CICE Consortium tutorial - Standalone CICE and Icepack activity

Learning Goals

In this activity you will clone the CICE and Icepack model code from the Consortium GitHub repository to the Cheyenne supercomputer and run standalone CICE and Icepack simulations. You will also make namelist changes and code modifications for experiments and compare these to control simulations.

Notes:

- A line with a right-facing angle bracket (>) followed by text, denotes a command you need to enter at the command line on the supercomputer but without the angle bracket.
- When there is the <X> syntax, you need to fill in your personal information (e.g. a URL or username) for that command but without the angle brackets. Your GitHub and Cheyenne usernames may not be the same so check which you need to use.
- ***You may work entirely on your own for these activities, but we recommend you find a partner or small group to turn to for help, discuss the questions, and coordinate some of the afternoon experiments.***

Part 2 - Making modifications (~1.5 hrs):

This afternoon you will build on what you learned this morning. You will make basic CICE modifications for experimental runs and run testing procedures to evaluate the impact of those changes on the experiments.

CICE code change

Studies have suggested that in the future the winds in polar regions may be stronger (e.g. https://doi.org/10.4236/acs.2019.94042 and https://doi.org/10.1126/sciadv.aao4719). We want to explore the impacts on sea ice of "windier" polar regions by increasing the zonal and meridional wind speed by 50%. To test this, we will make a non-physical change to the CICE code to see how sensitive the sea ice might be to increasing winds.

Whenever you are testing code you should make a branch for testing instead of doing the tests within your master branch. That way you always have a "pristine" branch you can go back to.

1. Clone your fork again to a new sandbox:
   ```
   >cd ~/cice-tutorial/
   >git clone <URL to repository> cice_windtest --recursive
   >cd cice_windtest
   ```

Check the branch you're on by default and which branches are available

```
>git status
>git branch
```

Now create a new branch in this sandbox. Check what branches are now available, checkout the branch you just made, and then verify which branch you are currently using.

```
>git branch windtest
>git branch
>git checkout windtest
```

```
>git status
```

You should see you are now on the “windtest” branch and it is clean (i.e. there are no changes).

Make sure the version of Icepack is consistent with your branch version. This is particularly useful if a branch is older (i.e. we didn’t just create it).

```
>git submodule update --init
```

2. Make code changes to increase the winds:

In /cicecore/cicedynB/general/ice_forcing.F90 make the following changes

```
use ice_constants, only: c0, c1, c2, c3, c4, c5, c8, c10, c12, c15, c20, &
        TO
use ice_constants, only: c0, c1, c1p5, c2, c3, c4, c5, c8, c10, c12, c15, c20, &
        AND
workx = uatm(i,j) ! wind velocity, m/s
worky = vatm(i,j)
        TO
workx = c1p5*uatm(i,j) ! wind velocity, m/s
worky = c1p5*vatm(i,j)
```

3. Now, go back to the main cice_windtest directory and check the status again

```
>git status
```

You should see ice_forcing.F90 has been modified but the changes are “not staged for commit”.

4. Stage the changes and commit the change to the “windtest” branch.

```
>git add cicecore/cicedynB/general/ice_forcing.F90
>git status
```

This should now return the file with the message “Changes staged for commit”. Go ahead and commit the change and push it to your fork on github.

```
>git commit -m "Increase zonal wind speed by 50%"
>git push origin windtest
(it will ask for github authentication)
```

Look online at your fork. Note that a new branch, “windtest”, has been created. If you look at the file you modified you will see the change you just made and pushed back to your fork with the message you specified.

Run a CICE Experiment

1. Work with a partner to complete the following directions. Using the instructions from this morning, set up two new CICE cases.

    a. The first partner will run the first case will be a baseline from your cice_master sandbox and should be named “cice_base”.
    b. The second partner will run the second case will from your cice_windtest sandbox and should be named “cice_windtest”.

2. Using the instructions from this morning for guidance, set up and run each case for a full calendar year with only monthly output. Each run should take ~10 minutes and they can be run at the same time.

3. Once the simulations have both completed the full year, you can look at the output using the following types of commands:
   >module load ncview
   >module load nco/4.7.9
   >ncview <filename>.nc
   >ncdiff <filename1>.nc <filename2>.nc diff.nc
   >ncview diff.nc

Keep in mind that the output from the simulations will be in different directories.

4. You can also plot timeseries of the hemispheric mean values using Consortium tools. Open a new window on Casper, the resource provided in tandem with Cheyenne for data analysis and visualization. The process for logging in is the same as for Cheyenne.
   >ssh -X <username>@casper.ucar.edu

5. Load python:
   >module load python/3.6.8
   >ncar_pylib

6. Once you are logged into Casper and have python and the ncar_pylib libraries loaded, you can use the Consortium timeseries scripts to look at the output.

