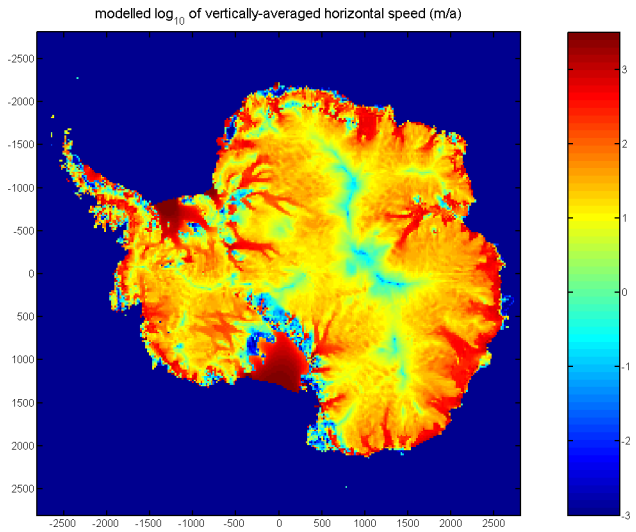
PISM, A PARALLEL ICE SHEET MODEL: USER'S MANUAL AND REFERENCE

ED BUELER*, JED BROWN, AND NATHAN SHEMONSKI

Contents



Date: June 21, 2007. *ffelb@uaf.edu. Manual version based on PISM revision 116 and PETSC release 2.3.3-p2. Get PISM by Subversion: `svn co http://svn.gna.org/svn/pism/trunk pism`; update to latest version by `svn update`.

Acknowledgements

The NASA Cryospheric Sciences Program supported this research with grant NAG5-11371. Dave Covey and Don Bahls have been the best possible system administrators for the machines on which we have developed PISM. Thanks to Art Mahoney for helpful comments on the manual and the installation process.

1. Installation

1. You will need a UNIX system with internet access. A GNU/Linux environment will be easiest but other UNIX versions have been used successfully. Package management systems are useful for installing many of the tools below, *but* neither PISM nor up-to-date PETSc distributions are currently available in the Debian repositories. You will need Python (`http://www.python.org/`) and Subversion (`http://subversion.tigris.org/`) installed, but these are included in all current Linux distributions.

2. To use the graphical output of PISM, which we recommend, you will need an X Windows server *and* the developers' version of X11, which will contain the needed header files.

3. You will need a version of MPI (= *Message Passing Interface*; `http://www-unix.mcs.anl.gov/mpi/`). Your system may have an existing MPI installation in which case the path to the MPI directory will be used when installing PETSc below. Otherwise we recommend that you allow PETSc to download MPICH2 (`http://www-unix.mcs.anl.gov/mpi/mpich2/`) as part of the PETSc configure process (next). In either case, once MPI is installed, you will want to add the MPI `bin` directory to your path so that you can invoke MPI using the `mpiexec` or `mpirun` command. Add the `bin` directory with the statement

```
export PATH=/home/user/mympi/bin:$PATH      (for bash shell)
```

or

```
setenv PATH /home/user/mympi/bin:$PATH      (for csh or tcsh shell).
```

Such a statement can, of course, appear in your `.bashrc`, `.profile`, or `.cshrc` file.

From now on this manual will assume use of `|`bash—.

4. Download PETSc (= *Portable Extensible Toolkit for Scientific computation*, pronounced "pet-see") from `http://www-unix.mcs.anl.gov/petsc/petsc-2/index.html`. You should configure and build PETSc *essentially* as described on the installation page `http://www-unix.mcs.anl.gov/petsc/petsc-2/documentation/installation.html`, but it might be best to read the following comments on the PETSc configure and build process first:

(i) PISM requires a version of PETSc which is `petsc-2.3.3-p2` (patch level 2) or later. The "lite" form of PETSc is fine if you are willing to depend on an internet connection for accessing the PETSc documentation.

(ii) The first step is to download the gzipped tar file for PETSc. Untar it in your home directory or sub; note PETSc should *not* be configured using root privileges. Note that you will need to define the environment variable `PETSC_DIR` before configuring PETSc (next). For instance, if you are in the PETSc directory just untarred, do `export PETSC_DIR=`\`pwd\`. (Note the use of the backprime (*accent-grave*) character, and not the single apostrophe `'`.)

(iii) When you run the configure script in the PETSc directory, the following options are recommended; note PISM builds with shared libraries by default (though other possibilities are illustrated by makefiles in `pism/config/`):

```
$  ./config/configure.py --with-shared --with-c-support
```

(iv) If there is an existing MPI installation, for example at `/home/user/mympi/`, one can point PETSc to it by adding the option `--with-mpi-dir=/home/user/mympi/`. The path used in this option must have MPI executables `mpiexec` and `mpicc` in subdirectory `bin/` and MPI library files in subdirectory `lib/`.

(v) On the other hand, it seems common that one needs to tell PETSc to download MPI into a place it understands, even if there is an existing MPI. If you get messages suggesting that PETSc cannot configure using your existing MPI, you might try `configure.py` with option `--download-mpich=1`.

(vi) Configuration of PETSc for a batch system requires special procedures described at the PETSc documentation site. The advice of an experienced system administrator for the machine in question—i.e. someone very familiar with the batch system—has been found to be useful in this case.

(vii) Configuring PETSc takes many minutes even when everything goes smoothly. A value for the environment variable `PETSC_ARCH` will be reported at the end of the configure process; take note of this value. Note that a previously installed PETSc can be reconfigured with a new `PETSC_ARCH` if necessary.

(viii) After `configure.py` finishes, you will need to `make all test` in the PETSc directory and watch the result. If the X Windows system is functional some example viewers will appear; you may need the X development files for this to work.

(ix) You will want to set the `PETSC_DIR` and (usually) the `PETSC_ARCH` environment variables in your `.profile` or `.bashrc` file. Also remember to add the MPI `bin` directory to your `PATH`. For instance, if you used the option `--download-mpich=1` in the PETSc configure, the MPI `bin` directory will have a path like `$PETSC_DIR/externalpackages/mpich2-1.0.4p1/$PETSC_ARCH/bin/`. Therefore the lines

```
export PETSC_DIR=/home/user/petsc-2.3.3-p2/
export PETSC_ARCH=linux-gnu-c-debug
export PATH=$PETSC_DIR/externalpackages/mpich2-1.0.4p1/$PETSC_ARCH/bin/:$PATH
```

could appear in one of those files.

5. PISM uses NetCDF (= *network Common Data Form*; `http://www.unidata.ucar.edu/software/netcdf/`) for an input and output file format. Install it using the instructions at the NetCDF webpage or using a package management system. You will need the development version (package) which includes the header files (e.g. `netcdf.h`); this is the `xorg-dev` package in the Debian repositories.

6. PISM uses two libraries called "FFTW" and "GSL" for approximating the deformation of the solid earth under ice loads [10]. If you want the functionality of this earth model, which is coupled to the ice flow and which we recommend, install the following or check that it is installed already:

(i) FFTW (= *Fastest Fourier Transform in the West*; `http://www.fftw.org/`). Get the development version with header files (`fftw3-dev` in Debian).

(ii) GSL (= *GNU Scientific Library*; `http://www.gnu.org/software/gsl/`). Get the development version with header files (`libgsl0-dev` in Debian).

If FFTW is not installed, turn off PISM's attempt to build with it by setting the environment variable `WITH_FFTW=0`. If GSL is not installed, turn off PISM's attempt to build with it by setting the environment variable `WITH_GSL=0`. If either of these libraries is absent, all of PISM will work *except* for the bed deformation model described in the paper [10].

See Table 1 for a summary of the dependencies on external libraries, including those mentioned so far.

At this point you have configured the environment which PISM needs. You are ready to build PISM itself.

6. Get the source for PISM by

```
$  svn co http://svn.gna.org/svn/pism/trunk pism
```

A directory called "`pism/`" will be created. Note that in the future when you enter that directory, `svn update` will get the latest revision of PISM.

7. Compile the source. That is, enter directory `pism/` and type "`make`". (Please report any problems you meet at this stage by sending us the output: `ffelb@uaf.edu`.) Several executables, including `pismv`, `pisms`, and `pismr`, should appear in the `pism/bin/` subdirectory. If these appear one can "`make clean`" to clean up the files saved during the build process; specifically `make clean` removes the `.d` and `.o` files.

8. PISM executables can be run most easily by adding the directories `pism/bin/` and `pism/test/` to your `PATH`. The former directory contains the major PISM executables while the latter contains several useful scripts. For instance, this command can be done in the `bash` shell or in your `.bashrc` file:

```
export PATH=/home/user/pism/bin/:/home/user/pism/test/:$PATH
```

You are done with installation at this point. The next few items are recommended as they allow you to observe that PISM is functioning correctly.

10. Try a serial verification run of PISM:

```
$  pismv -test G -y 100
```

If you see some output and a final

```
Writing model state to file 'verify.nc' ... done
```

then PISM completed successfully. Note that at the end of this run you will get measurements of the difference between the numerical result and the exact solution [9].

11. Try the MPI four processor version of the above run:

```
$  mpiexec -n 4 pismv -test G -y 100
```

This should work even if there is only one actual processor on your machine, in which case MPI will run multiple processes on the one processor, naturally. The reported errors should be very nearly the same as the serial run above, but the results should appear faster (if there really are four processors)!

12. Try a verification run on a finer vertical grid while watching the diagnostic views which use Xwindows:

```
$ pismv -test G -Mz 201 -y 2000 -d HTc
```

When using such diagnostic views and `mpiexec` the additional final option `-display :0` is sometimes required to enable MPI to use Xwindows:

```
$ mpiexec -n 2 pismv -test G -Mz 201 -y 2000 -d HTc -display :0
```

13. Run a verification test of the ice stream code:

```
$ pismv -test I -Mx 5 -My 401 -verbose
```

This runs a rather different part of the PISM code and then compares the numerical result to the exact solution appearing in [41].

14. Run a Python script for a basic suite of verifications:

```
$ verifynow.py
```

If you would like us to confirm that PISM is working as expected please save the one page or so of output from this script and send it to us (`ffelb@uaf.edu`). See section 4 for more on PISM verification.

Have fun!

At this stage you can do the EISMINT II tests and some verification tests without further downloads. Sections 6 and 7 describe uses of PISM based on freely available data already on the web. Setting up PISM to model real ice sheets requires some techniques not covered until sections 3 and 7 below. Use of PISM for real ice sheet modelling is something we welcome questions about and will attempt to help with.

A final reminder with respect to installation: Once you have checked out a copy of PISM using Subversion, as in step 6 above, you can update it to the latest version by `svn update` in the `pism/` directory, after which you will want to `make` again, of course.

