Modifying Code in the CLM

CLM Tutorial 2016
Danica Lombardozzi, Ahmed Tawfik,
Bill Sacks, Marysa Lague

NSF NCAR NATIONAL CENTER FOR ATMOSPHERIC RESEARCH U.S. DEPARTMENT OF ENERGY Office of Science

Why might you modify the code?

- Improve process representation based on new scientific findings
- Introduce a new concept
- Test the sensitivity of an existing representation
- And more…

Two methods for modifying code

Method 1: Modify code in-place

- Requires your own copy of the code: can't do this if you're working from the shared tutorial code, for example
- Best for changes that have some of these characteristics:
  - apply to many cases
  - long-term
  - incremental changes towards a final solution
  - apply to many files
- Allow you to leverage the power and convenience of a version control system

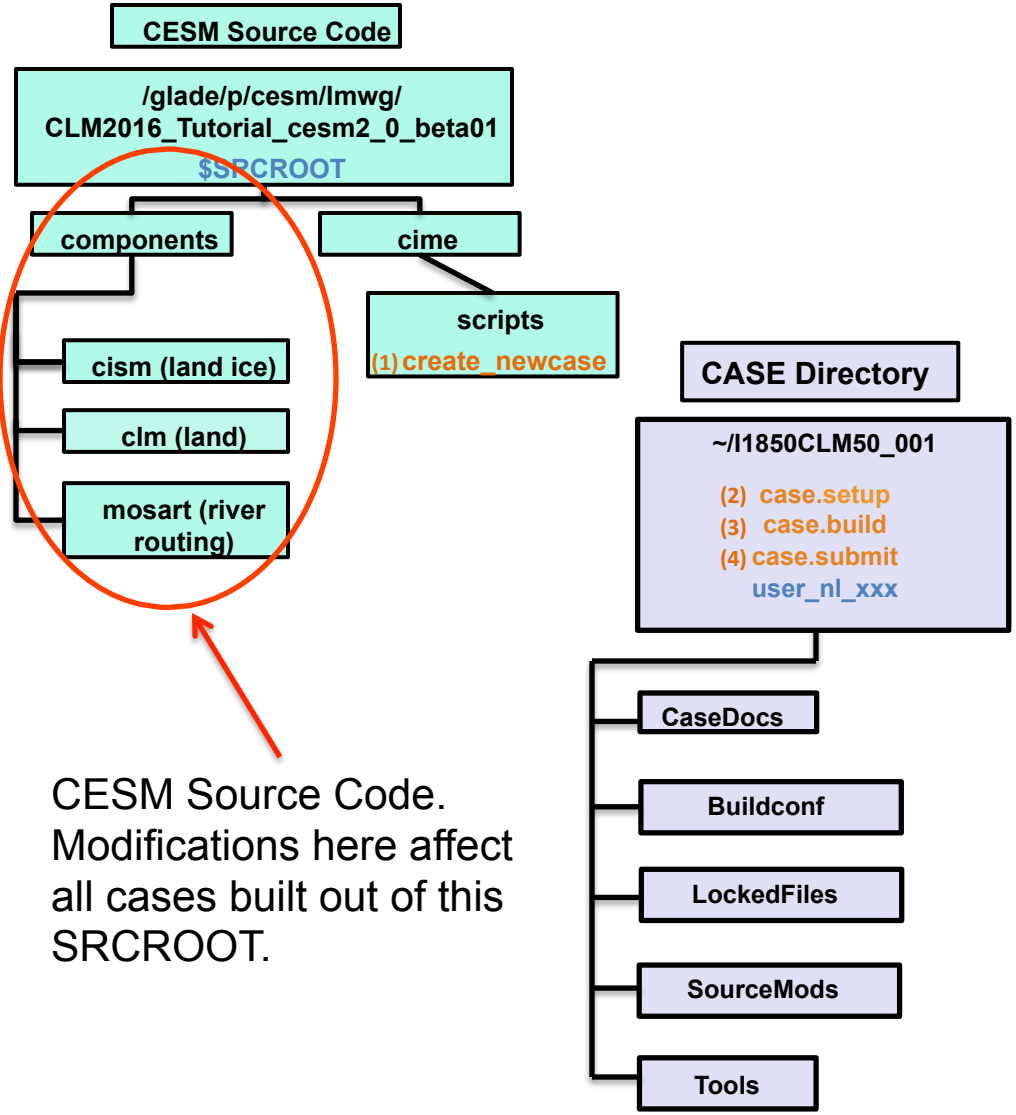
SUBVERSION®

git

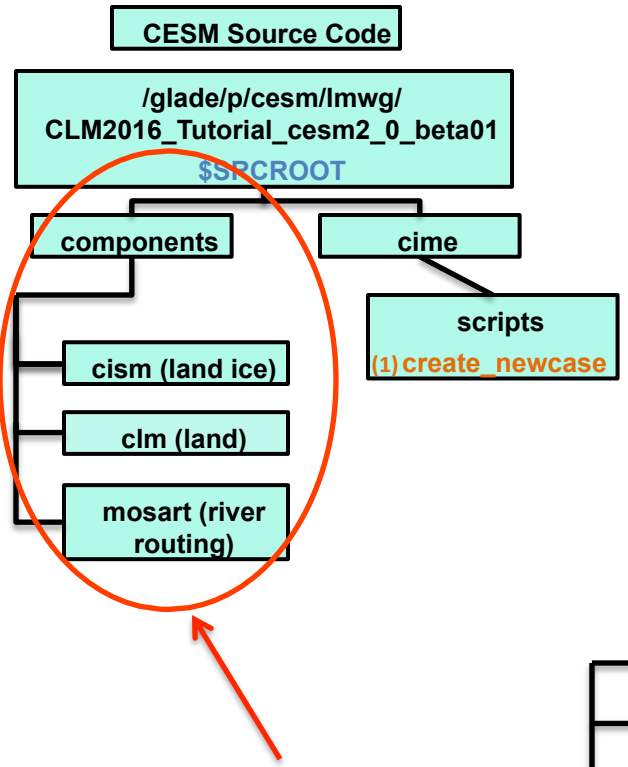
Two methods for modifying code

Method 2: SourceMods

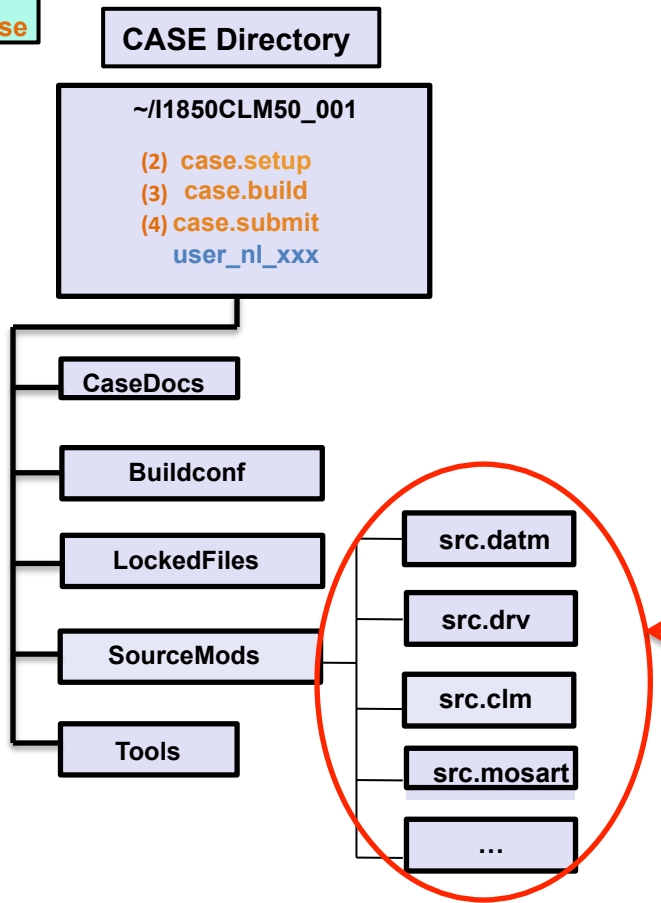
- Best for changes that have some of these characteristics:
  - apply to just one or two cases
  - short-term
    - merging with other CLM changes is a pain
    - can't use version control to help you keep track of your changes over time
  - ideally, limited changes to just a few files
- We'll use this method today