   Move to the sandbox with your code change and copy the timeseries script there.
   >cd ~/cice-tutorial/cice_windtest/
   >cp configuration/scripts/timeseries.py .

   Run the python script pointing to the case logs (e.g. cice.runlog.*)
   >./timeseries.py /glade/scratch/<username>/CICE_RUNS/cice_windtest --bdir /glade/scratch/<username>/CICE_RUNS/cice_base/

This script runs quickly ~1min and will create five *.png files comparing the two simulations over the Arctic and Antarctic. You will see hemispheric mean comparisons of the ice area, ice extent, ice speed, ice volume, and snow volume. Look at these to understand how changing the wind speed has changed the sea ice state.

Note that this script can point to at ANY cice.runlog* files. If you want to run multiple comparisons you will need to copy the *.png files this script creates into a folder of your choosing to keep track of which comparison they correspond to.

7. Questions to consider:
   - What differences do you see in the runs?
   - Do you think the experiments are bit-for-bit (i.e. not answer changing)?
   - Do you think that changing the winds has a climate impact?
   - How can we more rigorously test how changing the winds has impacted our simulation?

Testing your change

You want to test how the changes you made affect the experimental run as compared to a control run, so we will run a regression test and a quality control (QC) test - both are briefly described below.

Regression testing: Running a basic regression test will check if the code changes are bit-for-bit the same (i.e. not answer changing) or not as the base (master) code from which you forked. The basic regression tests you will run today take about 15-20 minutes to run.

QC testing: Sometimes bit-for-bit code changes can not be achieved during development. In this case, running a QC test is necessary to check if the changes impact the science. This QC testing uses statistical properties of sea ice thickness. It requires five years of higher resolution (gx1) daily instantaneous sea ice thickness data to run. The QC test takes much longer to run (~2 hours), so you will evaluate these tests using data we provide for the code change detailed above. However, you will run a QC test tomorrow and steps are listed below for how to set up a QC test that will be useful for tomorrow.

Thorough testing is important if you want to contribute code through a Pull Request (PR) to the Consortium code. The CICE resource index (https://github.com/CICE-Consortium/About-Us/wiki/Resource-Index) provides more information on code development.

Regression test

1. The first step is to run the quick_suite on the cice_master code to generate a baseline set of results. The directions for generating a baseline suite are below. **For this activity we have generated a baseline suite for you and you can move to step 2.**

    a. Start in your unmodified sandbox (cice_master). You can see the list of tests to be carried out here: configuration/scripts/tests/quick_suite.ts

    b. Run the quick suite,
    >qcmd -A UCGD0007 -- ./cice.setup -m cheyenne -e intel --suite quick_suite --testid base0 --tdir ~/cice-tutorial/suites/base0 --bgen cice.base0 --acct UCGD0007 --queue S623186

    cice.setup will automatically create the various cases, build, and submit them all for you. The test takes ~15-20 minutes to run and will print a lot of output to the screen as it runs. This is a good time to take a short break. The final results will be in the directory you specify with the --tdir flag.

    c. Once it has completed, look at the results:
    > cd ~/cice-tutorial/suites/base0
    > ./results.csh | more

    You can do these two steps in the pre-generated baseline directory: /glade/p/cgd/ppc/cice_tutorial_2020/testsuite.base0. You should see the tests have all PASSED.

2. Now we want to run the quick_suite again on the modified code so that we can compare with the original results from the baseline. If we had not pre-generated the baseline results you would need to run that quick_suite before you could run this quick_suite with the modified code.

a. Move to your sandbox with the “windtest” change.
>cd ~/cice-tutorial/cice_windtest/

b. Then run the quick_suite tests again.
>qcmd -A UCGD0007 -- ./cice.setup -m cheyenne -e intel --suite quick_suite --testid windtest --tdir ~/cice-tutorial/suites/windtest --bcmp cice.base0 --bdir /glade/p/cgd/ppc/cice_tutorial_2020/regression/CICE_BASELINE --acct UCGD0007 --queue S623186

Note that the testid differs from the baseline, and you point to the pre-generated baseline results using the --bcmp flag and the --bdir points to where the baseline results are located. This is why you must have a complete baseline suite first.

cice.setup will automatically create the various cases, build, and submit them all for you. The test takes ~15 minutes to run and will print a lot of output to the screen as it runs. This is a good time to take a short break.

c. Look at the results once the suite has completed - check both the status of all jobs and that that your command prompt has returned.
> cd ~/cice-tutorial/suites/windtest
> ./results.csh | more

d. Questions to consider:
- How many tests PASSED? FAILED?
- Which tests failed? Is this what you expected?
- What does this mean?