TABLE 1. Dependencies for PISM, listed alphabetically. Note developers' versions, which include the relevant header files, are needed for FFTW, GSL, and NetCDF.

Library /Program	**Site**	**Required?**	**Comment**
FFTW	`www.fftw.org`	*recommended*	if not present set `WITH_FFTW=0`
GSL	`www.gnu.org/software/gsl`	*recommended*	if not present set `WITH_GSL=0`
LaTeX	`http://www.latex-project.org/`	*optional*	only used for rebuilding manual from source
Matlab	`www.mathworks.com`	*optional*	only used for alternate display of results
MPI	`www-unix.mcs.anl.gov/mpi`	*required*	
NetCDF	`www.unidata.ucar.edu/software/netcdf`	*required*	
PETSc	`www-unix.mcs.anl.gov/petsc/petsc-2/index.html`	*required*	version $\geq$ 2.3.3-p2
Python	`python.org`	*required*	
Subversion	`subversion.tigris.org`	*required*	

2. Getting started

2.1. **Running the EISMINT II tests.** PISM's purpose is the realistic simulation of ice sheets. But real ice sheet simulations require real data. And real data is something we cannot generally distribute under the GNU Public License. (Real ice sheet and ice shelf data *is* available on the web as part of the EISMINT intercomparison efforts; see sections 6 and 7.) So this section describes how PISM does experiment F in the EISMINT II thermomechanical coupling intercomparison [34]. In this experiment one tries to approximate an unstable equilibrium of the thermomechanically coupled differential equations for a shallow, grounded ice sheet. One gets the infamous "spokes" [9, 35]. The prescribed grid has 60 subintervals in each direction, with each subinterval of length 25km, but the vertical grid is not prescribed. Runs are for 200,000 model years.

PISM always allows choice of the grid in three dimensions. A runtime option chooses the number of grid points in each direction; the number of points is one greater than the number of subintervals. We choose the standard 25km grid in the horizontal and use 201 grid points in the vertical for a 25 m (equally-spaced) grid because the computational box is 5000 m for EISMINT II experiment F in PISM. The executable is "`pisms`", a name which has trailing "`s`" for the "simplifed geometry mode" of PISM. Here is a short 2000 year run.

```
$  pisms -eisII F -Mx 61 -My 61 -Mz 201 -y 2000
PISMS (simplified geometry mode)
initializing EISMINT II experiment F ...
  [computational box for ice: ( 1500.00 km) x ( 1500.00 km) x ( 5000.00 m)]
  [grid cell dimensions     : (   25.00 km) x (   25.00 km) x (   25.00 m)]
running EISMINT II experiment F ...
$$$$$      YEAR (+    STEP[N$]):     VOL    AREA    MELTF     THICK0      TEMP0
$$$$$      0.00 (+  0.0000[0 ]):   0.000   0.000    0.000      0.000   223.150
$v$tf     60.00 (+ 60.0000[0m]):   0.017   0.628    0.000     30.000   223.150
$v$tf    120.00 (+ 60.0000[0m]):   0.034   0.628    0.000     60.000   223.538
$v$tf    180.00 (+ 60.0000[0m]):   0.051   0.628    0.000     90.000   223.869
$v$tf    240.00 (+ 60.0000[0m]):   0.068   0.628    0.000    120.000   224.162
$v$tf    300.00 (+ 60.0000[0m]):   0.085   0.631    0.000    150.000   224.425
$v$tf    360.00 (+ 60.0000[0m]):   0.102   0.631    0.000    180.000   224.667
...
$v$tf   1920.00 (+ 60.0000[0m]):   0.545   0.631    0.000    960.000   228.389
$v$tf   1980.00 (+ 60.0000[0m]):   0.562   0.631    0.000    990.000   228.487
$v$tf   2000.00 (+ 20.0000[0e]):   0.568   0.631    0.000   1000.000   228.520
done with run ... Writing model state to file 'simp_exper.nc' ... done.
```

This should have taken less than 30 seconds on any modern computer. In a moment we will address the standard output information provided by PISM, shown above. For now we simply illustrate how to restart and complete the 200,000 year run. Note that the model state was stored in a file with a default name "`simp_exper.nc`". We will see in a moment an example which uses an option to name the output file, and an option to choose its format. The above was a single processor run, but let's suppose we have a four processor machine; the following

should work as stated on a single processor machine under most MPI installations. Let's run things in the background so we can continue to experiment.

```
$  mpiexec -n 4 pisms -eisII F -if simp_exper.nc -y 198000 \
  -o eisIIF200k -of n >> eisIIF.out &
```

We have named the output and specified that it is in NetCDF format. The file `eisIIF200k.nc` will eventually appear, but this run will take at least an hour on a four processor computer. One can view the redirected standard output by `less eisIIF.out` as the job is running in the background. Also "`top`" is a convenient tool to see processor usage during the run. (If one kills the run with `pkill pism`, which sends a `SIGTERM` signal to all `pism` processes, then the run will stop with the model state saved in the named output file `eisIIF200k.nc` even though it will not represent the state at 200,000 years.)

While the above is running in the background, let's actually view the model state during a few time steps by starting from the saved 2000 year state. We use PISM's "diagnostic viewers", which are PETSc viewers working under Xwindows:

```
$  pisms -eisII F -if simp_exper.nc -y 200 -d HTt
PISMS (simplified geometry mode)
initializing from NetCDF format file  simp_exper.nc  ...
  [computational box for ice: ( 1500.00 km) x ( 1500.00 km) x ( 5000.00 m)]
  [grid cell dimensions     : (   25.00 km) x (   25.00 km) x (   25.00 m)]
running EISMINT II experiment F ...
$$$$$       YEAR (+     STEP[N$]):     VOL    AREA    MELTF     THICK0     TEMP0
$$$$$   2000.000 (+  0.00000[0 ]):   0.568   0.631    0.000   1000.000   228.520
$v$tf   2060.000 (+ 60.00000[0m]):   0.585   0.631    0.000   1030.000   228.616
$v$tf   2120.000 (+ 60.00000[0m]):   0.602   0.631    0.000   1060.000   228.710
$v$tf   2180.000 (+ 60.00000[0m]):   0.619   0.631    0.000   1090.000   228.804
$v$tf   2200.000 (+ 20.00000[0e]):   0.625   0.631    0.000   1100.000   228.834
done with run ... Writing model state to file 'simp_exper.nc' ... done.
```

Three figures should appear and be refreshed at each time step. One figure is a map-plane view of thickness, another is a map-plane view of the basal temperature in Kelvin, and third there is a graph of height above the bed versus temperature. When the above 200,000 year run above finishes, on can display the result by `pisms -eisII F -if eisIIF200k.nc -y 200 -d HTt`. In that case the result shown in figure 1 will appear.

At each time step PISM shows a summary of the model state using a few numbers. The format of the summary is

```
$$$$$       YEAR (+     STEP[N$]):     VOL    AREA    MELTF     THICK0     TEMP0
```

The first five columns are flags telling the user which quantities are being updated at each time step. A dollar sign appears if the quantity does not update. From the left the positions are: [`b$`] for bed elevation, [`vV$`] for velocity, [`g$`] for grain size, [`t$`] for temperature and age (which are always updated together), and [`f$`] for surface elevation (i.e. a step of the flow or mass conservation equation has occurred). Regarding velocity, a lower case "`v`" indicates that the 3D velocity field has been updated, e.g. as needed for the advection of temperature, while uppercase

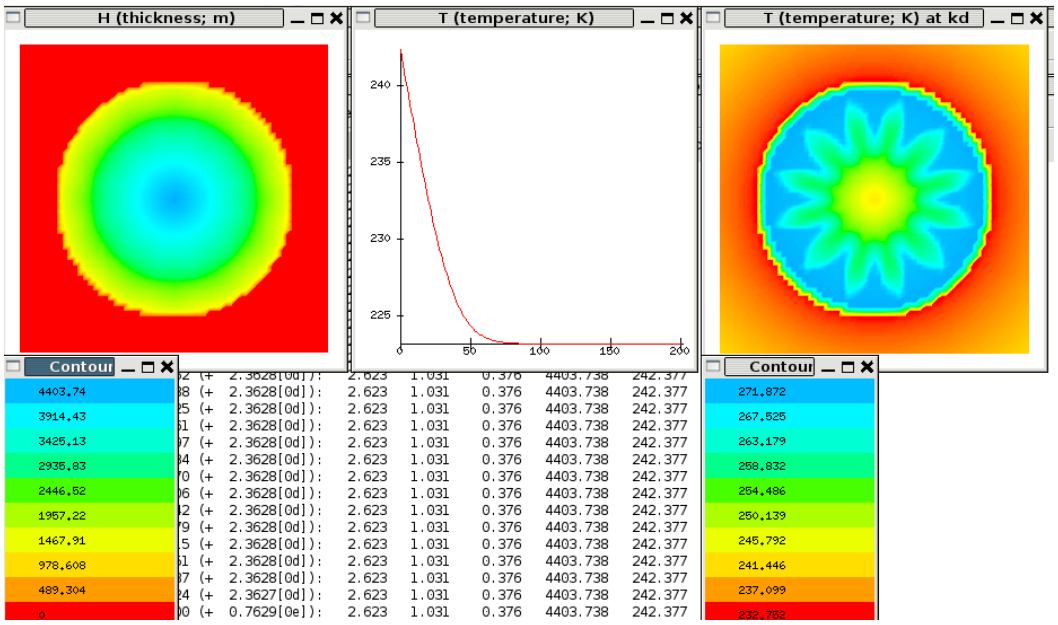


FIGURE 1. Diagnostic figures at the end of a 200,000 year EISMINT II experiment F run, showing the famous spokes.

"`V`" indicates that only the vertically-averaged velocity, and the associated diffusivity, has been updated.

The time ("`YEAR`") and time step ("`STEP`") are in years. In the above examples the time step is 60 years because that is the default maximum time step. A small whole number and a single character flag appear in square brackets after the time step, and these explain what part of the (somewhat elaborate) adaptive time-stepping scheme was used to determine the time step. For instance, "`m`" means that the step was the maximum allowed, "`e`" means that the time step was shortened to hit the end of the specified run, "`d`" means the step was determined by the diffusivity of the flow equation [9], and "`c`" means the CFL condition limited the time step [9, 32]. The small whole number before this single character flag is related to the `-tempskip` mechanism; see sections A and 5 for further description and examples of this mechanism.