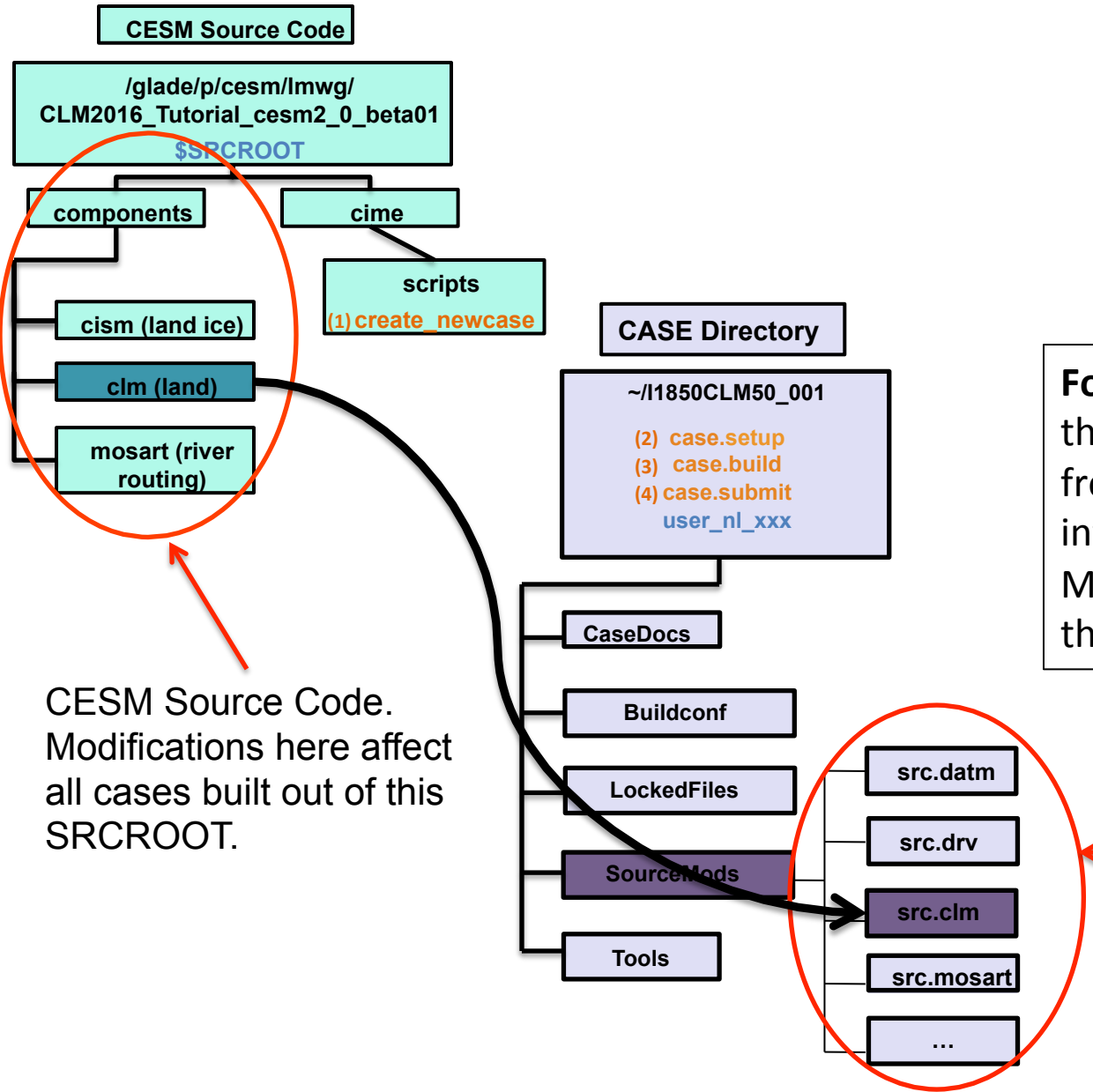
CESM Source Code. Modifications here affect all cases built out of this SRCROOT.



CESM Source Code. Modifications here affect all cases built out of this SRCROOT.



These directories start empty. Modifications here affect the current case only.



CESM Source Code. Modifications here affect all cases built out of this SRCROOT.

For these exercises: Copy the appropriate module from the CESM Source Code into the appropriate Source Mods directory in your case, then modify this copy.

These directories start empty. Modifications here affect the current case only.

Review: The 4 commands to run CLM

cd into scripts directory from the source code directory:

cd /glade/p/cesm/lmwg/CLM2016_Tutorial_cesm2_0_beta01/cime/scripts

(1) create a new case

./create_newcase -case ~/I1850CLM50_001 -res f19_g16 -compset IM1850CRUCLM50BGC

(2) invoke case.setup

cd into case directory:

cd ~/I1850CLM50_001

./case.setup

← **Make modifications BEFORE compiling**

(3) build the executable

Type this command line:

./case.build

(4) submit your run to the batch queue

Type this command line:

./case.submit

Steps for modifying code

After you create your case and run case.setup (steps 1 & 2):

cd /{path to CESMSourceCode}/components/clm/src

- **Find** the fortran files (.F90) that you will modify (use grep to search for specific variables or key words)

cp /{path to CESMSourceCode}/components/clm/src/{directory}/filename.F90 ~/{casename}/SourceMods/src.clm

Note: for the CLM to use this copy, do NOT CHANGE the FILENAME

cd ~/{casename}/SourceMods/src.clm

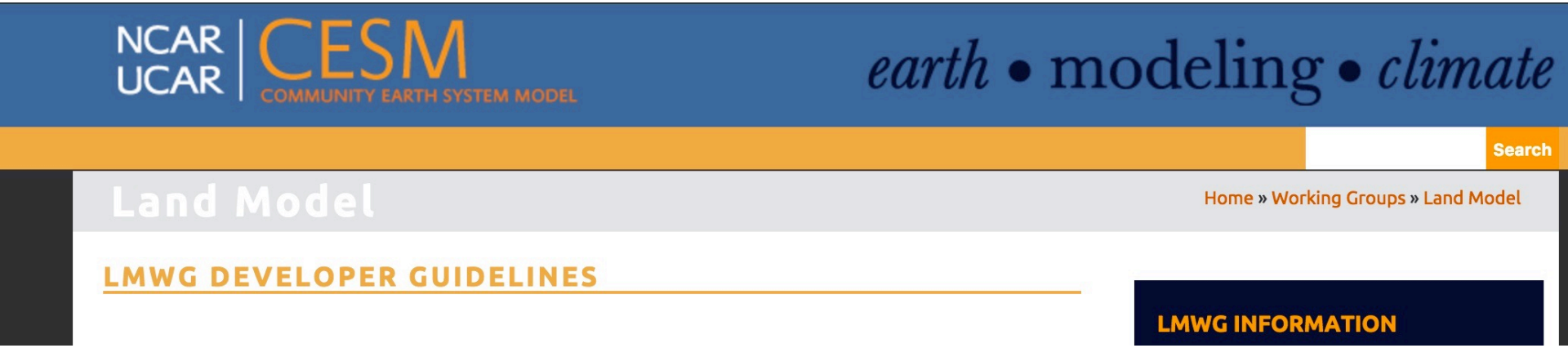
- **Modify** file.F90

Build the executable (case.build) and **Submit** the run (case.submit)

For more elaborate mods…

Can follow examples from existing code

- Keep in mind that some examples are better than others, or at least more appropriate for the changes you want to make
- So: best to check in with experienced CLM developers initially



https://www2.cesm.ucar.edu/working-groups/lmwg/developer-guidelines

#1 Best Practice: Don't Repeat Yourself

Please don't do this!

This is just a small sample (this continued for 1,200 lines)

```
! leafc
ptr => cnveg_carbonstate_inst%leafc_patch(p)
init_state = ptr*wt_old
change_state = ptr*dwt
new_state = init_state+change_state
if (wt_new /= 0._r8) then
   ptr = new_state/wt_new
   conv_cflux(p) = conv_cflux(p) + change_state
else
   ptr = 0._r8
   conv_cflux(p) = conv_cflux(p) - init_state
end if

! leafc_storage
ptr => cnveg_carbonstate_inst%leafc_storage_patch(p)
init_state = ptr*wt_old
change_state = ptr*dwt
new_state = init_state+change_state
if (wt_new /= 0._r8) then
   ptr = new_state/wt_new
   conv_cflux(p) = conv_cflux(p) + change_state
else
   ptr = 0._r8
   conv_cflux(p) = conv_cflux(p) - init_state
end if

! frootc
ptr => cnveg_carbonstate_inst%frootc_patch(p)
dwt_ptr0 => dwt_frootc13_to_litter(p)
init_state = ptr*wt_old
change_state = ptr*dwt
new_state = init_state+change_state
if (wt_new /= 0._r8) then
   ptr = new_state/wt_new
   dwt_ptr0 = dwt_ptr0 - change_state
else
   ptr = 0._r8
   dwt_ptr0 = dwt_ptr0 + init_state
end if
```

#1 Best Practice: Don't Repeat Yourself

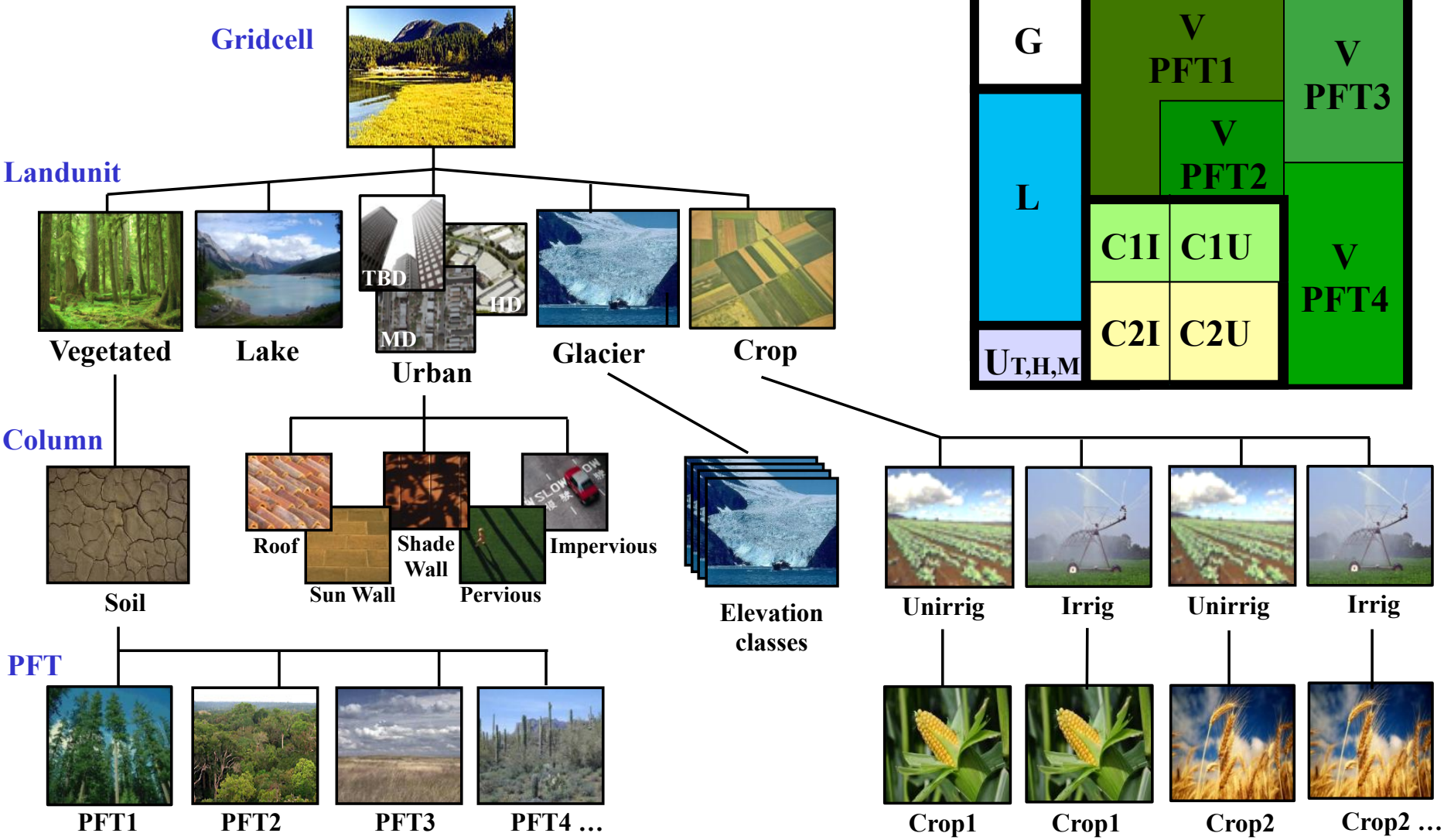
Instead do this

```
call update_patch_state( &
     var = this%leafc_patch(begp:endp), &
     flux_out = conv_cflux(begp:endp), &
     seed = seed_leafc_patch(begp:endp), &
     seed_addition = dwt_leafc_seed(begp:endp))

call update_patch_state( &
     var = this%leafc_storage_patch(begp:endp), &
     flux_out = conv_cflux(begp:endp), &
     seed = seed_leafc_storage_patch(begp:endp), &
     seed_addition = dwt_leafc_seed(begp:endp))

call update_patch_state( &
     var = this%frootc_patch(begp:endp), &
     flux_out = dwt_frootc_to_litter(begp:endp))
```

