

UKPN Cryospheric Sciences Workshop

Ice Sheet Modelling Practical Session

Introduction

In this session, we're going to download Glimmer-CISM and learn how to compile and run some simple examples using the core ice-sheet model. We'll be using a Linux computing environment, which may be unfamiliar to you. For this exercise, all you need to do is type the commands given at the prompt, but there is a brief guide to Linux available if you're interested.

The basic method of interaction with Linux is to type commands at the *prompt*, which will look something like this:

```
-sh-3.2$
```

The format of the prompt varies, but it usually ends in a dollar sign.

Glimmer-CISM is a complex piece of software written in Fortran, which means we need to *compile* the code before it will run. However, before we can compile Glimmer-CISM, we need to have an important prerequisite in place: the NetCDF library. For now, all you need to know is that NetCDF is used by Glimmer-CISM for reading and writing data files in a standard format. The NetCDF website is here:

```
http://www.unidata.ucar.edu/software/netcdf/
```

Installing NetCDF is relatively straightforward. The source code can be obtained from this location:

```
http://www.unidata.ucar.edu/downloads/netcdf/ftp/netcdf-4.0.1.tar.gz
```

Save this somewhere convenient: I tend to have a directory called *downloads* where I can build software like NetCDF, and another directory called *installs*, where I install the compiled binaries. You can create these directories using the *mkdir* command:

```
mkdir ~/downloads  
mkdir ~/installs
```

Assuming you've downloaded the NetCDF tarball into your *~/downloads* directory, here's what you need to do next...

First, change to the appropriate directory, unpack the tarball, and change to the NetCDF directory:

```
cd ~/downloads  
tar xzvf netcdf-4.0.1.tar.gz  
cd netcdf-4.0.1
```

NetCDF uses standard Linux build tools (autoconf and automake), so installation is relatively straightforward — the basic sequence is to *configure* the build and then *make* it. In addition, we need to specify where we want to install the binaries. We configure the build like this:

```
./configure --prefix=$HOME/installs/netcdf-4.0.1
```

Configure produces lots of diagnostic output. For example, it will tell you which Fortran compiler it finds, and confirm that the f90 interface will be built. Having done this, we can compile NetCDF by running *make*:

```
make
```

NetCDF comes with a thorough test suite — a commendable thing — so we should make use of this before continuing:

```
make check
```

Watch the output to check you get messages saying the test have been passed. If all is well, we can install the binaries in our chosen location:

```
make install
```

We could actually accomplish the build and installation with this command on its own (*make* knows not to try and install something which hasn't been built), but it's convenient to separate the steps, to aid troubleshooting.

Compiling Glimmer-CISM

The first step is to acquire the model code. In common with many open-source projects, the Glimmer-CISM team makes regular *releases* of code — these are numbered snapshots of the code, and represent a stable version of the model, in comparison to the development repository. The latest release can be downloaded from the Glimmer-CISM website:

```
http://developer.berlios.de/projects/glimmer-cism
```

In the centre of the page is a list of latest file releases, comprising not only Glimmer-CISM, but also packages of related code. Click on the *download* link for the latest Glimmer-CISM release; you'll be taken to a list of available packages, where you can click on the highlighted Glimmer-CISM release (version 1.0.18), and save it somewhere convenient. As before, we need to change to the appropriate directory, unpack the tarball and go into the Glimmer-CISM directory:

```
cd ~/downloads
tar xzvf glimmer-1.0.18.tar.gz
cd glimmer-1.0.18
```

Glimmer-CISM also uses *autoconf* and *automake*, and, as with NetCDF, we'll want to specify an install destination using the *--prefix* flag. However, we also need to tell the configure script where to find the NetCDF library, as well as the name of the compiler we want to use:

```
./configure --prefix=$HOME/installs/glimmer-1.0.18 \
--with-netcdf=$HOME/installs/netcdf-4.0.1
```

Now, we can run *make* and *make install*, as before. Glimmer-CISM takes a while to build, but when it's finished you can look in *~/installs/glimmer-1.0.18/bin* to find the executables, and *~/installs/glimmer-1.0.18/lib* to see the libraries which have been built.

The final step is to add the location of the Glimmer-CISM executables to your path variable, so that you can run the model from the command line prompt easily:

```
export PATH=$PATH:$HOME/installs/glimmer-1.0.18/bin
```

EISMINT 1 examples

The EISMINT project was a pioneering model intercomparison exercise involving an international collection of ice sheet models. The project defined a number of benchmark experiments as the basis for the intercomparison, and although other verification and comparison tests have been devised since then, the EISMINT experiments are still relevant and instructive. All EISMINT scenarios are designed to be used with shallow ice models: the first set of scenarios (EISMINT 1) concern *isothermal* models, while the second set (EISMINT 2) introduce thermomechanical coupling and basal sliding.

We'll take a look at the relatively simple EISMINT 1 tests to start with, and move on to the more interesting EISMINT 2 examples after that. One aspect of interest is the validation of Glimmer-CISM against these test scenarios has been published in Rutt et al. (2009), so you can check your results against those in the paper.