The next three columns in the summary are the volume of the ice in $10^6\,\mathrm{km}^3$, the area covered by the ice in $10^6\,\mathrm{km}^2$, and the basal melt fraction, that is, the fraction of the ice area where the basal homologous temperature is above 273.0 (i.e. slightly lower than the triple point). The next two columns "`THICK0`" and "`TEMP0`" are values at the center of the computational domain of the map plane, namely the thickness in meters and the basal absolute temperature in Kelvin.

This summary of the model state can be expanded by using the option `-verbose`. For more on the EISMINT II experiments see section 5.

2.2. **Visualizing the results.** There are three modes for visualizing the various quantities in PISM.

- At runtime, various diagnostic viewers can be specified by options of the form `-d HTt`; see the Diagnostic viewers section below. These viewers are updated at each step and

work under X windows. The format is limited by the style of PETSc viewers. We find that these viewers suffice for quick visualization. Those diagnostic viewers which show "soundings" are controlled by the options `-id`, `-jd`; see section A. Those diagnostic viewers which show slices parallel to the bed are controlled by the option `-kd`.

- The state of the model can be output in NetCDF format using the option `-o foo -of n` to create the NetCDF file `foo.nc`. The resulting NetCDF file can be viewed or modified with the tools described in table 2.

TABLE 2. Tools for viewing and modifying NetCDF files.

Tool	**Site**
`ncview`	`http://meteora.ucsd.edu/~pierce/ncview_home_page.html`
`ncBrowse`	`http://www.epic.noaa.gov/java/ncBrowse/`
CSIRO MATLAB/netCDF interface	`http://www.marine.csiro.au/sw/matlab-netcdf.html`
`NCO` = the NetCDF Operators	`http://nco.sourceforge.net/`

See `http://www.unidata.ucar.edu/software/netcdf/docs/software.html` for additional tools.

- A MATLAB (`http://www.mathworks.com/`) output file, specifically a MATLAB script, can be produced by the options `-o foo -of m`. When executed in MATLAB, the script `foo.m` records several two dimensional quantities, including two dimensional slices of three dimensional quantities like temperature and velocity. The location of these slices is controlled by the options `-id`, `-jd`, `-kd`; see the Runtime options section below.

2.3. **Verification of PISM.** The purpose of the separate executable `pismv` is to establish that the PISM code closely approximates an exact solution to the continuum equations of the model. Thus one can check the numerical "correctness" of PISM at any time, at least in certain simplified situations in which exact solutions are known. The executable `pismv` should certainly be used when changes to the source code occur.

Also, one wants to quantify the limits on *reportable accuracy* from an ice sheet simulation, and the verification tests give some indication of this. Of course the exact solutions have significantly simplified boundary conditions etc., and in some cases they are not very physical.

There are several types of exact solution based tests which verify parts of PISM, including the isothermal shallow ice approximation (SIA) [11], the thermomechanically coupled SIA [9, 8], sliding in the SIA [11], and the MacAyeal equations for ice streams as "dragging ice shelves" [27] with a plastic till assumption [41]. The underlying ice model code executed by `pismv` is identical to that executed by `pismr` and `pisms`, but the command line options are somewhat different.

As noted in the section 1, there is a Python script to execute a short selection of verifications, namely `verifynow.py`. The use of this command is illustrated in section 4.

Here is a basic isothermal verification example, which takes less than a minute on a single processor:

```
$  pismv -test B -ys 422.45 -y 25000 -Mx 31 -My 31
PISMV (verification mode)
initializing Test B ...
```

```
  [computational box for ice: ( 2400.00 km) x ( 2400.00 km) x ( 4000.00 m)]
  [grid cell dimensions     : (   80.00 km) x (   80.00 km) x (  133.33 m)]
running test B ...
$$$$       YEAR (+     STEP[N$]):     VOL    AREA MELTFabs     THICK0     TEMP0
$$$$    422.450 (+  0.00000[0 ]):   4.006   1.773    1.000   3600.000   283.454
$v$f    438.358 (+ 15.90848[0d]):   4.006   2.131   <same>   3587.401    <same>
$v$f    454.816 (+ 16.45735[0d]):   4.006   2.131   <same>   3574.205    <same>
$v$f    471.836 (+ 17.02012[0d]):   4.006   2.131   <same>   3560.570    <same>
$v$f    489.430 (+ 17.59408[0d]):   4.006   2.131   <same>   3546.692    <same>
...
$v$f  25310.273 (+ 60.00000[0m]):   4.006   3.130   <same>   2289.933    <same>
$v$f  25370.273 (+ 60.00000[0m]):   4.006   3.130   <same>   2289.330    <same>
$v$f  25422.450 (+ 52.17742[0e]):   4.006   3.130   <same>   2288.807    <same>
done with run
Actual ERRORS evaluated at final time (relative to exact solution):
geometry  :  prcntVOL  prcntAREA       maxH       avH   relmaxETA    domeH
               0.0083    11.8993   141.3549    8.4164    0.021347   5.3817
Writing model state to file 'verify.nc' ... done
```

Here the PISM numerical results were compared to the exact solution Test B from [11], which is the Halfar solution [17]. Test B is a zero accumulation isothermal shallow ice approximation (SIA) solution. The exact solution was used as the initial condition and then at the end of the run when the numerical result was compared to the exact solution to compute errors. As suggested in [11], we start at a convenient positive time in years "`-ys 422.45`" and do a run of 25000 years. Note that as the sheet became thinner the adaptive time-stepping scheme lengthens the steps up to the (default) maximum time step of 60 years. A grid with 31×31 points in the horizontal is used, matching the EISMINT I choice [22] in the horizontal. No diagnostic viewers were requested though they are available.

At the end of this `pismv` verification run the errors are reported. These are the differences between the values computed numerically on the grid and the known exact solution at the same grid points. In particular, while there are maximum thickness errors of up to 141 meters, the average thickness error is only 8.4 meters at the final time (on this very rough 31×31 point grid). For comparison, the errors for a finer 61×61 grid are smaller:

```
$  pismv -test B -ys 422.45 -y 25000 -Mx 31 -My 31
PISMV (verification mode)
initializing Test B ...
  [computational box for ice: ( 2400.00 km) x ( 2400.00 km) x ( 4000.00 m)]
  [grid cell dimensions     : (   40.00 km) x (   40.00 km) x (  133.33 m)]
running test B ...
$$$$      YEAR (+    STEP[N$]):     VOL    AREA MELTFabs     THICK0     TEMP0
$$$$    422.45 (+  0.0000[0 ]):   3.999   1.762    1.000   3600.000   283.454
$v$f    426.41 (+  3.9647[0d]):   3.999   1.934   <same>   3596.839    <same>
$v$f    430.41 (+  3.9981[0d]):   3.999   1.934   <same>   3593.587    <same>
$v$f    434.44 (+  4.0317[0d]):   3.999   1.934   <same>   3590.254    <same>
...
```

```
$v$f  25341.53 (+ 60.0000[0m]):   3.999   2.965   <same>   2284.171    <same>
$v$f  25401.53 (+ 60.0000[0m]):   3.999   2.965   <same>   2283.570    <same>
$v$f  25422.45 (+ 20.9180[0e]):   3.999   2.978   <same>   2283.361    <same>
done with run
Actual ERRORS evaluated at final time (relative to exact solution):
geometry  :  prcntVOL  prcntAREA       maxH       avH   relmaxETA    domeH
               0.0480     6.4037   133.5960    4.3618    0.009738   0.0638
Writing model state to file 'verify.nc' ... done
```

See [11] for a more complete discussion of this particular test. See the section 4 for a more complete discussion of verification in PISM, including verification of the thermomechanically coupled SIA and of the MacAyeal equations for ice streams.

3. More on usage

3.1. **The PISM coordinate system and grid.** PISM does all simulations in a computational box which is rectangular in the PISM coordinates.

The coordinate system has horizontal coordinates x, y and a vertical coordinate z. The z coordinate is measured positive upward from the base of the ice and it is exactly opposite to the vector of gravity. The surface $z = 0$ is the base of the ice, however, and thus is usually not horizontal in the sense of being parallel to the geoid. The surface $z = 0$ is the base of the ice both when the ice is grounded and when the ice is floating. Vertical lines $(x, y) = (a, b)$, for constant a, b, are generally not at right angles to the planes $z = c$, so the coordinate system is not technically rectilinear.

Bed topography is, of course, allowed. In fact, when the ice is grounded, the true physical vertical coordinate z' is described by $z' = z + b(x, y)$ where $b(x, y)$ is the bed topography. The surface $z' = h(x, y)$ is the surface of the ice. Thus in the grounded case the equation $h(x, y) = H(x, y) + b(x, y)$ applys if $H(x, y)$ is the thickness of the ice.

The computational box can extend downward into the bedrock. As $z = 0$ is the base of the ice, the bedrock corresponds to negative z values.

The extent of the computational box, along with its bedrock extension downward, is determined by four numbers `Lx`, `Ly`, `Lz`, and `Lbz`. The first two of these are half-widths and have units of kilometers when set by options or displayed. The last two are vertical distances in the ice and in the bedrock, respectively, and have units of meters. See the sketch in figure 2.

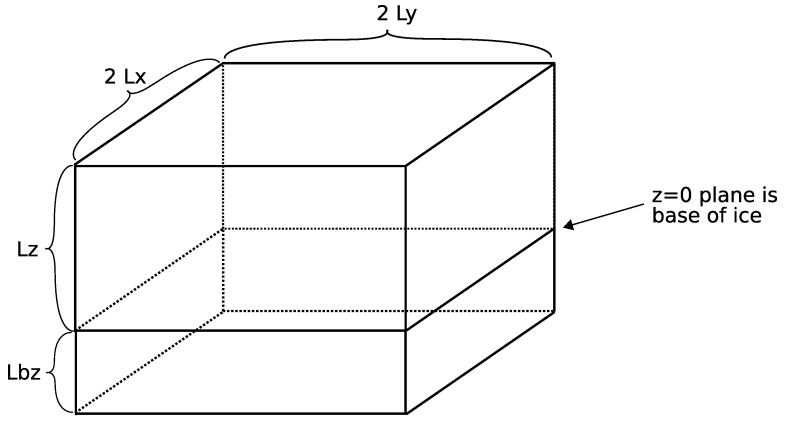


Figure 2. PISM's computational box.

The extent of the computational box for the ice is directly controlled by the options `-Lx`, `-Ly`, and `-Lz` as described in the Runtime options section. As noted `-Lx` and `-Ly` options should include values in kilometers while `-Lz` should be in meters.