#1 Best Practice: Don't Repeat Yourself

- Why not to copy & paste existing code
  - If the original code changes, it's hard for you or anyone else to realize that your code needs to change, too, to stay consistent
  - And once they diverge, it's very hard to tell if the divergence is intentional or accidental
- Why not to copy & paste your own code
  - It will be harder to make changes that apply to each instance
  - It's harder to have confidence: need to separately test each instance of the duplicated code
  - If the instances are subtly different, it's hard to see that, and introducing a new instance is error-prone

CLM Arrays, Loops and Filters



CLM Arrays, Loops and Filters

bounds%begp ↓ bounds%endp ↓

patch index (p)	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
o3coefgsun_patch	0.1	0.2	0.0	0.0	0.0	0.3	0.2	0.7	0.0	0.0	0.0	0.9	0.0	0.0	0.0	0.3	0.0	0.0

bounds%begc ↓ bounds%endc ↓

column index (c)	35	36	37	38	39	40	41	42
qflx_drain_col	3.7	5.2	0.0	8.9	0.0	2.7	0.9	0.0

The _patch or _col often doesn't appear in the body of the code, but you can find it by looking at the 'associate' statement for a subroutine, which defines aliases:

```
associate( &
     o3coefvsun => this%o3coefvsun_patch, &
     )
```

CLM Arrays, Loops and Filters

bounds%begp ↓ bounds%endp ↓

patch index (p)	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
o3coefgsun_patch	0.1	0.2	0.0	0.0	0.0	0.3	0.2	0.7	0.0	0.0	0.0	0.9	0.0	0.0	0.0	0.3	0.0	0.0

You could loop through a patch-level array like this:

```
do p = bounds%begp, bounds%endp
   c = patch%column(p)

   o3coefgsun(p) = ...
```

This line is only needed if you need to access column-level arrays in the same loop

But typically in CLM we use “filters” for efficiency...

CLM Arrays, Loops and Filters

Filter of vegetated patches not covered by snow:

These patches are non-vegetated (glacier, lake, wetland or urban)

patch index (p)	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
o3coefgsun_patch	0.1	0.2	0.0	0.0	0.0	0.3	0.2	0.7	0.0	0.0	0.0	0.9	0.0	0.0	0.0	0.3	0.0	0.0

And these patches are covered by snow

filter index (fp)	1	2	3	4	5	6	7
filter_exposedvegp	11	12	15	16	22	23	26

CLM Arrays, Loops and Filters

patch index (p)	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
o3coefgsun_patch	0.1	0.2	0.0	0.0	0.0	0.3	0.2	0.7	0.0	0.0	0.0	0.9	0.0	0.0	0.0	0.3	0.0	0.0

filter index (fp)	1	2	3	4	5	6	7
filter_exposedvegp	11	12	15	16	22	23	26

A loop using this filter looks like this:

```
do fp = 1, num_exposedvegp
   p = filter_exposedvegp(fp)

   o3coefgsun(p) = ...
```

7 in this case

when fp = 1, p = 11
when fp = 2, p = 12
when fp = 3, p = 15
etc.

Model Spinup

What is it? Why do we do it? When do we need it?

Model Spinup

What is it? Why do we do it? When do we need it?

- It takes a while for the model's carbon pools and fluxes to come into equilibrium after starting a model run. We call the period of time when these values are still changing the **spinup**.

- The carbon pools and fluxes "stabilize" once the model is spun up; when forced with constant climate and CO_2, there shouldn't be any drift in the simulated carbon pools and fluxes (i.e. the model is in **equilibrium**).

Model Spinup

Why do we need spinup?

- If the simulated carbon pools and fluxes are still drifting under constant forcing (i.e. constant climate and CO_2), it is impossible to tell what results are simply caused by the model drifting, rather than a real response to an actual forcing.

Example:

Suppose you run a simulation where you double CO_2, but you don't spin the model up first. Are the increases in carbon you observe in your results due to the increased CO_2 concentration (forcing), or are the carbon changes just a result of the model still just trying to come into equilibrium with your initial CO_2 concentration (drift)? Its impossible to tell if you don't spin the model up to equilibrium with the initial CO_2 concentration first!

Model Spinup

How can we tell if a model is spun up?

- By definition, a model is "spun up" when NBP (Net Biome Production) ≈ 0 under steady-state boundary conditions (i.e. constant climate forcing), or when total ecosystem carbon changes less than 0.02 Pg C/year

- At NCAR, we typically test several carbon pools and fluxes:

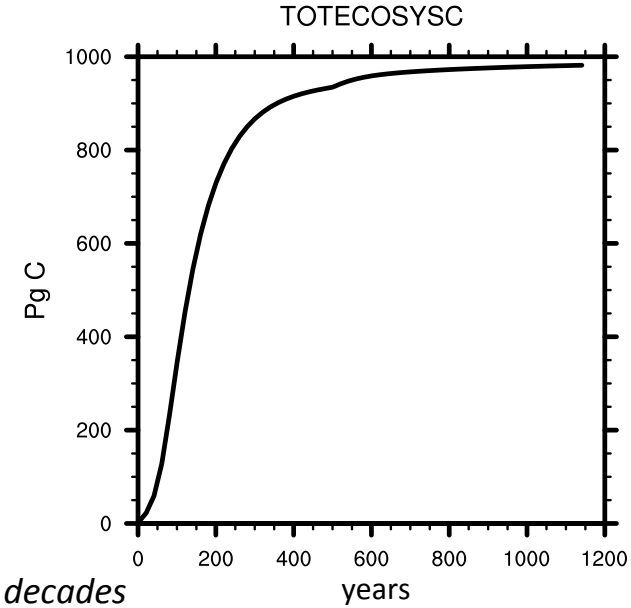
TOTECOSYSC = total ecosystem carbon

TOTSOMC = total soil organic matter carbon

TLAI = total leaf area index

GPP = gross primary productivity

TWS = total water storage

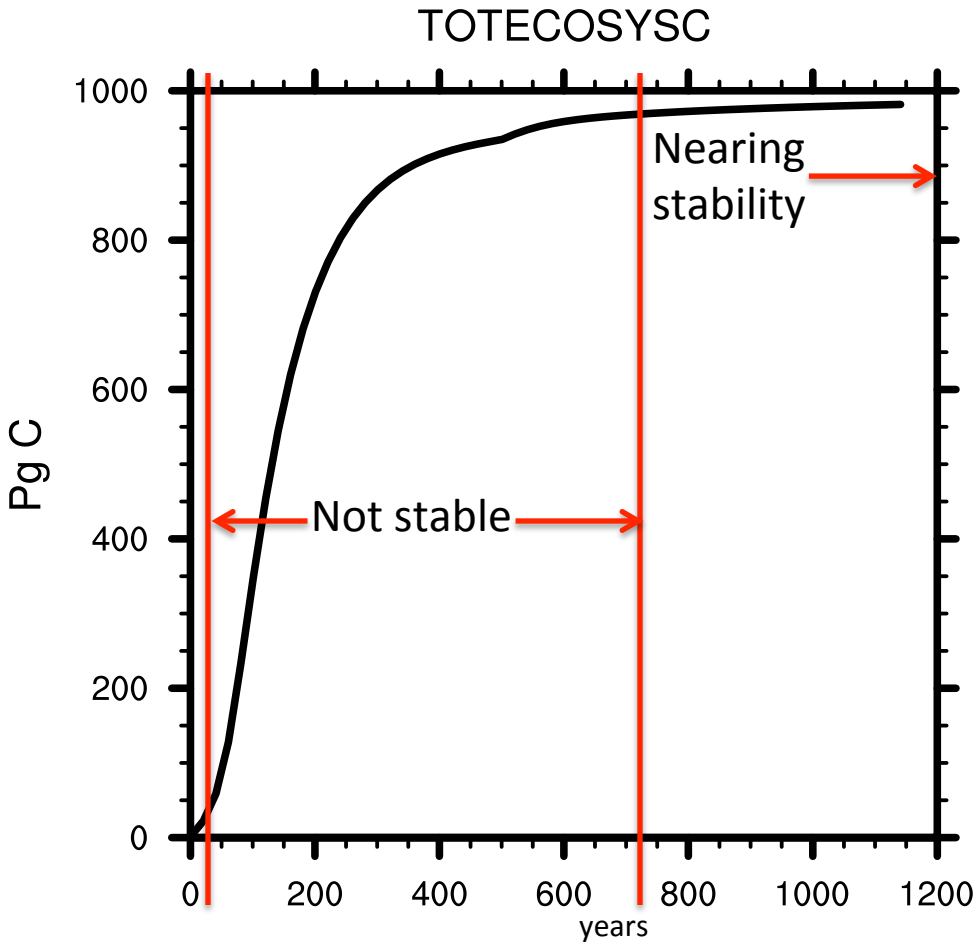


Note: The trends in ecosystem C and NBP should be analyzed over multiple decades

Model Spinup

How can we tell if a model is spun up?