QC test

1. The first step is to run the cice_master code to generate a baseline set of results. The directions for generating a base suite are below. **For this activity we have generated the baseline test for you, so you can move to step 2.**

   a. Set up the baseline case
   >./cice.setup -m cheyenne -e intel --test smoke -g gx1 -p 36x1 --testid qc_base --tdir ~/cice-tutorial/qc/ -s qc,medium --acct UCGD0007 --queue S623186

   b. Go to the qc_base directory (it will be in the cice_master sandbox)
   >cd ~/cice-tutorial/qc/cheyenne_intel_smoke_gx1_36x1_medium_qc.qc_base

   c. Build the code and submit the case. The output will automatically have the fields and frequency we will need for the QC tests for the full 5 years.

2. The second step is to run an experiment on the cice_windtest code to generate a comparable set of results. **For this activity we have also generated the test results for you because of time constraints, so you can move to step 3.**

   a. Set up the baseline case
   >./cice.setup -m cheyenne -e intel --test smoke -g gx1 -p 36x1 --testid qc_windtest --tdir ~/cice-tutorial/qc/ -s qc,medium --acct UCGD0007 --queue S623186

   b. Go to the qc_windtest directory.
   >cd ~/cice-tutorial/qc/cheyenne_intel_smoke_gx1_36x1_medium_qc.qc_windtest

c. Build the code and submit the case. The run will take a while (~2hrs for the 36x1 configuration), so you should take a break now.

d. Wait for the case to finish and check that it completed successfully. The output will automatically have the fields and frequency we will need for the QC tests for the full 5 years (years 2017-2022). Note: this job can be run at the same time as the baseline test, which is a different workflow than the Regression test.

3. Run the QC test to check if the non bit-for-bit code changes are also climate changing. We will compare the test against the baseline.

   a. Open a new window on Casper, the resource provided in tandem with Cheyenne for data analysis and visualization. The process for logging in is the same as for Cheyenne.
   >ssh -X <username>@casper.ucar.edu

   b. Load python:
   >module load python/3.6.8
   >ncar_pylib

These commands loads python and the NCAR provided python packages, including the numpy and netCDF4 packages that are needed for the following Python script. Note: you, if you are using a shell other than your default login shell, you may need to load python with the following command: >module load ncarenv python/3.6.8

   c. Move to the sandbox with your code change and copy the necessary QC script there.
   >cd ~/cice-tutorial/cice_windtest/
   > cp configuration/scripts/tests/QC/cice.t-test.py .

   d. Run the script but pointing at the directories with the results
   >./cice.t-test.py
/glade/p/cgd/ppc/cice_tutorial_2020/qc/cheyenne_intel_smoke_gx1_36x1_medium_qc.qc_base/history
/glade/p/cgd/ppc/cice_tutorial_2020/qc/cheyenne_intel_smoke_gx1_36x1_medium_qc.qc_windtest/history

This script takes about 5 minutes to run. The output will be written to the screen and will indicate the status of the test, and there are several figures made that will be written to this directory.

   e. Questions to consider:
- Did the QC test pass? Is that expected?
- What does this mean?
- Could you merge this change into the master branch of the Consortium code?
- Look at the figures created by the script? Where, spatially, do you see the biggest differences in the experiments? Is this what you'd expect given the code change we made?
- **Today you did not run the QC test yourself. However, tomorrow morning you will be running a QC test yourself. Look at the directions detailed above to prepare for tomorrow.**

Create Comparison Timeseries

1. Using the directions above for the timeseries.py script, make a timeseries comparing the two QC five year runs. Point to the two directories with the gx1 data used for the QC test (listed

above). Once the figures are complete, compare these windtest results with your gx3 one year experiment results

Advanced code mods

If you have finished all the above experiments, try this more advanced change:

The ksno parameter is the thermal conductivity of snow. The constant value is set in /icepack/columnphysics/icepack_parameters.F90. The default value for ksno is 0.3 W/m/deg, but it can range from 0.03 to 0.65 W/m/deg (see https://doi.org/10.1002/jame.20039 and https://doi.org/10.3189/S0022143000002781)

As above, you should set up a new test branch in which you will change the ksno parameter to something within this observed range. Follow the steps above to create the experiment, run it, and do testing.

Note that if you are developing code and changing code within the icepack directory any changes you make here would NOT be committed back to your Icepack fork. That is because in your CICE sandbox on Cheyenne you are pointing at the Consortium version of Icepack for the submodule. The Git Workflow Guidance page of the wiki (https://github.com/CICE-Consortium/About-Us/wiki/Git-Workflow-Guidance) provides information about the process for how to update Icepack in CICE and sync it properly.