The PISM grid covering the computational box is equally spaced in each of the three dimensions. Because of the bedrock extension, the grid of points is described by four numbers, namely the number of grid points in the x direction, the number in the y direction, the number in the z direction within the ice, and the number in the z direction within the bedrock. These are specified by options `-Mx`, `-My`, `-Mz`, and `-Mbz`, respectively, as described in the Runtime options

section. The defaults for these four values are 61, 61, 31, and 1, respectively. Note that `Mx`, `My`, `Mz`, and `Mbz` all indicate the number of grid *points*. Therefore the number of grid *spaces* are, respectively, 60, 60, 30, and 0 (zero) in the default case. Note that the lowest grid point in a column of ice, that is the one at $z = 0$, coincides with the highest grid point in the bedrock. Also `Mbz` must always be at least one.

The distance `Lbz` is controlled by specifying the number `Mbz` of grid points in the bedrock, noting that the vertical spacing `dz` is the same within the ice and within the bedrock. To avoid conflicts, the distance `Lbz` should not be set directly by the user. In particular, `Lbz` = `dz` (`Mbz`-1) while `dz` = `Lz`$/($`Mz` $- 1)$, and so the distance `Lbz` into the bedrock is determined by setting `Lz`, `Mz`, and `Mbz`.

One is allowed to specify the grid when PISM is started *without* a pre-existing model state (i.e. as stored in a NetCDF input file output by PISM). For instance, a EISMINT II experiment F [34] run is

```
$  pisms -eisII F -Mx 61 -My 61 -Mz 101 -y 200000 -o foo
```

Note that PISM (i.e. the executable `pisms`) knows about the size of the computational box appropriate to each of the EISMINT II experiments.

If one initializes PISM from a saved model state then the input model state controls the parameters `Mx`, `My`, `Mz`, and `Mbz`. For instance, the command

```
$  pisms -eisII F -if foo.nc -Mz 201 -y 100
```

will give a warning that "`user option -Mz ignored; value read from file foo.nc`." To change the model grid one must explicitly "regrid", as described next.

3.2. **Regridding.** It is common to want to interpolate a coarse grid model state onto a finer grid or vice versa. For example, one might want to do the EISMINT II experiment F as above, producing `foo.nc`, but then interpolate both the ice thickness and the temperature onto a finer grid. Speaking conceptually, the idea in PISM is that one starts over from the beginning of EISMINT II experiment F on the finer grid, but one extracts the thickness and ice temperature stored in the coarse grid file and interpolates onto the finer grid before proceeding with the actual computation. The transfer from grid to grid is actually quite general—one can go from coarse to fine or vice versa in each dimension x, y, z—but the transfer must always be done by *interpolation* and never *extrapolation*. (An attempt to do the latter should always produce a PISM error.)

Such "regridding" is done using the `-regrid` and `-regrid_vars` commands as in this example:

```
$  pisms -eisII F -Mx 101 -My 101 -Mz 201 -y 1000 \
     -regrid foo.nc -regrid_vars HT -o bar
```

Note one specifies "`HT`" to indicate that the ice thickness and temperature values from the old grid should be interpolated onto the new grid. (One doesn't need to specify the regridding of the bed elevation, which is set identically zero as part of the EISMINT II experiment F description, nor of the ice surface elevation, which is computed as the bed elevation plus the ice thickness at each time step anyway.)

A slightly different use of regridding occurs when "bootstrapping" as described in section 7. Here it is reasonable to have the sequence of commands, run on 8 processors,

```
mpiexec -n 8 pismr -bif init.nc -Mx 141 -My 141 -Mz 101 -Mbz 21 -gk -e 1.2 \
```

```
   -verbose -y 10 -o ant10yr_40km -of n

mpiexec -n 8 pismr -if ant10yr_40km.nc -gk -e 1.2 -no_mass -y 149990 \
   -o ant150k_40km -of n

mpiexec -n 8 pismr -bif init.nc -Mx 281 -My 281 -Mz 201 -Mbz 41 -gk -e 1.2 \
   -regrid ant150k_40km.nc -regrid_vars TBeL -verbose -y 10 -o ant150k_20km -of n
```

Here we bootstrap from an incomplete set of data in `init.nc` and smooth the surface for a short 10 year run on a coarseer 40km grid. Then we do a long simulation to complete 150,000 years on this coarse grid to approximate the temperature and age fields, assuming steady boundary conditions and fixed geometry. Finally we do a brief 10 year run with evolving geometry on a finer 20km grid, but we go back to the original data for the ice thickness; this last stage involves regridding temperature but not thickness, in particular.

3.3. **Understanding and controlling adaptive time-stepping.** Recall that at each time step we get a summary of the model state using a few numbers. The format of the summary is

```
   $$$$$      YEAR (+    STEP[N$]):     VOL    AREA    MELTF     THICK0     TEMP0
```

Here we will explain what appears in the '`(+ STEP[N$])`' part of this summary.

`STEP` is the time step just taken by PISM, in model years. This time step is determined by a somewhat complicated adaptive mechanism. Note that PISM does each step explicitly when numerically approximating mass conservation in the map-plane. This requires that PISM have adaptive time-stepping for stability in the shallow ice approximation regions. Other issues like numerically approximating transport of temperature and age require adaptivity too [9].

Note that most of the time '`N`' will be zero. The exception is when the option `-tempskip` is used. If `-tempskip` M is used, then `N` will be at most M, and will countdown the mass conservation steps when the adaptive scheme determines that a long temperature/age evolution time step, relative to the diffusity controlled time step for mass conservation, would be allowed. To see an example, do:

```
$  pismv -test G -Mx 141 -My 141 -Mz 51 -tempskip 4
```

Table 3 explains the meaning of the one character adaptive-timestepping flag '`$`'.

TABLE 3. Meaning of the adaptive time-stepping flag '`$`' in '`(+ STEP[N$])`'.

Flag	Active adaptive constraint
`c`	3D CFL for temperature/age advection [9]
`d`	diffusivity for SIA mass conservation [9]
`e`	end of prescribed run time
`f`	`-dt_force` set; generally option `-dt_force`, which overrides the adaptive scheme, should not be used
`m`	maximum allowed Δt applies; set with `-maxdt`
`t`	maximum Δt was *t*emporarily set by a derived class; e.g. see effect of deliverables `-time`n in `pisms -ismip H \time`n
`u`	2D CFL for mass conservation in Macayeal regions (where mass conservation is *u*pwinded)

4. Verification

"Verification" is a crucial task for a code as complicated as PISM. It is a task not directly related to the physical behavior of real ice sheets in the sense that it is exclusively mathematical and numerical. It is the task of checking that the predictions of a numerical code are close to the predictions of the continuum model which the numerical code claims to approximate. In particular, one compares exact solutions of the continuum model, if available, to their numerical approximations.

See [11] and [9] for discussion of verification issues for the isothermal and thermomechanically coupled shallow ice approximation (SIA), respectively, and for exact solutions to these models. See [41] for an exact solution to the MacAyeal equations for ice streams using a plastic till assumption.

In PISM there is a separate executable `pismv` which is used for verification. The numerical code which is verified by `pismv` is, however, exactly the same lines of code in exactly the same source files as is run by the non-verification executables `pismr` and `pisms`. (In technical terms, `pismv` runs a derived class of the core class `IceModel`. The core class corresponds to `pismr`.)

Table 4 summarizes the many exact solutions contained within PISM, and shows how to use them individually to verify and to measure accuracy. Note that each of these exact solutions is a solution to a free boundary problem for partial differential equations.

For serious attempts at verification, one must go down a grid refinement path and measure error. For example, the runs

```
pismv -test B -ys 422.45 -y 25000 -Mx 31 -My 31 -Mz 11
pismv -test B -ys 422.45 -y 25000 -Mx 61 -My 61 -Mz 11
pismv -test B -ys 422.45 -y 25000 -Mx 121 -My 121 -Mz 11
pismv -test B -ys 422.45 -y 25000 -Mx 241 -My 241 -Mz 11
```

verify the basic function of the isothermal shallow ice approximation components of PISM. The data produced by these four runs appears in figures 7, 8, 9, and 10 of [11]. Thereby we see that the isothermal mass conservation scheme does a reasonable job of approximating the evolving surface.

For thermocoupled tests one would refine in three dimensions. For example, the runs

```
pismv -test G -maxdt 10.0 -y 25000 -Mx 61 -My 61 -Mz 61
pismv -test G -maxdt 10.0 -y 25000 -Mx 91 -My 91 -Mz 91
pismv -test G -maxdt 10.0 -y 25000 -Mx 121 -My 121 -Mz 121
pismv -test G -maxdt 10.0 -y 25000 -Mx 181 -My 181 -Mz 181
pismv -test G -maxdt 10.0 -y 25000 -Mx 241 -My 241 -Mz 241
pismv -test G -maxdt 10.0 -y 25000 -Mx 361 -My 361 -Mz 361
```

produce the error data in figures 13, 14, and 15 of [9]. (Don't do the last couple of these without a supercomputer! The $361 \times 361 \times 361$ run involves more than 100 million unknowns, updated at each of millions of time steps.)

Note there is a Python script, `verifynow.py`, which runs Tests C, G, and I, and suffices for basic verification of most of the numerical methods in PISM, and is a good test for newly-installed copies of PISM. It takes two options:

TABLE 4. Exact solutions for verification in PISM.

Test	Continuum model tested	Default invocation	Reference
A	isothermal SIA (mass conservation)	`pismv -test A -y 25000`	[11]
B	isothermal SIA (")	`pismv -test B -ys 422.45 -y 25000`	[11]
C	isothermal SIA (")	`pismv -test C -y 15208.0`	[11]
D	isothermal SIA (")	`pismv -test D -y 25000`	[11]
E	isothermal SIA (" and sliding)	`pismv -test E -y 25000`	[11]
F	thermomechanically coupled SIA (mass conservation and energy)	`pismv -test F -y 25000`	[9, 8]
G	thermomechanically coupled SIA (")	`pismv -test G -y 25000`	[9, 8]
H	bed deformation coupled with isothermal SIA	`pismv -test H -y 40034 -bed_def_iso`	[10] [rev 113: BROKEN!!]
I	velocity computation using MacAyeal eqns and plastic till	`pismv -test I -Mx 5 -My $N` (e.g. `$N`≥ 200)	[27, 41]

- "`-n` N" specifies N processors to use (i.e. `mpiexec -n` N is used if $N > 1$); the default is $N = 1$;
- "`-l` L" specifies L levels of refinement; $L = 1, 2, 3, 4, 5$ are allowed values and the default is $L = 3$.