Our current metric:
Carbon pools should be changing less than 0.02 Pg C / yr



Model Spinup

When should we spin up the model?

- Any time you are running a simulation with biogeochemistry (BGC, CN), you should spin up the simulation.
- Initial condition files provided for historical simulations are often already spun up, but its always good to double check!
- Any time you make source code changes, you should spin up.
- The amount of time it takes the model to spin up will depend on how far from equilibrium you are when you start. Check the drift in your carbon pools (goal: less than 0.02 Pg C / yr) rather than running for some set amount of time.

Note: even variables like surface temperature need to spin up – but they spin up very quickly compared to carbon pools!

Exercise 1:
Modifying Source Code

Exercise 1: Modify source code

1) Run a control case for 5-days
   - Create and setup a case
   - Change namelist to enable ozone damage
   - Build and submit case

2) Run another case where we change the ozone plant stress coefficient
   - Create and setup a case
   - Change namelist to enable ozone damage
   - *Copy OzoneMod.F90 to the SourceMod directory*
   - *Modify OzoneMod.F90*
   - Build and submit case

3) Run another case that is the same as 2 but with an additional option
   - Create and setup a case
   - Change namelist to enable ozone damage
   - *Copy OzoneMod.F90 to the SourceMod directory*
   - *Modify OzoneMod.F90*
   - Change a setting in *env_build.xml* to build in DEBUG mode rather than optimized mode
   - Build and submit case

Exercise 1: Modify source code

Control Case: Setup and run the control simulation

1) Create and setup a new case

```
cd /glade/p/cesm/lmwg/CLM2016_Tutorial_cesm2_0_beta01/cime/scripts
./create_newcase -case ~/Control -res f19_g16 -compset IM1850CRUCLM50BGC -mach yellowstone

cd ~/Control
./case.setup
```

2) Open the user_nl_clm and add an option

<add this line to user_nl_clm> use_ozone = .true.

3) Build and submit the control simulation

```
./case.build
./case.submit
```

Exercise 1: Modify source code

Second Simulation: Create another case for code modification

1) Create and setup a new case

```
cd /glade/p/cesm/lmwg/CLM2016_Tutorial_cesm2_0_beta01/cime/scripts
./create_newcase -case ~/Ozone_Test -res f19_g16 -compset IM1850CRUCLM50BGC -mach yellowstone

cd ~/Ozone_Test
./case.setup
```

2) Open the user_nl_clm and add an option

<add this line to user_nl_clm> use_ozone = .true.

Exercise 1: Modify source code

Second Simulation: Create another case for code modification

3) Copy over the file we want to change into your case directory SourceMod directory

```
cp /glade/p/cesm/lmwg/CLM2016_Tutorial_cesm2_0_beta01/components/clm/src/biogeophys/
OzoneMod.F90 ~/Ozone_Test/SourceMods/src.clm/
```

4) Go to SourceMod directory

```
cd ~/Ozone_Test/SourceMods/src.clm/
```

5) Open the OzoneMod.F90 using your preferred text editor and add this line

<in OzoneMod.F90 on line 377 add> o3coefgsun(p) = o3coefgsun(c) ** 3._r8

The intent here is to make ozone's effect on stomatal conductance much more extreme, just for sunlit leaves. Note that o3coefgsun varies from 0 to 1, with 1 meaning no effect and 0 shutting down stomatal conductance. So with this change, an effect of 0.5 should get turned into 0.125.

(Type this new code in exactly as written. If it turns out there is a bug, we'll work through it together in the following slides.)

Exercise 1: Modify source code

Second Simulation: Contents of OzoneMod.F90, with some helpful notes

```
359
360     do fp = 1, num_exposedvegp
361        p = filter_exposedvegp(fp)
362        c = patch%column(p)
363
364        ! Ozone stress for shaded leaves
365        call CalcOzoneStressOnePoint( &
366             forc_ozone=forc_ozone, forc_pbot=forc_pbot(c), forc_th=forc_th(c), &
367             rs=rssha(p), rb=rb(p), ram=ram(p), &
368             tlai=tlai(p), tlai_old=tlai_old(p), pft_type=patch%itype(p), &
369             o3uptake=o3uptakesha(p), o3coefv=o3coefvsha(p), o3coefg=o3coefgsha(p))
370
371        ! Ozone stress for sunlit leaves
372        call CalcOzoneStressOnePoint( &
373             forc_ozone=forc_ozone, forc_pbot=forc_pbot(c), forc_th=forc_th(c), &
374             rs=rssun(p), rb=rb(p), ram=ram(p), &
375             tlai=tlai(p), tlai_old=tlai_old(p), pft_type=patch%itype(p), &
376             o3uptake=o3uptakesun(p), o3coefv=o3coefvsun(p), o3coefg=o3coefgsun(p))
377        o3coefgsun(p)  =  o3coefgsun(c) ** 3._r8
378        tlai_old(p) = tlai(p)
379
380     end do
381
```

We're in a loop over a patch (p) filter

This loop also sets the column (c) index associated with each patch

Code duplication removed via repeated call to a subroutine that does all the work

Line to be added

Exercise 1: Modify source code

Second Simulation: Create another case for code modification

6) Build and submit the new simulation

```
cd ~/Ozone_Test
./case.build
./case.submit
```

7) Examine model output

```
cd /glade/scratch/$USER/archive/Ozone_Test/lnd/hist
module load ncview
ncview Ozone_Test.clm2.h0.0001-01-01-00000.nc
```

Look at the variable O3UPTAKESUN. Do you see 0 values everywhere, for all times?
With 0 ozone uptake, ozone should have no effect.

(Q1) So what do you expect to see if you look at differences between the test case and the control case?
(See slides at the end of the presentation for answers to these questions.)

Exercise 1: Modify source code

Second Simulation: Create another case for code modification

8) Examine differences between test case and control case

```
cd /glade/scratch/$USER/archive/Ozone_Test/lnd/hist
module load nco
ncdiff Ozone_Test.clm2.h0.0001-01-01-00000.nc /glade/scratch/$USER/archive/Control/lnd/hist/
Control.clm2.h0.0001-01-01-00000.nc diffs.nc
ncview diffs.nc
```

Look at the variable FCTR to examine differences in canopy transpiration between the two cases. Click on the "Range" button, and set the range to go from -0.1 to 0.1. Then click on the time box (with the text 1-Jan-0001) to scroll through times.

Are the differences what you expected to see?

Exercise 1: Modify source code

So we are seeing differences that appear pretty random, when we expected to see 0 differences. When we see something unexpected like this, it's good to retry a case built in DEBUG mode. This turns on various checks, for things like using uninitialized variables, dividing by 0, or accessing array elements outside the bounds of the array. These checks are too expensive to run all the time, but it's very important to run your new code with these checks turned on during initial development. It's a great idea to run in DEBUG mode when something seems wrong, when the run crashes with a cryptic error message – and even if everything looks right, just to make sure that there's nothing subtly wrong.

Exercise 1: Modify source code

Third Simulation: Same as second simulation

1) Create and setup a new case

```
cd /glade/p/cesm/lmwg/CLM2016_Tutorial_cesm2_0_beta01/cime/scripts
./create_newcase -case ~/Ozone_Test_Debug -res f19_g16 -compset IM1850CRUCLM50BGC -mach yellowstone

cd ~/Ozone_Test_Debug
./case.setup
```

2) Perform same namelist and code modification as the second simulation

3) Turn on the "debug" option

```
./xmlchange DEBUG=TRUE
```

4) Build and submit the control simulation

```
./case.build
./case.submit
```

Exercise 1: Modify source code

Third Simulation: Same as second simulation

1) Create and setup a new case

```
cd /glade/p/cesm/lmwg/CLM2016_Tutorial_cesm2_0_beta01/cime/scripts
./create_newcase -case ~/Ozone_Test_Debug -res f19_g16 -compset IM1850CRUCLM50BGC -mach yellowstone

cd ~/Ozone_Test_Debug
./case.setup
```

2) Perform same namelist and code modification as the second simulation

3) Turn on the "debug" option

```
./xmlchange DEBUG=TRUE
```

4) Build and submit the control simulation

```
./case.build
./case.submit
```

What happened when the model ran?
Is there output?

Exercise 1: Modify source code

Third Simulation: Same as second simulation

5) Check the case status

```
cd ~/Ozone_Test_Debug
cat CaseStatus
```

Do you see a message saying, "Model did not complete", and pointing you to the cesm.log file? Generally, when your run crashes, you can find useful output in either the cesm.log or lnd.log files – at least if you built with DEBUG=TRUE.

