere core Registry.xml file. The model definition of “scalars” is a little more complicated…

Adding the tracer to the model

Based on what we did in the `init_atmosphere` core, you may be tempted to think we are done editing the `Registry.xml` file

It turns out that in the model itself, we also need to define an array to hold the tendency for the new tracer

- This is done in a variable array named `scalars_tend`

```
<!-- scalar tendencies -->
<var_array name="scalars_tend" type="real"
           dimensions="nVertLevels nCells Time">

    <!- we will see on the next slide what goes here... -->

</var_array>
```

Adding the tracer to the model

One might expect that the units for the radon tendency would be kg kg^{-1} s^{-1}…

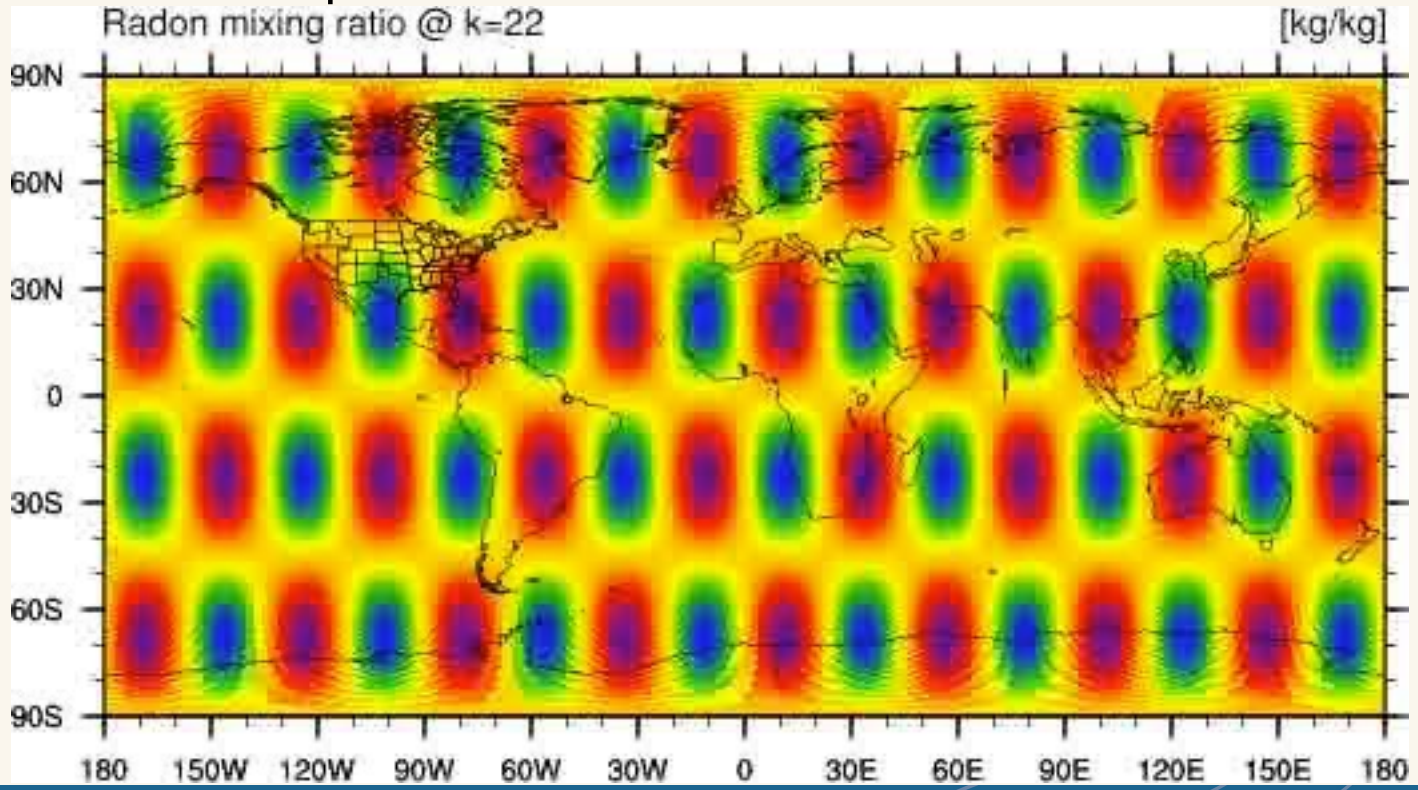
```
<var_array name="scalars_tend" type="real"
           dimensions="nVertLevels nCells Time">
     <var name="tend_qv" name_in_code="qv" array_group="moist"
          units="kg m^{-3} s^{-1}"
          description="Tendency of water vapor mass per unit volume
divided by d(zeta)/dz"/>

      <var name="tend_radon" name_in_code=”radon" array_group="passive"
           units="kg m^{-3} s^{-1}"
           description="Tendency of radon mass per unit volume divided
by d(zeta)/dz" />
</var_array>
```