Here is an example run:

```
$ verifynow -n 2 -l 3
  VERIFYNOW using 2 processor(s) and 3 level(s) of refinement
  ++++ verifying isothermal SIA using test C (Mx=My=41,61,81,101,121
       corresponds to dx=dy=50,33.3,25,20,16 km) ++++
  trying "mpiexec -n 2 pismv -test C -Mx 41 -My 41 -Mz 31 -y 15208.0 -verbose 1"
  finished in 10.8591 seconds; reported numerical errors as follows:
    |Actual ERRORS evaluated at final time (relative to exact solution):
    |geometry  :  prcntVOL  prcntAREA       maxH       avH   relmaxETA    domeH
    |               0.7189    12.6255   250.0035   12.0155    0.019159  10.0400
  trying "mpiexec -n 2 pismv -test C -Mx 61 -My 61 -Mz 31 -y 15208.0 -verbose 1"
  finished in 36.7092 seconds; reported numerical errors as follows:
    |Actual ERRORS evaluated at final time (relative to exact solution):
    |geometry  :  prcntVOL  prcntAREA       maxH       avH   relmaxETA    domeH
    |               0.1268     6.5122   224.3489    7.1291    0.012296   4.4014
  trying "mpiexec -n 2 pismv -test C -Mx 81 -My 81 -Mz 31 -y 15208.0 -verbose 1"
  finished in 112.6159 seconds; reported numerical errors as follows:
    |Actual ERRORS evaluated at final time (relative to exact solution):
    |geometry  :  prcntVOL  prcntAREA       maxH       avH   relmaxETA    domeH
    |               0.1970     5.5536   194.2413    4.1316    0.008131   2.1267
  ++++ verifying plastic till ice stream using test I (My=49,193,769,3073,12289
       corresponds to dy=5000,1250,312.5,78.13,19.53 m) ++++
  trying "mpiexec -n 2 pismv -test I -My 49 -Mx 5 -mv_rtol 5e-07 -ksp_rtol 1e-12 -verbose 1"
  finished in  0.9357 seconds; reported numerical errors as follows:
    |Actual ERRORS in velocity relative to exact solution:
    |      maxvector   avvector  prcntavvec      maxu      maxv       avu       avv
    |        23.3329    7.69421     0.98956   23.3329    0.0000    7.6942    0.0000
  trying "mpiexec -n 2 pismv -test I -My 193 -Mx 5 -mv_rtol 5e-07 -ksp_rtol 1e-12 -verbose 1"
  finished in  3.8574 seconds; reported numerical errors as follows:
```

```
  |Actual ERRORS in velocity relative to exact solution:
  |      maxvector   avvector  prcntavvec      maxu      maxv       avu       avv
  |         1.3246    0.44147     0.05678    1.3246    0.0000    0.4415    0.0000
trying "mpiexec -n 2 pismv -test I -My 769 -Mx 5 -mv_rtol 5e-07 -ksp_rtol 1e-12 -verbose 1"
finished in 16.2246 seconds; reported numerical errors as follows:
  |Actual ERRORS in velocity relative to exact solution:
  |      maxvector   avvector  prcntavvec      maxu      maxv       avu       avv
  |         0.0923    0.03117     0.00401    0.0923    0.0000    0.0312    0.0000
++++ verifying thermocoupled SIA using test G (Mx=My=Mz=61,91,121,181,241
     corresponds to dx=dy=30,20,15,10,7.5 km and dz=66.7,44.4,33.3,22.2,16.7 m) ++++
trying "mpiexec -n 2 pismv -test G -Mx 61 -My 61 -Mz 61 -y 25000.0 -verbose 1"
finished in 119.0501 seconds; reported numerical errors as follows:
  |Actual ERRORS evaluated at final time (relative to exact solution):
  |geometry  :  prcntVOL  prcntAREA       maxH       avH   relmaxETA    domeH
  |               2.1076     0.0000    64.3642   19.5506    0.017980   2.8224
  |base temps:        maxT         avT      domeT
  |               1.533177    0.471904   0.069055
trying "mpiexec -n 2 pismv -test G -Mx 91 -My 91 -Mz 91 -y 25000.0 -verbose 1"
finished in 878.9009 seconds; reported numerical errors as follows:
  |Actual ERRORS evaluated at final time (relative to exact solution):
  |geometry  :  prcntVOL  prcntAREA       maxH       avH   relmaxETA    domeH
  |               0.9508     0.0000    29.7903    9.0329    0.007974   0.0095
  |base temps:        maxT         avT      domeT
  |               0.962611    0.238721   0.025394
trying "mpiexec -n 2 pismv -test G -Mx 121 -My 121 -Mz 121 -y 25000.0 -verbose 1"
finished in 3777.5895 seconds; reported numerical errors as follows:
  |Actual ERRORS evaluated at final time (relative to exact solution):
  |geometry  :  prcntVOL  prcntAREA       maxH       avH   relmaxETA    domeH
  |               0.5216     0.0000    28.3494    4.9974    0.004423   0.8286
  |base temps:        maxT         avT      domeT
  |               0.781534    0.140056   0.003918
```

5. Simplified geometry experiments

5.1. **Historical note.** There have been at least three stages of ice sheet model intercomparisons based on simplified geometry experiments since the early 1990s.

EISMINT I [22] was the first of these and involved only the isothermal shallow ice approximation (SIA). "EISMINT" stands for European Ice Sheet Model INTercomparison initiative [CHECK THIS!]. Both fixed margin and moving margin experiments were performed in EISMINT I, and various conclusions were drawn about the several numerical schemes used in the intercomparison. Nonetheless we straightforwardly assert that EISMINT I is superceded by verification using the full variety of exact solutions to the isothermal SIA, especially as described in [11] and noting the "rediscovery" since EISMINT I of the very useful Halfar similarity solution with zero accumulation [17]. For this reason there has been no attempt to support the reproduction of the EISMINT I experiments in PISM. (EISMINT I is a perfectly reasonable task for PISM. It is strictly easier to perform the EISMINT I experiments than the verification relative to tests A, B, C, D, and E described in the previous section, not to mention than the EISMINT II intercomparison or ISMIP-HEINO. But one learns much more about numerical performance through verification with exact solutions [11].)

EISMINT II [34] was a possibly more important, and certainly more controversial, intercomparison which pointed out interesting and surprising properties of the thermocoupled SIA. Here is not the place for a discussion of the interpretations which have followed the EISMINT II results, but [9, 18, 19, 35, 40] each interpret the EISMINT II experiments and/or describe attempts to add more complete physical models to "fix" the perceived shortfalls of ice sheet models (i.e. shortfalls revealed by the EISMINT II results and their interpretations). PISM has built-in support for reproduction of all of the EISMINT II experiments; these are described below.

The ISMIP round of intercomparisons are ongoing at the time of this writing (mid 2007); "ISMIP" stands for Ice Sheet Model Intercomparison Program [CHECK THIS!!]. At this time there are three components of ISMIP namely HOM = Higher Order Models, HEINO = HEInrich events ????, and POLICE = POLar ICE sheets; none of these are completed and published and the last is still in early planning. Of these PISM currently only supports HEINO; see below.

Again as of mid 2007, the PISM authors plan to participate in ISMIP-POLICE and an upcoming Marine Ice Sheet Intercomparison exercise. Watch this space ...

5.2. **EISMINT II in PISM.** There are seven experiments described in the EISMINT II writeup [34]. They are labeled A, B, C, D, F, G, and H. As specified in the writeup, they are 200,000 year runs on a fixed 61×61 horizontal grid, but, as usual, the grid and the length of the run are command line options in PISM. The experiments differ from each other in their various combinations of surface temperature and accumulation parameterizations, and in that experiments H and G involve basal sliding, while the others don't. Some experiments start with zero ice (A,F,G,H) while some start from the end state of other experiments. In particular, experiments B, C, and D start from the final state of experiment A.

The vertical grid is not specified in the EISMINT II writeup. It seems that good simulation of the complex thermomechanically coupled conditions near the base of the ice requires relatively

fine resolution there. Because PISM (currently) has an equally-spaced grid, we recommend the use of about 200 vertical levels. There is no thermal model for the bedrock in EISMINT II, however.

Thus a reasonable experiment A run on four processors is

```
mpiexec -n 4 pisms -eisII A -Mx 61 -My 61 -Mz 201 -y 200000 -o eisIIA
```

Note that once the above run is completed, which might take an hour, something like

```
mpiexec -n 4 pisms -eisII A -if eisIIA.nc -d HTcf -display :0
```

will give a quick view of thickness, basal temperature, the horizontal speed of the ice, and the residual rate of change of the thickness at the final state. A more complete view of the final state might be

```
ncview -minmax all eisIIA.nc
```

Table 5 shows how the other EISMINT II experiments can be done in PISM. As noted, experiments B,C,D start from the saved state of experiment A. Note that "`-y 200000`" is equivalent to "`-y 2e5`" as a specification of run lenghth

TABLE 5. Running the EISMINT II experiments in PISM; command is "`pisms`" plus the given options.

Command: "`pisms`" +	**Relation to experiment A**
`-eisII A -Mx 61 -My 61 -Mz 201 -y 2e5 -o eisIIA`	
`-eisII B -if eisIIA.nc -y 2e5 -o eisIIB`	warmer
`-eisII C -if eisIIA.nc -y 200000 -o eisIIC`	less snow
`-eisII D -if eisIIA.nc -y 200000 -o eisIID`	smaller area of accumulation
`-eisII F -Mx 61 -My 61 -Mz 201 -y 200000 -o eisIIF`	colder; famous spokes [9]
`-eisII G -Mx 61 -My 61 -Mz 201 -y 200000 -o eisIIG`	sliding (regardless of temperature)
`-eisII H -Mx 61 -My 61 -Mz 201 -y 200000 -o eisIIH`	melt-temperature activated sliding

The EISMINT II experiments can be run with various modifications of the default settings. Certainly the grid can be refined. For instance a twice as fine grid in the horizontal is "`-Mx 121 -My 121`". Table 6 lists the optional settings which are particular to the EISMINT II experiments; these options will only work with "`-eisII ?`" set.

TABLE 6. Changing the default settings for the EISMINT II experiments in PISM.

Option	**Default values [expers]**	**Units**	**Meaning**
`-Mmax`	0.5 [ABDFGH], 0.25 [C]	m/a	max value of accumulation rate
`-Rel`	450 [ABFGH], 425 [CD]	km	radial distance to equilibrium line
`-Sb`	10^{-2} [*all*]	(m/a)/km	radial gradient of accumulation rate
`-ST`	1.67×10^{-2} [*all*]	K/km	radial gradient of surface temperature
`-Tmin`	238.15 [ACDGH], 243.15[B], 223.15[F]	K	max of surface temperature

6. Validation

> Two types of errors may be distinguished: modelling errors and numerical errors. Modelling errors arise from not solving the right equations. Numerical errors result from not solving the equations right. The assessment of modelling errors is *validation*, whereas the assessment of numerical errors is called *verification* ... Validation makes sense only after verification, otherwise agreement between measured and computed results may well be fortuitous.
>
> P. Wesseling, *Principles of Computational Fluid Dynamics*, pp. 560–561 [43]

6.1. **Validation of ice shelf numerics relative to Ross ice shelf data.** In [28] a well-known validation of several ice shelf numerical models was performed relative to data from RIGGS (= Ross Ice shelf Geophysical and Glaciological Survey) [5, 4]. The RIGGS data was acquired in the period 1973–1978 by classical surveying techniques. This validation was part of the EISMINT I series of intercomparisons; the intercomparison reported in [22] was also in that series.