Exercise 1: Modify source code

Third Simulation: Same as second simulation

6) Examine the cesm.log file using the unix command 'less', a text editor, or some other method

Search for the first instance of the word 'exit'. You should see something like this nearby:

```
18:forrtl: severe (408): fort: (3): Subscript #1 of the array O3COEFGSUN has value 7101 which
is less than the lower bound of 16308
```

Then you'll see a list of source files and line numbers (a 'backtrace'). The place that caused the crash is at the top of the list:

```
103:cesm.exe           00000000018EC97B  ozonemod_mp_calco         377  OzoneMod.F90
```

Aha! So the problem is on line 377 of OzoneMod.F90, which is indeed our new line of code!

Open up this file again, and take a close look.

(Q2) Do you see anything wrong?
(See next slide for hint, and end of presentation for answer.)

Exercise 1: Modify source code

<u>Third Simulation:</u> Same as second simulation

(Q2 hint) Hint: o3coefgsun is a patch-level array (just above the loop, you can see that it is aliased to o3coefgsun_patch). How do you see this array being indexed in other parts of this subroutine?

(Q3) So what do you think happened in your first (non-DEBUG) run? Why did you see the seemingly random speckling when you looked at the difference map?
(See end of presentation for answer.)

If you'd like, you can fix this bug in your original, non-DEBUG case, and rerun it. You should now see that the results are identical to the control case.

Or just move on to Exercise 2 (next slide)....

Exercise 2:
Getting a compilation error

Exercise 2: Compiler errors

1) Run another case where we change the ozone coefficient
   - Create and setup a case
   - Change namelist to enable ozone damage
   - *Copy OzoneMod.F90 to the SourceMod directory*
   - *Modify OzoneMod.F90*
   - Add a “debug” option in *env_run.xml*
   - Build and submit case

Exercise 2: Compiler errors

Create another case for code modification

1) Create and setup a new case

```
cd /glade/p/cesm/lmwg/CLM2016_Tutorial_cesm2_0_beta01/cime/scripts
./create_newcase -case ~/Ozone_Test_Reduce -res f19_g16 -compset IM1850CRUCLM50BGC -mach yellowstone

cd ~/Ozone_Test
./case.setup
```

2) Open the user_nl_clm and add an option

<add this line to user_nl_clm> use_ozone = .true.

Exercise 2: Compiler errors

Create another case for code modification

3) Copy over the file we want to change into your case directory SourceMod directory

```
cp /glade/p/cesm/lmwg/CLM2016_Tutorial_cesm2_0_beta01/components/clm/src/biogeophys/
OzoneMod.F90 ~/Ozone_Test_Reduce/SourceMods/src.clm/
```

4) Go to SourceMod directory

```
cd ~/Ozone_Test_Reduce/SourceMods/src.clm/
```

5) Open the OzoneMod.F90 using your preferred text editor and add this line

<in OzoneMod.F90 on line 377 add> o3coefgsun(p) = o3coefgsun(p) ^ 0.5_r8

Exercise 2: Compiler errors

Contents of the OzoneMod.F90

```
359
360     do fp = 1, num_exposedvegp
361        p = filter_exposedvegp(fp)
362        c = patch%column(p)
363
364        ! Ozone stress for shaded leaves
365        call CalcOzoneStressOnePoint( &
366             forc_ozone=forc_ozone, forc_pbot=forc_pbot(c), forc_th=forc_th(c), &
367             rs=rssha(p), rb=rb(p), ram=ram(p), &
368             tlai=tlai(p), tlai_old=tlai_old(p), pft_type=patch%itype(p), &
369             o3uptake=o3uptakesha(p), o3coefv=o3coefvsha(p), o3coefg=o3coefgsha(p))
370
371        ! Ozone stress for sunlit leaves
372        call CalcOzoneStressOnePoint( &
373             forc_ozone=forc_ozone, forc_pbot=forc_pbot(c), forc_th=forc_th(c), &
374             rs=rssun(p), rb=rb(p), ram=ram(p), &
375             tlai=tlai(p), tlai_old=tlai_old(p), pft_type=patch%itype(p), &
376             o3uptake=o3uptakesun(p), o3coefv=o3coefvsun(p), o3coefg=o3coefgsun(p))
377        o3coefgsun(p)  =  o3coefgsun(p) ^ 0.5_r8
378        tlai_old(p) = tlai(p)
379
380     end do
381
```

Line to be added

Exercise 2: Compiler errors

What happened during the build step?

6) Build the case

```
cd ~/Ozone_Test_Reduce
./case.build
```

Exercise 2: Compiler errors

What happened during the build step?

6) Build the case

```
cd ~/Ozone_Test_Reduce
./case.build
```

-- what you will see on the screen after building --

```
 .... calling builds for utility libraries (compiler is intel)
      build libraries: mct gptl pio csm_share
      Wed Sep 14 17:36:51 2016 /glade/scratch/abtawfik/Ozone_Test_Reduce/bld/intel/mpich2/nodebug/nothreads/mct.bldlog.160914-173625
      Wed Sep 14 17:38:22 2016 /glade/scratch/abtawfik/Ozone_Test_Reduce/bld/intel/mpich2/nodebug/nothreads/gptl.bldlog.160914-173625
      Wed Sep 14 17:38:32 2016 /glade/scratch/abtawfik/Ozone_Test_Reduce/bld/intel/mpich2/nodebug/nothreads/pio.bldlog.160914-173625
      Wed Sep 14 17:39:12 2016 /glade/scratch/abtawfik/Ozone_Test_Reduce/bld/intel/mpich2/nodebug/nothreads/csm_share.bldlog.160914-173625
    .... calling builds for component libraries
model = atm, obj = /glade/scratch/abtawfik/Ozone_Test_Reduce/bld/atm/obj
      .... calling /glade/p/cesm/lmwg/CLM2016_Tutorial_cesm2_0_beta01/cime/components/data_comps/datm/cime_config/buildlib
           Wed Sep 14 17:39:50 2016 /glade/scratch/abtawfik/Ozone_Test_Reduce/bld/atm.bldlog.160914-173625
         - Building clm4_5/clm5_0 shared library
      .... calling /glade/p/cesm/lmwg/CLM2016_Tutorial_cesm2_0_beta01/components/clm/cime_config/buildlib
           Wed Sep 14 17:39:55 2016 /glade/scratch/abtawfik/Ozone_Test_Reduce/bld/lnd.bldlog.160914-173625
ERROR: clm.buildlib failed, see /glade/scratch/abtawfik/Ozone_Test_Reduce/bld/lnd.bldlog.160914-173625
Uncaught exception from user code:
     ERROR: clm.buildlib failed, see /glade/scratch/abtawfik/Ozone_Test_Reduce/bld/lnd.bldlog.160914-173625 at ./case.build line 620
 at ./case.build line 620
     main::buildModel() called at ./case.build line 228
     main::main() called at ./case.build line 697
```

Exercise 2: Compiler errors

What happened during the build step?

6) Build the case

```
cd ~/Ozone_Test_Reduce
./case.build
```

-- what you will see on the screen after building --

```
 .... calling builds for utility libraries (compiler is intel)
      build libraries: mct gptl pio csm_share
      Wed Sep 14 17:36:51 2016 /glade/scratch/abtawfik/Ozone_Test_Reduce/bld/intel/mpich2/nodebug/nothreads/mct.bldlog.160914-173625
      Wed Sep 14 17:38:22 2016 /glade/scratch/abtawfik/Ozone_Test_Reduce/bld/intel/mpich2/nodebug/nothreads/gptl.bldlog.160914-173625
      Wed Sep 14 17:38:32 2016 /glade/scratch/abtawfik/Ozone_Test_Reduce/bld/intel/mpich2/nodebug/nothreads/pio.bldlog.160914-173625
      Wed Sep 14 17:39:12 2016 /glade/scratch/abtawfik/Ozone_Test_Reduce/bld/intel/mpich2/nodebug/nothreads/csm_share.bldlog.160914-173625
    .... calling builds for component libraries
model = atm, obj = /glade/scratch/abtawfik/Ozone_Test_Reduce/bld/atm/obj
      .... calling /glade/p/cesm/lmwg/CLM2016_Tutorial_cesm2_0_beta01/cime/components/data_comps/datm/cime_config/buildlib
           Wed Sep 14 17:39:50 2016 /glade/scratch/abtawfik/Ozone_Test_Reduce/bld/atm.bldlog.160914-173625
         - Building clm4_5/clm5_0 shared library
      .... calling /glade/p/cesm/lmwg/CLM2016_Tutorial_cesm2_0_beta01/components/clm/cime_config/buildlib
           Wed Sep 14 17:39:55 2016 /glade/scratch/abtawfik/Ozone_Test_Reduce/bld/lnd.bldlog.160914-173625
ERROR: clm.buildlib failed, see /glade/scratch/abtawfik/Ozone_Test_Reduce/bld/lnd.bldlog.160914-173625
Uncaught exception from user code:
     ERROR: clm.buildlib failed, see /glade/scratch/abtawfik/Ozone_Test_Reduce/bld/lnd.bldlog.160914-173625 at ./case.build line 620
 at ./case.build line 620
     main::buildModel() called at ./case.build line 228
     main::main() called at ./case.build line 697
```

Look at the build log
What is going wrong?