… but what we actually need is for the units of the radon tendency to be (kg m^{-3} s^{-1}) / ($d\zeta/dz$)

Considerations for sources and sinks

If we don't specify sources/sinks for our new passive tracer, it will simply be transported by MPAS, and it will have no impact on the rest of the simulation

- I.e., all other prognostic fields should be the same whether the passive tracer is present or not!



Considerations for sources and sinks

But, we can specify sources/sinks if we like…

- As with the tracer and its tendency, we will need to define the source/sink field in the `Registry.xml` file of the model

```
<var name="radonprod" type="real”
     dimensions="nVertLevels nCells Time”
     units="kg m^{-2} s^{-1}"
     description="Radon gas areal production rate" />
```

The source/sink field can be a regular variable in the Registry.xml file – there's no need to add it to any special place like a scalars variable array

The units of the source/sink can be whatever is needed. We will only need to convert these units to kg m^{-3} s^{-1} when applying the source/sink!

Considerations for sources and sinks

Where can we set the sources/sinks for our passive scalar?

- If time-invariant, the simplest would be to specify them in the initialization routine for MPAS, `atm_core_init(…)`

```
call mpas_pool_get_array(tend_physics, 'radonprod', radonprod)
call mpas_pool_get_array(sfc_input, 'ivgtyp', ivgtyp)

radonprod(:,:) = 0.0_RKIND
do iCell=1,nCells
   ! Assume that vegetation category 17 is water
   ! (true for MODIS land use, not true for USGS)
   if (ivgtyp(iCell) == 17) then
      ! 1 g/m^2/day destruction over water
      radonprod(1,iCell) = -0.001 / 86400.0
   else
      ! 2 g/m^2/day production over land
      radonprod(1,iCell) = 0.002 / 86400.0
   end if
end do
```