Performing this validation requires data from the website `http://homepages.vub.ac.be/~phuybrec/eismint/iceshelf.html`. In particular, these files must be downloaded:

- `111by147Grid.dat`
- `kbc.dat`
- `inlets.dat`

For easiest use, create the directory `pism/eisROSS`, and put these three `.dat` files there.

Note that the data is for a fixed 6.822km grid and *this is an unusual use of PISM in that the grid cannot be adjusted*.

Run this validation with `pisms -ross`. The next several examples are suggested methods for performing the validation. Note that this Ross ice shelf validation can only be run using a single processor; "`mpiexec -n` N `pisms -ross`" generates an error. Table 7 shows options available for this validation; note these options are also described in the Runtime Options section above.

Example 1. Basic run with all info displayed:

```
pisms -ross -d cnmu -pause 10 -showobsvel -verbose
```

Note $\bar{B} = 2.22 \times 10^8$ versus $\bar{B} = 1.9 \times 10^8$ as in [28]. Files `pism/eisROSS/111by147Grid.dat`, `pism/eisROSS/kbc.dat`, and `pism/eisROSS/inlets.dat` must be present (though `-prefix` can be used if they are at a different location).

Example 2. Do same as above but save Matlab results:

```
pisms -ross -verbose -o ross_2p22e8 -of m
```

Note there is no graphical display. The resulting file `ross_2p22e8.m` can be run in Matlab as a script.

The files `README.rossPISM`, `riggs_ELBclean.dat`, and `ross_plot.m` in subdirectory `pism/test/ross/` can be used to produce χ^2 statistic relative to RIGGS data [28], and to give a nice picture. In particular, once the above PISM run finishes and `ross_2p22e8.m` is saved in a location on the

MATLAB path, and once `pism/test/ross/` is added to the MATLAB path, one can enter these commands at the MATLAB prompt:

```
>> ross_2p22e8
...
>> ross_plot
ChiSqr =
       3043.7
max_computed_speed =
       1129.9
```

These results compare well to those in table 1 of [28]. The figures 4 and 3 below are also produced.

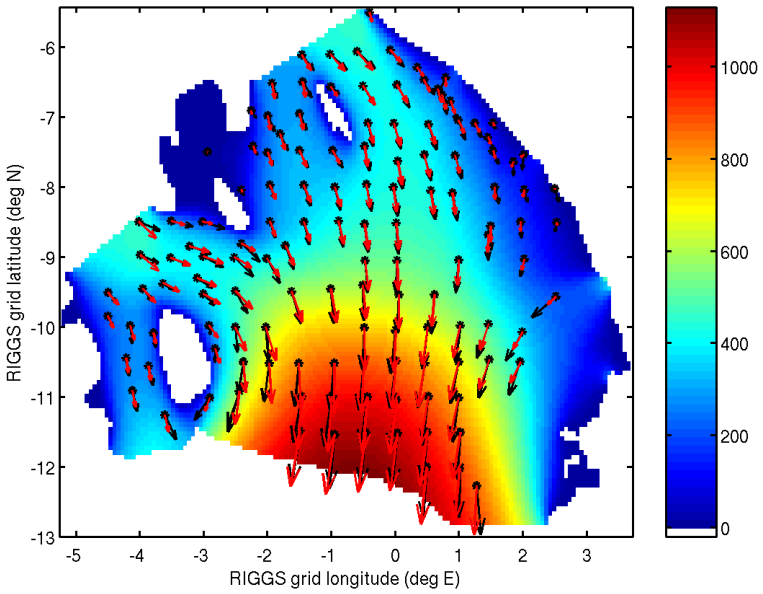


FIGURE 3. Color is speed in m/a. Arrows are observed (black) and computed (red) velocities at RIGGS points.

Example 3. Same as **Example 1**, but asking for a lot more accuracy:

```
pisms -ross -d cnmu -pause 10 -showobsvel -verbose \
  -mv_rtol 1e-7 -ksp_rtol 1e-10
```

In fact one gets nearly the same result, which suggests the default tolerances (`-mv_rtol 1e-4 -ksp_rtol 1e-6`) suffice.

Example 4. Tune across range of values of $\bar{B}$, including [28] value.

```
pisms -ross -verbose -tune 1.7e8,1e7,2.4e8
```

One sees why $\bar{B} = 2.22 \times 10^8$ is used as default value by PISM.

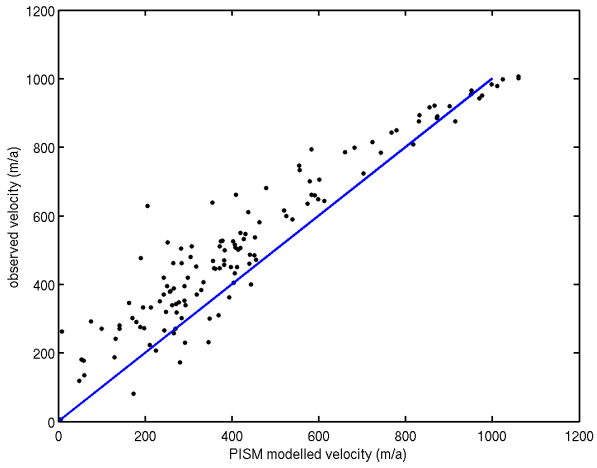


FIGURE 4. Comparison between modelled and observed velocities at RIGGS points; compare figure 2 in [28].

TABLE 7. Options available and/or recommended when validating using EISMINT Ross data; run with `obj/pisms -ross`.

Option	**Explanation/Comments**
`-d cnmu`	most useful way to see what is going on
`-if foo`	NOT allowed! (there is no way to initialize from an input file)
`-Mx`, `-My`	do not adjust
`-o foo -of m`	writes data to foo.m; note that an output file `foo.nc` (from `-o foo -of n`) may also be useful
`-pause N`	pause for N seconds when refreshing viewers
`-prefix foo`	looks for files `111by147Grid.dat`, `kbc.dat`, and `inlets.dat` in `pism/foo/`; files are from `http://homepages.vub.ac.be/~phuybrec/eismint/iceshelf.html`; by default PISM looks in directory `pism/eisROSS/` if `-prefix foo` is not specified
`-ross`	to start the Ross validation; note it works only under executable `pisms`
`-showobsvel`	shows observed (but interpolated) speeds from `111by147Grid.dat`
`-tune x,y,z`	run through $\bar{B}$=`x:y:z` (MATLAB syntax), that is, $\bar{B} = x, x+y, x+2y, \ldots,$ $x + Ny = z$ as hardness parameters; note no spaces in "`x,y,z`"
`-verbose`	shows information on nonlinear iteration and Krylov solve
`-verbose 5`	shows, in particular, which lines were ignored during reads of `???.dat` files, and parameters related to solving the ice shelf equations

7. Realistic ice sheet modelling

7.1. **Obtaining Greenland Data.** As stated earlier, real data is not something we can freely distribute under the GNU Public License. Thus, this section describes the use of a Python script (`eis_green.py`) to convert the EISMINT Greenland data into a NetCDF file useable by PISM.

The Python script `eis_green.py` can be found in the `pism/test` directory. In order to use the script, you must have the following on your system:

- The data files `grid20-EISMINT` and `suaq20-EISMINT` from `http://homepages.vub.ac.be/~phuybrec/eismint/greenland.html` in your `pism/test` directory.
- An installation of Numpy (`http://numpy.scipy.org/`).

The script `eis_green.py` is used by running the following command in the `pism/test` directory:

```
$ python eis_green.py
```

As a result, a NetCDF file named `eis_green.nc` will be created in the same directory. This NetCDF file is the EISMINT Greenland data consisting of values for latitude (`lat`), longitude (`lon`), surface altitude (`S`), bedrock altitude (`b`), and ice thickness (`H`). These values can easily be viewed using Ncview (`http://meteora.ucsd.edu/~pierce/ncview_home_page.html`) with the command:

```
$ ncview eis_green.nc
```

In order to perform somewhat of a checksum on the resulting data, the NCO tools (`http://nco.sourceforge.net/`) can be used. The following command creates a new NetCDF file named `eis_green_check.nc` with a variable called `check`:

```
$ ncap -vs "check=S-(H+b)" eis_green.nc eis_green_check.nc
```

This variable holds the difference of `S` and `b+H`. Viewing this variable will show that `S` is within 1 meter of `b + H` and is considered to be redundant. Thus, this data and will be ignored.

7.2. **Bootstrapping EISMINT Greenland Data.**

7.3. **Running Greenland Experiments Using Bootstrapped Data.**

8. Inside PISM: overviews of models, schemes, and sources

8.1. **The continuum models in PISM.** Significant features of the continuum model approximated by the PISM include:

- The inland ice sheet is modeled with the thermocoupled shallow ice approximation equations [13], and some temperature-activated basal sliding is allowed.
- Ice shelves and ice streams are modeled by shallow equations which describe flow by longitudinal strain rates and basal sliding. These equations are different from the shallow ice approximation. In shelves and streams the velocities are independent of depth within the ice. The equations were originally established for ice shelves [30, 31, 28]. They were adapted for ice streams, as "dragging ice shelves," by MacAyeal [27]; see also [21].
- The regions of grounded ice in which the ice stream model is applied can be determined from mass balance velocities [3], from observed surface velocities, or from a plastic till assumption and the associated free boundary problem [41].
- A three dimensional age field is computed.
- A temperature model for the bedrock under an ice sheet is included.
- Geothermal flux which varies in the map-plane can be used, for instance based on the new Antarctic results in [42] and [29].
- Within the shallow ice sheet regions the model can use the constitutive relation of Goldsby and Kohlstedt [14, 37]. For inclusion in this flow law, grain size can be computed using a age-grain size relation from the Vostok core data [12], for example.
- The Lingle-Clark [10, 26] bed deformation model can be used. It combines a spherical elastic earth and viscous half-space asthenosphere/mantle. It generalizes the better known elastic lithosphere, relaxing asthenosphere ("ELRA") model [16]. This model can be initialized by an observed bed uplift map [10], or even an uplift map computed by an external model like that in [23].