Exercise 2: Compiler errors

Contents of the build log → lnd.bldlog.*

-- Go to the end of the file --

```
/glade/u/home/abtawfik/Ozone_Test_Reduce/SourceMods/src.clm/OzoneMod.F90(378): error #5078: Unrecognized token '^' skipped
       o3coefgsun(p) = o3coefgsun(p) ^ 0.5_r8
-------------------------------------^
/glade/u/home/abtawfik/Ozone_Test_Reduce/SourceMods/src.clm/OzoneMod.F90(378): error #5082: Syntax error, found REAL_KIND_CON '0.5' when
expecting one of: ( * ) :: , <END-OF-STATEMENT> ; . % (/ + - [ : ] /) . ' ** / ...
       o3coefgsun(p) = o3coefgsun(p) ^ 0.5_r8
-------------------------------------^
compilation aborted for /glade/u/home/abtawfik/Ozone_Test_Reduce/SourceMods/src.clm/OzoneMod.F90 (code 1)
/glade/u/home/abtawfik/Ozone_Test_Reduce/Tools/Makefile:739: recipe for target 'OzoneMod.o' failed
gmake: *** [OzoneMod.o] Error 1
gmake: *** Waiting for unfinished jobs....
rm dynVarMod.F90 array_utils.F90 dynVarTimeUninterpMod.F90 initInterp2dvar.F90 dynVarTimeInterpMod.F90 ncdio_pio.F90 restUtilMod.F90
ERROR: clm.buildlib gmake complib -j 8 MODEL=clm COMPLIB=/glade/scratch/abtawfik/Ozone_Test_Reduce/bld/intel/mpich2/nodebug/nothreads/MCT/
noesmf//lib/libclm.a USER_CPPDEFS=" " -f /glade/u/home/abtawfik/Ozone_Test_Reduce/Tools/Makefile failed: 512
```

Here is the error!
It looks like we used the wrong syntax for the exponent

Bonus Exercise 3:
Modifying model parameters

Bonus Exercise 3: Modify zlnd parameter

We will modify the CLM parameter

zlnd = roughness length for soil (m)

We will then compare this simulation against the Control experiment from Exercise 1

1) Create and setup a new case

```
cd /glade/p/cesm/lmwg/CLM2016_Tutorial_cesm2_0_beta01/cime/scripts
./create_newcase -case ~/Test_ZLND_0.02 -res f19_g16 -compset IM1850CRUCLM50BGC -mach yellowstone

cd ~/Test_ZLND_0.02
./case.setup
```

2) Open the user_nl_clm and add an option

<add this line to user_nl_clm> use_ozone = .true.

Bonus Exercise 3: Modify zlnd parameter

3) Find the zlnd parameter

```
cd /glade/p/cesm/lmwg/CLM2016_Tutorial_cesm2_0_beta01/components/clm/src
grep zlnd */*
```

-- what you will see on the screen after using grep --

```
biogeophys/CanopyHydrologyMod.F90:     use clm_varcon, only : hfus, denice, zlnd, rpi, spval,
tfrz
biogeophys/CanopyHydrologyMod.F90:     frac_sno(c) = tanh(snow_depth(c)/(2.5_r8*zlnd* &
biogeophys/CanopyHydrologyMod.F90:     frac_sno(c) = tanh(snow_depth(c)/(2.5_r8*zlnd* &
biogeophys/CanopyTemperatureMod.F90:   use clm_varcon, only : denh2o, denice, roverg, hvap,
hsub, zlnd, zsno, tfrz, spval
biogeophys/CanopyTemperatureMod.F90:   z0mg(c) = zlnd
biogeophys/WaterStateType.F90:         use clm_varcon, only : h2osno_max, zlnd, tfrz, spval, pc
biogeophys/WaterStateType.F90:         this%frac_sno_col(c) = tanh( this%snow_depth_col(c) /
(2.5 * zlnd * fmelt) )
main/clm_varcon.F90:                   real(r8) :: zlnd = 0.01_r8 ! Roughness length for soil
```

-- Find the file where zlnd is being set --

Bonus Exercise 3: Modify zlnd parameter

4) Copy over the file we want to change into your case directory SourceMod directory

```
cp /glade/p/cesm/lmwg/CLM2016_Tutorial_cesm2_0_beta01/components/clm/src/main/clm_varcon.F90 ~/
Test_ZLND_0.02/SourceMods/src.clm/
```

5) Go to SourceMod directory

```
cd ~/Test_ZLND_0.02/SourceMods/src.clm/
```

6) Open the clm_varcon.F90 using your preferred text editor and change zlnd

<in clm_varcon.F90> change zlnd = 0.01 to zlnd = 0.02 **<in clm_varcon.F90>**

Bonus Exercise 3: Modify zlnd parameter

6) Build and submit the case

```
cd ~/Test_ZLND_0.02/
./case.build
./case.submit
```

Bonus Exercise 3: Modify zlnd parameter

Once the simulation is done → check the effects of changing **zlnd**

We will use ***ncdiff*** to get a difference and then use ***ncview*** to take a look

9) Take a difference between files

```
module load nco
ncdiff /glade/scratch/<user_name>/Test_ZLND_0.02/run/Test_ZLND_0.02.clm2.h0.0001-01.nc /glade/
scratch/<user_name>/Control/run/Control.clm2.h0.0001-01.nc difference.nc
```

10) Look to see the difference between simulations

```
ncview difference.nc
```

Does fraction snow cover change (FSNO)? What other variables changed?

Bonus Exercise 4: Adding a history field

a.k.a. including a new variable in the model output

We wanted to include Vcmax (the maximum rate of carboxylation) in the model output, so we added the variable “VCMAX25TOP” to the history files.

- Note: All modifications are flagged by “!KO”, so you can search for this to find the changes.

Example of modifying history fields:

Copy the following code into the SourceMods/src.clm directory in your I1850 simulation:

/glade/u/home/oleson/I1850CLM50_001/SourceMods/src.clm/PhotosynthesisMod.F90

```
cd ~/I1850CLM50_001

cp  /glade/u/home/oleson/I1850CLM50_001/SourceMods/src.clm/PhotosynthesisMod.F90  ~/I1850CLM50_001/SourceMods/src.clm
```

We wanted to include Vcmax (the maximum rate of carboxylation) in the model output, so we added the variable “VCMAX25TOP” to the history files.

- Note: All modifications are flagged by “!KO”, so you can search for this to find the changes.

Example of modifying history fields:

Copy the following code into the SourceMods/src.clm directory in your I1850 simulation:

/glade/u/home/oleson/I1850CLM50_001/SourceMods/src.clm/PhotosynthesisMod.F90

After you copy the code into your directory, compile and submit the simulation.

```
./case.build

#Note: Check the env_run.xml and env_batch.xml files. Set these to run for a few months.

./case.submit
```

After the simulation completes, check the history files (try using “ncview”) to see if the variable VCMAX25TOP is recorded. (/glade/scratch/{USERID}/archive/{CASENAME}/lnd/hist

We wanted to include Vcmax (the maximum rate of carboxylation) in the model output, so we added the variable “VCMAX25TOP” to the history files.

- Note: All modifications are flagged by “!KO”, so you can search for this to find the changes.

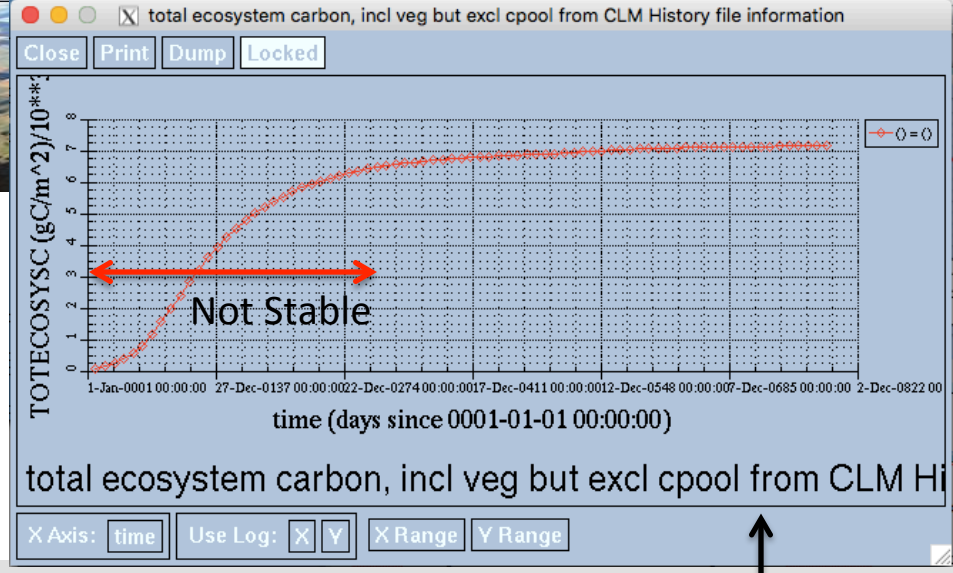
To see the differences between this version of the PhotosynthesisMod.F90 subroutine and the original source code, you can difference the new file from the original file. You can use “xxdiff”:

```
xxdiff ~/I850CLM50_001/SourceMods/src.clm/PhotosynthesisMod.F90 /glade/p/cesm/lmwg/CLM2016_Tutorial_cesm2_0_beta01/
components/clm/src/biogeophys/PhotosynthesisMod.F90
```

[Basic syntax: xxdiff file1 file2]

Spinup Extras

A quick way to assess drift: create a time series of the global mean of some carbon pool



Sample shell script to create global averages:
(script courtesy of Marysa Laguë)

This script creates a netcdf file (.nc) that can be opened using ncview. You can visually assess whether a specific pool is still drifting.

```
#! /bin/tcsh
# Example shell script to make a time series of a globally averaged variable

set casename = TRENDY2016_n03_clm4_5_1_r087_1860Spin
set var = TOTECOSYSC
set filedir = /glade/scratch/dll/CLMTutorial2016_DataForAnalysis/SpinupFiles
set workdir = /glade/scratch/$user/spinup

# make a directory for your spinup netcdf files, if it doesn't already exist:
mkdir -p $workdir

# ncrcat a segment of these files into a single file timeseries - since there are a LOT of files, I'm only
looking at January every 10 years
ncrcat -O -v $var,area $filedir/$casename.clm2.h0.0[0-8]?0-01.nc $workdir/$casename.clm2.h0.10yr-01_${var}
_ts.nc

# make global average of this timeseries (easier to look at if it is spun up or not)
ncwa -O -w area -a lat,lon $workdir/$casename.clm2.h0.10yr-01_${var}_ts.nc $workdir/
$casename.clm2.h0.10yr-01_${var}_ts.global_mean.nc

# delete the larger file with the non-global average maps in them:
rm -r $workdir/$casename.clm2.h0.10yr-01_${var}_ts.nc
echo "complete"
```