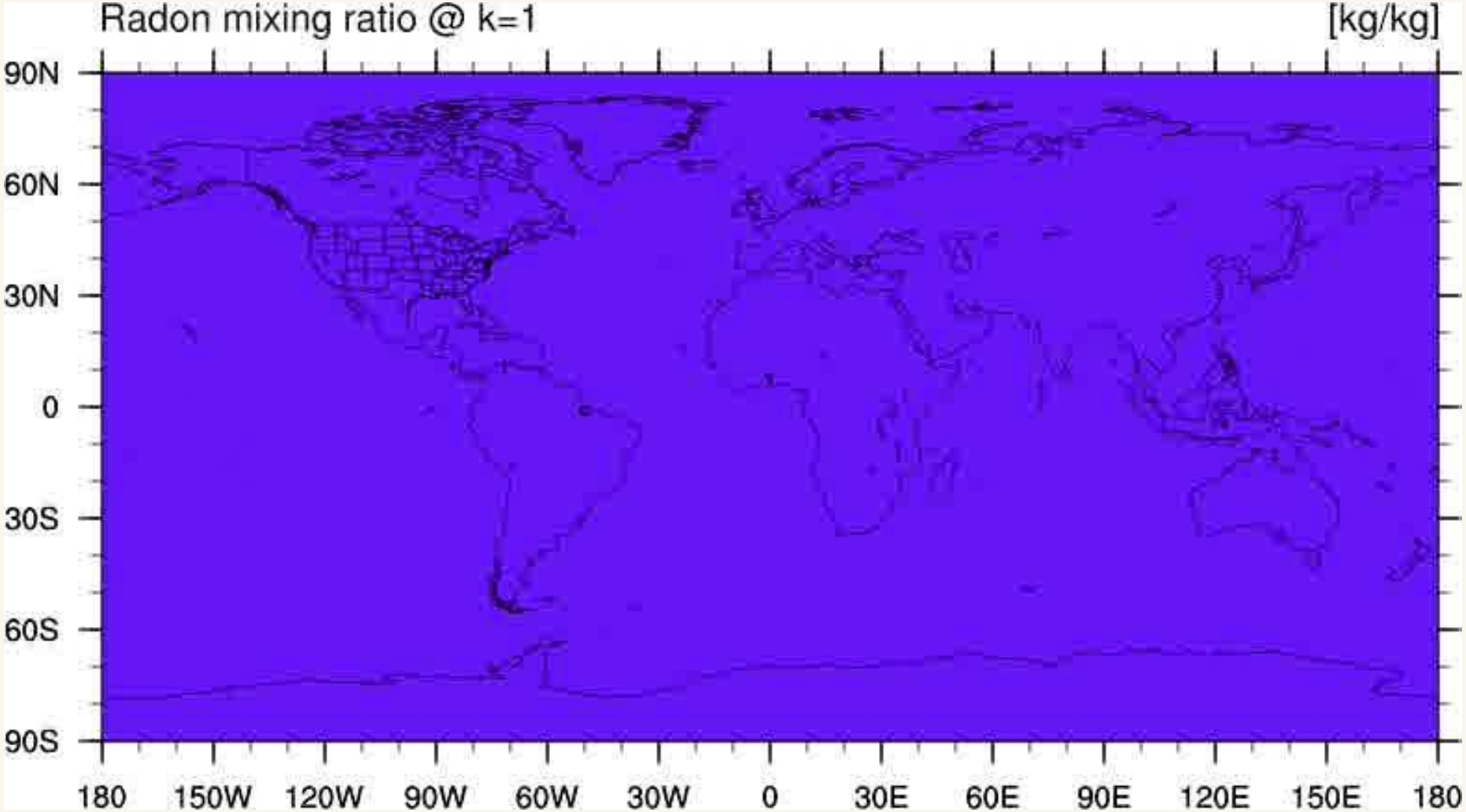
Considerations for sources and sinks

Regardless of how we get the source/sink field, we need to set the tracer tendency based on this field in the `physics_get_tend(…)` routine

```
!
! Account for sources and sinks for radon (in the radonprod array)
! in the tendency for radon. If radonprod has units of kg/m^2/s,
! we need to divide by layer thickness and then by d(zeta)/dz
! to get the proper units of kg/m^3/s divided by d(zeta)/dz for
! the tendency.
!
do i = 1, nCellsSolve
   do k = 1, nVertLevels
      tend_scalars(index_radon,k,i) = tend_scalars(index_radon,k,i) &
                + radonprod(k,i) / (zgrid(k+1,i) - zgrid(k,i)) / zz(k,i)
   end do
end do
```

Considerations for sources and sinks

All of this work pays off, though!



Summary

In summary, there are several pieces to adding a new passive tracer in MPAS-Atmosphere:

1. Define the tracer in the *init_atmosphere* core's `Registry.xml` file
2. Provide initial conditions for the tracer
3. Define the tracer in the *atmosphere* core's `Registry.xml` file
4. Define the tracer tendency in the atmosphere core's `Registry.xml` file
5. Optionally, define sources/sinks for the tracer
   - These may be programmatically specified, or read from a periodic input stream