Many of the parts of the model described above are optional. For instance, the Paterson-Budd-Glen [33] flow can replace the Goldsby-Kohlstedt law, a simple isostasy model can be substituted for the more sophisticated one, and so on. Options can be chosen at the command line as described in section A above.

The following features are *not* included in the continuum model, and would (or will) require major additions:

- Inclusion of all components of the stress tensor (i.e. longitudinal stresses within the shallow ice approximation region and additional shear stress components in shelves/streams) through either an intermediate order scheme [6, 19, 40] or the full Stokes equations [13].
- A model for water–content within the ice. In the current model the ice is *cold* and not *polythermal*; compare [15]. On the other hand, in the current model the energy used to melt the ice within a given column, if any, is conserved. In particular, a layer of basal melt water evolves by conservation of energy in the column. This layer can activate basal sliding and its latent heat energy is available for refreezing.
- A model for basal water mass conservation in the map-plane; compare [25].

- A fully spherical Earth deformation model, for example one descended from the Earth model of [36], like the one used to estimate current Antarctic uplift in [23].

8.2. **The numerical schemes in PISM.** Significant features of the numerical methods in PISM include:

- Verification [39] is a primary concern and is built into the code. Nontrivial verifications are available for isothermal ice sheet flow [11], thermocoupled sheet flow [8, 9], the earth deformation model [10], the coupled (ice flow)/(earth deformation) system in an isothermal and pointwise isostasy case [10], and the MacAyeal equations for ice stream flow [41, 7].
- The code is *structurally* parallel. In fact, the PETSc toolkit is used at all levels [1]. PETSc manages the MPI-based communication between processors, and provides an interface to parallel numerical linear algebra and other numerical functions.
- The grid can be chosen at the command line. Regridding can be done at any time, for example taking the result of a rough grid computation and interpolating it onto a finer grid.
- A moving boundary technique is used for the temperature equation which does not stretch the vertical in a singular manner; the Jenssen [24] change of variables is not used.
- The model uses an explicit time stepping method for flow and a partly implicit method for temperature. Advection of temperature is upwinded [32]. As described in [9], reasonably rigorous stability criteria are applied to the time-stepping scheme, including a diffusivity-based criteria for the explicit mass continuity scheme and a CFL criteria [32] for temperature/age advection.
- The local truncation error is generically first order (i.e. $O(\Delta x, \Delta y, \Delta z, \Delta t)$).
- MacAyeal equations [27, 41] are used to determine velocity in the ice shelf and ice stream regions. Like the full non-Newtonian Stokes equations, they are nonlinear and nonlocal equations for the velocity given the geometry of the streams and shelves and given the ice temperature. They are solved by straightforward iteration of linearized equations with numerically-determined (vertically-averaged) viscosity. Either plastic till or linear drag is allowed. As with all of PISM, the numerical approximation is finite difference. The linearized finite difference equations are solved by any of the Krylov subspace methods in PETSc [7, 1].
- The bed deformation model is implemented by a new Fourier collocation (spectral) method [10].
- Implementation is in C++ and is object-oriented. For example, verification occurs in a derived class.

8.3. **The PISM source code.** This ice sheet model is implemented as a collection of C++ object classes, the most central of which is `IceModel`.

More elementary than `IceModel` are the classes

- `IceGrid`, which describes the shape of the grid and parallel layout. This abstraction could be used to streamline transferring model data between different grids.

- `MaterialType`. Various materials are derived from the class `MaterialType` which merely defines a couple physical constants. `IceType` is still an abstract class which defines the interface to ice flow. Concrete classes derived from `IceType` are `ThermoGlenIce` (which uses the EISMINT constants) and `GKIce` as well as `HybridIce`. Similarly, there is `BedrockType` and `OceanType` which merely define associated physical constants.

An example derived class of `IceType` is `HybridIce`, as mentioned. Note that it is difficult, at least, to implement a complete Goldsby-Kohlstedt ice type [14] since the inverse constitutive relation is required for computation of MacAyeal-type ice shelf and dragging ice shelf [27] velocity fields. `HybridIce` is Goldsby-Kohlstedt ice in the interior of the ice sheet and Glen ice in ice streams and shelves.

The methods for `IceModel` are many. They initialize the model (from input data files or from formulas describing various exact solutions), they read user options, they allocate arrays in a distributed manner under PETSc, they compute the terms in the various continuum equations (mass balance, conservation of energy, and velocity), they control diagnostic viewers, they run the central time-stepping, and they write out the model state to files.

A derived class of `IceModel` called `IceCompModel` is used for verification; see the next section. It has additional structures which allows `IceModel` to have compensatory sources and compute initial conditions from, and especially to report errors relative to, known exact solutions [11, 9, 8]. Another derived class used for verification is `IceExactStreamModel`, which implements the exact solution described in [41].

Other derived classes of `IceModel` are `IceEISModel`, `IceHEINOModel`, and `IceROSSModel`. These correspond to simplified geometry experiments and the Ross ice shelf validation described in previous sections of this manual.

There are three established drivers which call constructors and destructors for `IceModel`, namely `run.cc`, `simplify.cc`, and `verify.cc`. These produce the executables `pismr`, `pisms`, and `pismv` described in the rest of this manual. These drivers do no computation but differ in which derived class is constructed and in how certain options are handled. In particular, the driver `run.cc` and its executable `pismr` only use the base class `IceModel`.

The different derived classes invoke the same numerical procedures to handle the various partial differential equations of the continuum model. In the case of `IceCompModel`, this fact is at the heart of the verification mechanism.

8.4. **PETSc: An overview for PISM users.** The PETSc library [1, 2] provides essential support for distributed arrays and linear solvers in a parallel computing environment. "PETSc" stands for "Portable, Extensible Toolkit for Scientific Computation." It is a suite of data structures and routines in C for the scalable parallel solution of partial differential equations and their scientific applications. Large parts of PETSc relate especially to finite-difference-type regular, rectangular grids but PETSc has been used for unstructured grids as well. Documentation for PETSc is available from the web site at `http://www-unix.mcs.anl.gov/petsc/petsc-as/`. PETSc employs the MPI standard for all message-passing communication. PETSc is deliberately at a higher level of abstraction than MPI; PETSc protects the programmer from explicit consideration of message passing.

Most variables in a PETSc program are of newly-defined distributed types including

```
DA   da;
Vec  v;
KSP  ksp;
```

In fact most of the PETSc types merely declare pointers but they should be regarded as objects (abstract data types). The objects must be created with calls to functions like `DACreate2d()`, `VecCreate()`, etc. They should be destroyed when they are not needed with calls to corresponding `Destroy()` functions.

8.4.1. *Distributed arrays and vectors.* PETSc has an abstract date type called a Distributed Array (DA). Objects of type DA contain information about the grid and stencil. They can have information about coordinates, but the code does not use this feature at present. Vectors are created with `DAVecCreate()` and similar. These vectors will be distributed across the processors as indicated by the distributed array.

There are two parameters of note: stencil type and stencil width. The stencil types are `DA_STENCIL_STAR` and `DA_STENCIL_BOX`. They are generalizations of the five point and nine point stencils typical of two dimensional discretizations respectively. In particular, `DA_STENCIL_STAR` indicates that ghosted points (information owned by a different processor) will be needed only along the coordinate axes while `DA_STENCIL_BOX` indicates that ghosted points will be needed in the box shaped region surrounding each point. The stencil width indicates how many points in each direction will be needed. We never need a stencil width greater than 1 and only need BOX style stencils when gradient terms must be evaluated on a staggered grid (h in SIA velocity and $\bar{u}, \bar{v}$ in computation of effective viscosity in Macayeal velocity). Keeping all other two dimensional vectors on a STAR type stencil would reduce the necessary communication slightly, but would complicate the code. For this reason, all two dimensional vectors are kept on a box type distributed array.

The three dimensional distributed arrays are aligned so that they have the same horizontal extent as the associated two dimensional distributed array, but have complete vertical extent. One point of confusion is the redefinition of the x, y, z axes. Contrary to the PETSc default, our z axis changes most rapidly through memory while the x axis changes most slowly. That is, our C style arrays will be addressed as `u[i][j][k]` where `i,j,k` are the coordinate indices in the directions x, y, z respectively.

DA-based vectors can be accessed by `DAVecGetArray()` and restored with `DAVecRestoreArray()`. The resulting pointer should be addressed using normal multidimensional array indexing where values range over the global array.

PETSc DA based vectors can be "local" or "global". Local vectors include space for the ghosted points. That is, when `DAVecGetArray()` is called, the resulting array can be indexed on all the ghosted points. However, all vector operations act only on the local portion. `DALocalToLocalBegin()` and then `DALocalToLocalEnd()` should be called to update the ghost points before they will be needed. Global vectors do not hold ghosted values, but array operations will act on the entire vector. Hence local vectors typically need to be mapped to global vectors before viewing or using in a linear system. This is achieved with `DALocalToGlobal()`.

8.4.2. *Solving linear systems.* PETSc is designed for solving large, sparse systems in a distributed environment. Iterative methods are the name of the game and especially Krylov subspace methods such as conjugate gradients and GMRES. For consistency, all methods use the Krylov subspace interface. For this, the user declares an object of type `KSP`. Various options can be set and the preconditioner context can be extracted. PETSc has an options database which holds command line options. To allow these options to influence the `KSP`, one should call `KSPSetFromOptions()` prior to solving the system. The default method is GMRES(30) with ILU preconditioning.

To solve the system, a matrix must be attached to the `KSP`. The first time `KSPSolve()` is called, the matrix will be factored by the preconditioner and reused when the system is called for additional right hand sides. The default matrix format is similar to the Matlab `sparse` format. Each processor owns a range of rows. Elements in matrices and vectors can be set using `MatSetValues()` and `VecSetValues()`. These routines use the global indexing and can set values on any processor. The values are cached until one calls `MatAssemblyBegin()` followed by `MatAssemblyEnd()` to communicate the values.

In the Macayeal velocity computation, the solution and right hand side vectors are not DA based.

The vector (field) components are interlaced and distributed. This seemed to be the most straightforward method to solve the system (as opposed to using more advanced features intended for multiple degrees of freedom on DA based vectors). This also allows the matrix to have an optimal parallel layout.

8.4.3. *PETSc utility functions.* The `PetscViewer` interface allows PETSc objects to be displayed. This can be in binary to disk, in plain text to the terminal, in graphical form to an X server, to a running instance of Matlab, etc. Typically, one will want to view an entire vector, not just the local portion, so DA based local vectors are mapped to global vectors before viewing.