/glade/p/work/mlague/spinup/variable_drift.csh

<- Path where the script can be copied from

How to spin up a simulation

1. Set up an 1850 simulation

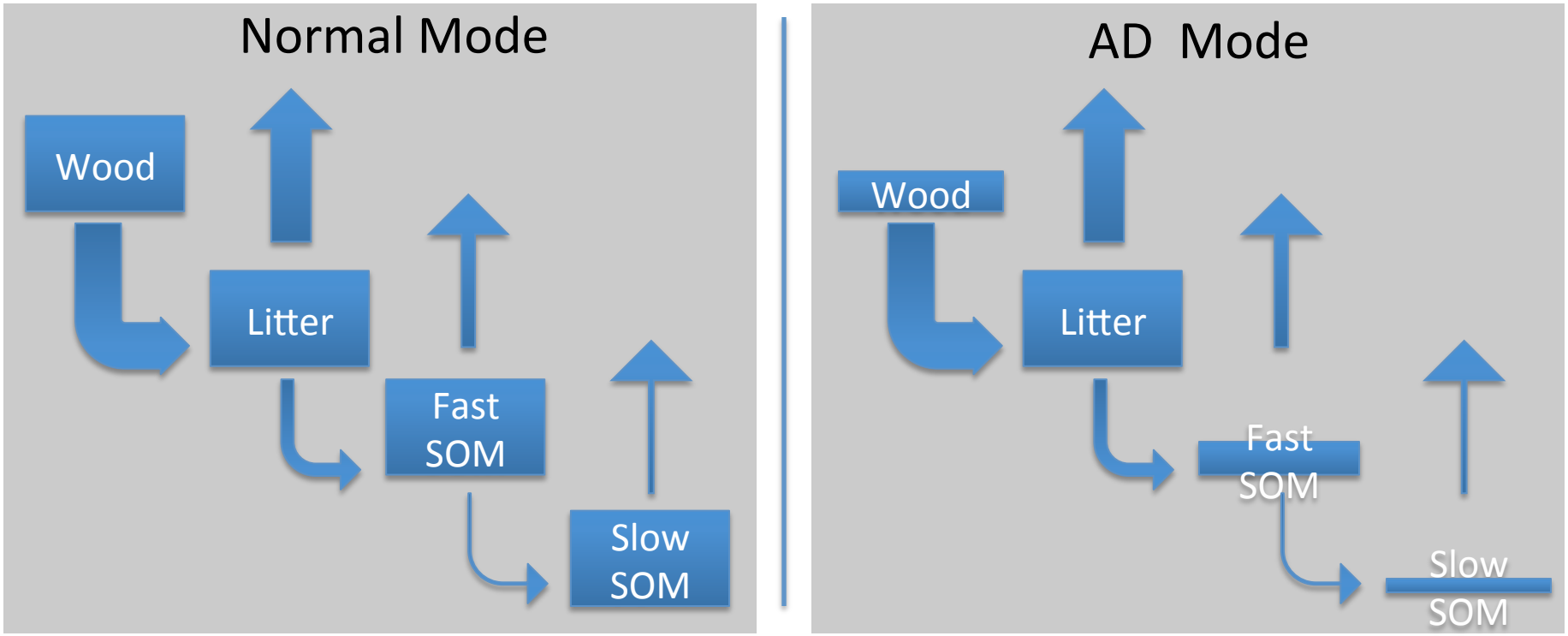
2. Run until C pools and fluxes are stable

   Accelerated decomposition mode (AD) makes this quicker

   - Stabilization will take different amounts of time depending on the starting conditions (starting from bare ground with 0 carbon will take longer to stabilize than starting from a file that has some carbon)

3. Use the restart file from the stabilized simulation as the "`finidat`" in `user_nl_clm`

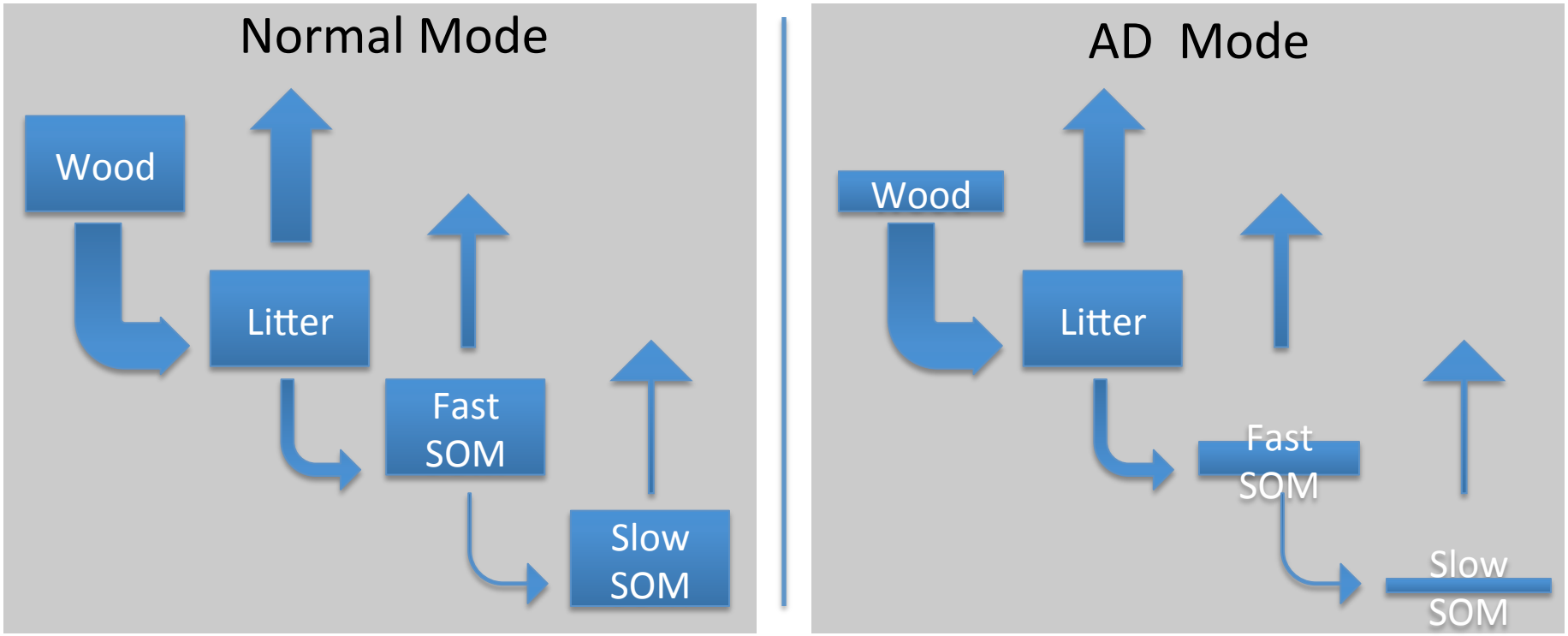
“Accelerated Decomposition” Spinup: how it works



By definition, model is “spun up” when NBP ≈ 0 under steady-state boundary conditions.
Goal: rapidly find a system of carbon stocks that has the same C and N fluxes as full model

The *problem* is that N mineralization ties GPP to slowest pools, so can’t just solve for stocks (unless we iterate, as in Xia et al., 2013, but this ends up being not really any faster). It takes a *long* time for the large, slow pools to equilibrate.

“Accelerated Decomposition” Spinup: how it works



By definition, model is “spun up” when NBP ≈ 0 under steady-state boundary conditions.
Goal: rapidly find a system of carbon stocks that has the same C and N fluxes as full model

Method: With “accelerated decomposition”, we drop the turnover times of slow pools so that the “bucket” is smaller and fills faster. Once the model is spun up with the small pools, we scale the pools back up to reflect the acceleration terms.

Accelerated Decomposition spinup: How to do it

1. Run simulation (starting from bare ground or a prior restart file) with AD spinup set to 'on' until the model equilibrates.

1) In the case directory, modify env_run.xml so that CLM_BLDNML_OPTS is set to "–bgc_spinup on"

```
cd ~/$casename/
./xmlchange -file env_run.xml -id CLM_BLDNML_OPTS -val "-bgc_spinup on"
```

2. Once the model has spun up, set `–bgc_spinup` to `off`

2) After spinup, modify env_run.xml so that CLM_BLDNML_OPTS is set to "–bgc_spinup off"

```
cd ~/$casename/
./xmlchange -file env_run.xml -id CLM_BLDNML_OPTS -val "-bgc_spinup off"
```

3. Use the restart file from step 1 to continue your run in step 2.

   The model automatically adjusts the carbon stocks between step 1 and step 2 to reflect acceleration terms.

How to spin up a simulation: details

1. Set up an 1850 simulation
2. Run until C pools and fluxes are stable (possibly using accelerated decomposition)
   - Check the drift in carbon in the output history files from your 1850 simulation.
   - Once the drift is < 0.2 Pg C / yr, locate the last restart file from your 1850 run. Restart files have the filename

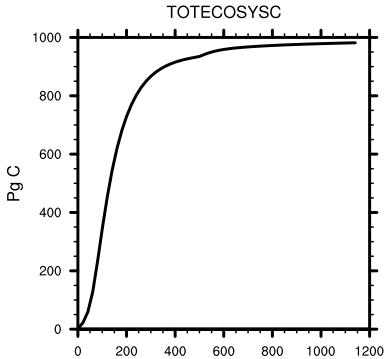
     `$casename.clm2.r.*.nc`

     located in:

     `/glade/scratch/$user/$casename/run/`

     or

     `/glade/scratch/$user/archive/$casename/lnd/rest/[time]/`



3. In user_nl_clm, set `finidat` to the last restart file from the spinup:

```
finidat = '/path/to/file/$casename.clm2.r.0099-12.nc'
```

Where to find help ?