The EISMINT 1 and EISMINT 2 experiment scenarios are both available as part of a package called *glimmer-tests*. The latest version is numbered 1.2, and can be downloaded from the

Glimmer-CISM website (see above). Unpack the tarball and `cd` into the resulting directory. If you list the contents of the top-level directory, you'll see directories for various test suites. It's possible to use `configure` and `make` to run each suite of test cases, but it's more instructive to look at them individually. So, let's change to the EISMINT 1 directory and see what we've got:

```
cd EISMINT-1
ls
```

There are configuration files for six experiments, as described in the EISMINT 1 paper (Huybrechts et al., 1996):

```
e1-fm.1.config  e1-fm.3.config  e1-mm.2.config
e1-fm.2.config  e1-mm.1.config  e1-mm.3.config
```

We run any of these files using the `simple_glide` executable which gets built as part of Glimmer-CISM. Assuming the Glimmer-CISM executables are in your `PATH`, you can just run this at the command line:

```
simple_glide
```

You'll be prompted to enter the name of the configuration file, so choose one of the ones listed. There are two basic types of experiment: **fixed margin**, where the same (positive) mass balance is applied to the whole domain, and the shape of the resulting ice sheet is determined by the edge of the domain, and **moving margin**, where the mass-balance field is arranged so that a circular ice sheet forms in the centre of the domain. For each configuration, one scenario has time-constant forcing (**1**), while two scenarios have time-varying forcing (**2** and **3**). As a minimum, you should run the time-constant moving-margin scenario (`e1-mm.1.config`) – this will take about 7-10 minutes. A full description is given in Huybrechts et al. (1996).

Viewing the output

The output from these scenarios is in the form of NetCDF files. The filenames are logical: running `e1-mm.1.config` produces an output file called `e1-mm.1.nc`, as well a log file (`e1-mm.1.config.log`). Once the run has finished, you can look at the NetCDF file it produced. For a quick look at the data, the `ncview` tool is an excellent choice. You can bring up the data on screen with this command:

```
ncview e1-fm.1.nc &
```

The interface is simple: details of how to use `ncview` are given in a separate document.

Finding out more

You can view the most recent version of the Glimmer-CISM documentation on the Glimmer-CISM website. These documents are *mostly* up to date, but they're not perfect.

Exercises

1. Compare your results with those from Huybrechts et al. (1996) and Rutt et al. (2009). Can you suggest causes for any differences you observe?
2. Look at the configuration file, and make sure you understand how it defines the experiment you're running. Note that the forcing is *not* defined in the config file: this is specified in the `simple_glide` code.
3. Adjust the configuration file to change the resolution of one of the experiments. How well does the model converge as the resolution increase? What happens if you don't change the timestep as well? How long does it take before you get bored with waiting for the model run to complete?
4. Look at the output from the moving-margin experiment. The ice sheet is supposed to be

circular in this case: why isn't it? How is this related to the previous question?

EISMINT 2 examples

The EISMINT 2 scenarios follow on directly from EISMINT 1, and are described in Payne et al. (2000). These experiments generate idealised hypothetical ice sheets similar in scale to the first EISMINT experiments, but at double the resolution (61x61 cells at 20km). Additionally, the models are vertically resolved to deal with temperature calculations and thermomechanical coupling.

There are eleven EISMINT 2 idealised scenarios that mainly consider the thermomechanical behaviour of the models, and also introduce a form of sliding. To run the scenarios, just *cd* back up to the main *glimmer-tests* directory, and then to *EISMINT-2*. Again, these experiments are run using the *simple_glide* executable you built earlier. The available configuration files correspond to the experiments described in Payne et al. (2000). Some of the experiments require the output from previous runs to be available before starting.

It takes a relatively long time to run one of these experiments so you will not be able to explore every one. However, each group should have access to enough computing resource to run a few in the time available. Probably the most interesting experiment to look at is Scenario H, where feedbacks between sliding, heat generation and ice rheology cause fluctuating patterns of slow and fast flow to form.

Additional questions and exercises

1. How does the output from Scenario A compare qualitatively with the output from the EISMINT 1 moving margin scenario we looked at earlier? Look at the temperature field in particular.
2. How sensitive is the solution to the relationship between the timesteps of the mechanics and thermodynamics?

Finally: Greenland...

The final example uses real ice sheet geometry: the Greenland Ice Sheet. This model configuration comes from an unpublished extension to the EISMINT programme, and uses what were then (early 2000s) the best available bedrock and ice thickness DEMs, forced by an estimation of annual mass-balance (based on the positive degree-day method).

There isn't time to run the model during this session, so you can download the completed output from this location:

```
http://ggluck.swan.ac.uk/ftp/IanRutt/e3gl\_steady.nc.gz
```

You'll need to use *gunzip* to unzip it:

```
gunzip e3gl_steady.nc.gz
```

Use *ncview* to look at the output. The model is run for 100ka, with output every 5ka. The data demonstrate the importance of model initialisation: the simulated ice sheet eventually settles down into an equilibrium state (somewhat more extensive than the real GrIS), but only after the initial imbalance between geometry, thermal structure and mass-balance has been resolved. The thermal structure in particular is not initialised in a realistic fashion, but is set to annual mean surface air temperature throughout the ice column.

References

Huybrechts P, T Payne, and the EISMINT Intercomparison Group (1996) The EISMINT benchmarks for testing ice-sheet models. *Annals of Glaciology*, **23** 1-14

Payne AJ, P Huybrechts, A Abe-Ouchi, R Calov, JL Fastook, R Greve, SJ Marshall, et al. (2000)

Results from the EISMINT model intercomparison: the effects of thermomechanical coupling. *Journal of Glaciology*, **46**(153), 227-238

Payne AJ, and DJ Baldwin (2000) Analysis of ice-flow instabilities identified in the EISMINT intercomparison exercise. *Ann. Glaciol.*, **30**, 204-210

Rutt, IC, M Hagdorn, NRJ Hulton, and AJ Payne (2009) The Glimmer community ice sheet model. *Journal of Geophysical Research*, **114**, F02004.