Search

CESM Models

Home » CESM Models » CESM1.2 Series Public Release

CESM1.2 SERIES PUBLIC RELEASE

ABOUT THIS RELEASE SERIES

The CESM1.2 release has numerous new key features among which are the addition of CLM4.5, new science changes to CAM5 running with the CAM-SE dynamical core, and new scripting infrastructure for the generation of component sets, grids and model testing.

CESM1.2 SERIES RELEASE NOTES

Please read the CESM1.2 Series Release Notes which includes What's New - Science, What's New - Software, Answer-Changing Features, Supported Machines, and Known Problems. The new scripting infrastructure is described in detail in the CESM1.2 User's Guide.

SCIENTIFIC VALIDATION

Scientific validation consists of a multi-decadal model run of the given component set at the target resolution, followed by scientific review of the model output diagnostics. All scientifically supported component sets are also accompanied by diagnostic and model output data. Validated CESM1.2 model results and diagnostics will be added to the CESM1.2 website as they become available.

What version of the model should I use?
For a scientifically supported target component set and resolution, please refer to the Scientifically Validated Configurations for that target configuration. For component sets and resolutions that are not scientifically validated in any supported release (e.g. cesm1.0.5 and cesm1.1.1), CSEG strongly urges you to use the latest model release (in this case cesm1.2.0).

DIAGNOSTIC PACKAGES AND NAMING CONVENTIONS

- Post Processing Utilities
- Model File Naming Conventions
- Experiment Case Naming Conventions

MODEL DOCUMENTATION

CESM1.2
- User's Guide
- Machines, Resolutions, Component sets
- Model Component Namelists
- $CASEROOT xml files

Atmosphere Models
- Community Atmosphere Model (CAM5, CAM-CHEM, WACCM)
- Climatological Data Model (DATM)

Land Models
- Community Land Model (CLM4.0, CLM4.5)
- Climatological Data Model (DLND)

Sea Ice Models
- Community Ice CodE (CICE4)
- Climatological Ice Model (DICE)

Coupler
- CESM Coupler (CPL7)

Ocean Models
- Parallel Ocean Program (POP2, POP2-BGC)
- Climatological/Slab-Ocean Data Model (DOCN)

Land Ice Models
- Community Ice Sheet Model (Glimmer-CISM)

River Models
- River Transport Model (RTM)
- Climatological River Runoff Model (DROF)

CESM PROJECT

The Community Earth System Model (CESM) is a fully-coupled, global climate model that provides state-of-the-art computer simulations of the Earth's past, present, and future climate states.

CESM is sponsored by the National Science Foundation (NSF) and the U.S. Department of Energy (DOE). Administration of the CESM is maintained by the Climate and Global Dynamics Division (CGD) at the National Center for Atmospheric Research (NCAR).

MODEL SOURCE CODE

Copyright and Terms of Use

All CESM source code is subject to the following Copyright Notice and Disclaimer.

Acquiring the Release Code

The source code for CESM releases is distributed through a public Subversion code repository. This code can be checked out using Subversion client software such as the command tool svn, or simply view the latest version with a web browser.

A short registration is required to access the repository. After registering, you will receive an email containing a user name and password that is necessary to gain access to the repository.

Acquistion of the code is more fully described in the most recent version of the CESM1.2 User's Guide.

REPORTING A PROBLEM

If you have any problems, please first read the User's Guide including the sections on FAQs and Use Cases. Please also refer to the CESM Bulletin Board, which is in place to facilitate communication within the CESM community. Finally, please also refer to the Release Notes entries that are provided with every release and release update. If questions or problems still exist, then please send an email to cesm-help@cgd.ucar.edu. Support questions will be answered as resources are available.

CESM SUPPORT POLICY

CESM Support Policy - November 2012

CESM DATA MANAGEMENT & DISTRIBUTION PLAN

The Community Earth System Model (CESM) Data Management and Data Distribution Plan documents the procedures for the storage and

The CESM webpage is a gold mine for model documentation

If you cannot find an answer in the documentation, post your question on the CESM Bulletin Board

The CESM Bulletin Board is a forum to ask your questions and to facilitate communication within the CESM community

Google™ Custom Search | Search

CESM Models

Home » CESM Models » CESM1.2 Series Public Release

CESM1.2 SERIES PUBLIC RELEASE

ABOUT THIS RELEASE SERIES

The CESM1.2 release has numerous new key features among which are the addition of CLM4.5, new science changes to CAM5 running with the CAM-SE dynamical core, and new scripting infrastructure for the generation of component sets, grids and model testing.

CESM1.2 SERIES RELEASE NOTES

Please read the CESM1.2 Series Release Notes which includes What's New - Science, What's New - Software, Answer-Changing Features, Supported Machines, and Known Problems. The new scripting infrastructure is described in detail in the CESM1.2 User's Guide.

SCIENTIFIC VALIDATION

Scientific validation consists of a multi-decadal model run of the given component set at the target resolution, followed by scientific review of the model output diagnostics. All scientifically supported component sets are also accompanied by diagnostic and model output data. Validated CESM1.2 model results and diagnostics will be added to the CESM1.2 website as they become available.

What version of the model should I use?
For a scientifically supported target component set and resolution, please refer to the Scientifically Validated Configurations for that target configuration. For component sets and resolutions that are not scientifically validated in any supported release (e.g. cesm1.0.5 and cesm1.1.1), CSEG strongly urges you to use the latest model release (in this case cesm1.2.0).

DIAGNOSTIC PACKAGES AND NAMING CONVENTIONS

- Post Processing Utilities
- Model File Naming Conventions
- Experiment Case Naming Conventions

MODEL DOCUMENTATION

CESM1.2
- User's Guide
- Machines, Resolutions, Component sets
- Model Component Namelists
- $CASEROOT xml files

Atmosphere Models
- Community Atmosphere Model (CAM5, CAM-CHEM, WACCM)
- Climatological Data Model (DATM)

Land Models
- Community Land Model (CLM4.0, CLM4.5)
- Climatological Data Model (DLND)

Sea Ice Models
- Community Ice CodE (CICE4)
- Climatological Ice Model (DICE)

Coupler
- CESM Coupler (CPL7)

Ocean Models
- Parallel Ocean Program (POP2, POP2-BGC)
- Climatological/Slab-Ocean Data Model (DOCN)

Land Ice Models
- Community Ice Sheet Model (Glimmer-CISM)

River Models
- River Transport Model (RTM)
- Climatological River Runoff Model (DROF)

CESM PROJECT

The Community Earth System Model (CESM) is a fully-coupled, global climate model that provides state-of-the-art computer simulations of the Earth's past, present, and future climate states.

CESM is sponsored by the National Science Foundation (NSF) and the U.S. Department of Energy (DOE). Administration of the CESM is maintained by the Climate and Global Dynamics Division (CGD) at the National Center for Atmospheric Research (NCAR).

MODEL SOURCE CODE

Copyright and Terms of Use

All CESM source code is subject to the following Copyright Notice and Disclaimer.

Acquiring the Release Code

The source code for CESM releases is distributed through a public Subversion code repository. This code can be checked out using Subversion client software, such as the command tool svn, or simply view the latest version with a web browser.

A short registration is required to access the repository. After registering, you will receive an email containing a user name and password that is necessary to gain access to the repository.

Acquisition of the code is more fully described in the most recent version of the CESM1.2 User's Guide.

REPORTING A PROBLEM

If you have any problems, please first read the User's Guide including the sections on FAQs and Use Cases. Please also refer to the CESM Bulletin Board, which is in place to facilitate communication within the CESM community. Finally, please also refer to the Release Notes entries that are provided with every release and release update. If questions or problems still exist, then please send an email to cesm-help@cgd.ucar.edu. Support questions will be answered as resources are available.

CESM SUPPORT POLICY

CESM Support Policy - November 2012

CESM DATA MANAGEMENT & DISTRIBUTION PLAN

The Community Earth System Model (CESM) Data Management and Data Distribution Plan documents the procedures for the storage and

---

About | FAQ | Contact Us

NCAR UCAR **DiscussCESM** — COMMUNITY Earth System MODEL

FORUMS | REGISTER | LOGIN | Search

Home » Forums

FORUMS

View Forums | Active topics | Unanswered topics

CESM - General
The Community Earth System Model (CESM) is a fully coupled, global climate model that provides state-of-the-art computer simulations of the Earth's past, present, and future climate states.

Forum	Topics	Posts	Last post
Announcements	19	46	CESM1.1.2 Release Announcement by aliceb July 30, 2013 - 11:07am
Bug reporting	120	335	Error in executing CESM 1.2.0 T85_gx1v6 compset B_2000 by yli_ouc@... August 6, 2013 - 2:16am
Input Data inquiries	113	269	F_2000 compset SST data by torbenmllr@... 20 hours 22 min ago
Output Data inquiries	89	210	Difference between SNOTTOPL and TG in CLM history output (CLM4.0)? by fyke@... August 5, 2013 - 2:43pm
Tools A place for questions about the ESMF mapping tools and the cprnc tool as well as any topics related to grid generation.	4	15	-grid_file equivalent for CESM1.3 (-user_grid_file)? by erik July 25, 2013 - 9:52pm
Software Development Includes issues for building/running on supported machines and porting to unsupported machines	188	563	CESM 1.0.4 run failed when initializing lnd component by jedwards 1 day 19 hours ago
General Discussion Includes requests for new features and configuration inquiries	209	546	basic example by jedwards August 6, 2013 - 2:59pm
Subversion Issues Forum for issues related to the new version control system	11	24	CCSM4 on yellowstone by jedwards August 5, 2013 - 11:44am
Tutorials For discussion regarding the web based modeling tutorials	5	13	Basic B_1850 Compilation by sstrey2@... June 4, 2013 - 9:10am