When viewing multiprocessor jobs, the display may have to be set on the command line, for instance as `-display :0` or similar; this must be given as the final option. For example,

```
mpiexec -n 2 pismv -test C -Mx 101 -My 101 -Mz 31 -y 1000 -d HT -display :0
```

views a two processor run of test C.

PETSc allows the programmer to access command line arguments at any time during program execution. This is preferable to using `getopt.h` for this purpose.

Quite ellaborate error tracing and performance monitoring is possible with PETSc. All functions return `PetscErrorCode` which should be checked by the macro `CHKERRQ()`. Normally, runtime errors print traceback information when the program exits. If this information is not present, you may need to use a debugger which is accessible with the command line options `-start_in_debugger` and `-on_error_attach_debugger`. Also consider options such as `-log_summary` to get diagnostics written to the terminal.

References

[1] S. Balay and eight others, *PETSc users manual*, Tech. Rep. ANL-95/11 - Revision 2.3.2, Argonne National Laboratory, 2006.

[2] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, *Efficient management of parallelism in object oriented numerical software libraries*, in Modern Software Tools in Scientific Computing, E. Arge, A. M. Bruaset, and H. P. Langtangen, eds., Birkhäuser Press, 1997, pp. 163–202.

[3] J. L. Bamber, D. G. Vaughan, and I. Joughin, *Widespread complex flow in the interior of the Antarctic ice sheet*, Science, 287 (2000), pp. 1248–1250.

[4] C. R. Bentley, *Glaciological studies on the Ross Ice Shelf, Antarctica, 1973–1978*, Antarctic Research Series, 42 (1984), pp. 21–53.

[5] ———, *The Ross Ice shelf Geophysical and Glaciological Survey (RIGGS): Introduction and summary of measurments performed*, Antarctic Research Series, 42 (1984), pp. 1–20.

[6] H. Blatter, *Velocity and stress fields in grounded glaciers: a simple algorithm for including deviatoric stress gradients*, J. Glaciol., 41 (1995), pp. 333–344.

[7] J. Brown, *Verifying PISM*. UAF Master's project presentation `www.dms.uaf.edu/~bueler/slides_JedBrown.pdf`, 2006.

[8] E. Bueler and J. Brown, *On exact solutions and numerics for cold, shallow, and thermocoupled ice sheets*. preprint `arXiv:physics/0610106`, 2006.

[9] E. Bueler, J. Brown, and C. Lingle, *Exact solutions to the thermomechanically coupled shallow ice approximation: effective tools for verification*, J. Glaciol., (2007). to appear.

[10] E. Bueler, C. S. Lingle, and J. A. Kallen-Brown, *Fast computation of a viscoelastic deformable Earth model for ice sheet simulation*, Ann. Glaciol., 46 (2007). to appear.

[11] E. Bueler, C. S. Lingle, J. A. Kallen-Brown, D. N. Covey, and L. N. Bowman, *Exact solutions and numerical verification for isothermal ice sheets*, J. Glaciol., 51 (2005), pp. 291–306.

[12] EPICA community members, *Eight glacial cycles from an Antarctic ice core*, Nature, 429 (2004), pp. 623–628. doi: 10.1038/nature02599.

[13] A. C. Fowler, *Mathematical Models in the Applied Sciences*, Cambridge Univ. Press, 1997.

[14] D. L. Goldsby and D. L. Kohlstedt, *Superplastic deformation of ice: experimental observations*, J. Geophys. Res., 106 (2001), pp. 11017–11030.

[15] R. Greve, *A continuum–mechanical formulation for shallow polythermal ice sheets*, Phil. Trans. Royal Soc. London A, 355 (1997), pp. 921–974.

[16] R. Greve, *Glacial isostasy: Models for the response of the Earth to varying ice loads*, in Continuum Mechanics and Applications in Geophysics and the Environment, B. Straughan et al., eds., Springer, 2001, pp. 307–325.

[17] P. Halfar, *On the dynamics of the ice sheets 2*, J. Geophys. Res., 88 (1983), pp. 6043–6051.

[18] R. C. A. Hindmarsh, *Thermoviscous stability of ice-sheet flows*, J. Fluid Mech., 502 (2004), pp. 17–40.

[19] ———, *Stress gradient damping of thermoviscous ice flow instabilities*, J. Geophys. Res., 111 (2006). `doi:10.1029/2005JB004019`.

[20] R. Hooke, *Flow law for polycrystalline ice in glaciers: comparison of theoretical predictions, laboratory data, and field measurements*, Rev. Geophys. Space. Phys., 19 (1981), pp. 664–672.

[21] C. L. Hulbe and D. R. MacAyeal, *A new numerical model of coupled inland ice sheet, ice stream, and ice shelf flow and its application to the West Antarctic Ice Sheet*, J. Geophys. Res., 104 (1999), pp. 25349–25366.

[22] P. Huybrechts et al., *The EISMINT benchmarks for testing ice-sheet models*, Ann. Glaciol., 23 (1996), pp. 1–12.

[23] E. R. Ivins and T. S. James, *Antarctic glacial isostatic adjustment: a new assessment*, Antarctic Science, 17 (2005), pp. 537–549.

[24] D. Jenssen, *A three–dimensional polar ice–sheet model*, J. Glaciol., 18 (1977), pp. 373–389.

[25] J. Johnson and J. L. Fastook, *Northern Hemisphere glaciation and its sensitivity to basal melt water*, Quat. Int., 95 (2002), pp. 65–74.

[26] C. S. Lingle and J. A. Clark, *A numerical model of interactions between a marine ice sheet and the solid earth: Application to a West Antarctic ice stream*, J. Geophys. Res., 90 (1985), pp. 1100–1114.

[27] D. R. MacAyeal, *Large-scale ice flow over a viscous basal sediment: theory and application to ice stream B, Antarctica*, J. Geophys. Res., 94 (1989), pp. 4071–4087.

[28] D. R. MacAyeal, V. Rommelaere, P. Huybrechts, C. Hulbe, J. Determann, and C. Ritz, *An ice-shelf model test based on the Ross ice shelf*, Ann. Glaciol., 23 (1996), pp. 46–51.

[29] C. F. Maule, M. E. Purucker, N. Olsen, and K. Mosegaard, *Heat flux anomalies in Antarctica revealed by satellite magnetic data*, Science, 309 (2005), pp. 464–467.

[30] L. W. Morland, *Unconfined ice-shelf flow*, in Dynamics of the West Antarctic ice sheet, C. J. van der Veen and J. Oerlemans, eds., Kluwer Academic Publishers, 1987, pp. 99–116.

[31] L. W. Morland and R. Zainuddin, *Plane and radial ice-shelf flow with prescribed temperature profile*, in Dynamics of the West Antarctic ice sheet, C. J. van der Veen and J. Oerlemans, eds., Kluwer Academic Publishers, 1987, pp. 117–140.

[32] K. W. Morton and D. F. Mayers, *Numerical Solutions of Partial Differential Equations: An Introduction*, Cambridge University Press, second ed., 2005.

[33] W. S. B. Paterson and W. F. Budd, *Flow parameters for ice sheet modeling*, Cold Reg. Sci. Technol., 6 (1982), pp. 175–177.

[34] A. Payne et al., *Results from the EISMINT model intercomparison: the effects of thermomechanical coupling*, J. Glaciol., 153 (2000), pp. 227–238.

[35] A. J. Payne and D. J. Baldwin, *Analysis of ice–flow instabilities identified in the EISMINT intercomparison exercise*, Ann. Glaciol., 30 (2000), pp. 204–210.

[36] W. R. Peltier, *The impulse response of a Maxwell earth*, Rev. Geophys. Space Phys., 12 (1974), pp. 649–669.

[37] W. R. Peltier, D. L. Goldsby, D. L. Kohlstedt, and L. Tarasov, *Ice–age ice–sheet rheology: constraints from the last Glacial Maximum form of the Laurentide ice sheet*, Ann. Glaciol., 30 (2000), pp. 163–176.

[38] C. Ritz, V. Rommelaere, and C. Dumas, *Modeling the evolution of Antarctic ice sheet over the last 420,000 years: Implications for altitude changes in the Vostok region*, J. Geophys. Res., 102 (1997), pp. 12219–12233.

[39] P. Roache, *Verification and Validation in Computational Science and Engineering*, Hermosa Publishers, Albuquerque, New Mexico, 1998.

[40] F. Saito, A. Abe-Ouchi, and H. Blatter, *European Ice Sheet Modelling Initiative (EISMINT) model intercomparison experiments with first-order mechanics*, J. Geophys. Res., 111 (2006). doi:10.1029/2004JF000273.

[41] C. Schoof, *A variational approach to ice stream flow*, J. Fluid Mech., 556 (2006), pp. 227–251.

[42] N. M. Shapiro and M. H. Ritzwoller, *Inferring surface heat flux distributions guided by a global seismic model: particular application to antarctica*, Earth and Planetary Science Letters, 223 (2004), pp. 213–224.

[43] P. Wesseling, *Principles of Computational Fluid Dynamics*, Springer-Verlag, 2001.

Appendix A. PISM command line options

Much of the behavior of PISM can be set at the command line by options. For example, the command

```
$  pismv -test C -Mx 61 -gk -e 1.2 -o foo
```

includes five options "`-test`", "`-Mx`", "`-gk`", "`-e`", and "`-o`", The first of these options includes a single character argument, the second an integer argument, the third has no argument, the fourth has a floating point argument, and the fifth has a string argument.

The format of the option documentation below is

"`-optionname` [*A*] [`B` *only*]: Description."

Here "A" is the default value and "`B`" is a list of the allowed executables. The option applies to all executables (`pismr`, `pisms`, `pismv`) unless the allowed executables are specifically stated by giving "[`B` *only*]".

As PISM is a PETSc program, all PETSc options are available [1]. See the next section, which recalls some of these PETSc options.

`-adapt_ratio` [*0.12*]: Adaptive time stepping ratio for the explicit scheme for the mass balance equation.

`-bed_def_iso`: Compute bed deformations by simple pointwise isostasy. Assumes that the bed at the starting time is in equilibrium with the load so the bed elevation is equal to the starting bed elevation minus a multiple of the increase in ice thickness from the starting time, roughly: $b(t, x, y) = b(0, x, y) - f[H(t, x, y) - H(0, x, y)]$. Here f is the density of ice divided by the density of the mantle. See Test H in Verification section.

`-bed_def_lc`: Compute bed deformations, caused by the changing load of the ice, using a viscoelastic earth model. Uses the model and computational technique described in [10], based on the continuum model in [26].

`-bif` [`pismr` *only*]: The model can be "bootstrapped" from certain NetCDF files with just the right information. See section 7. Compare `-if`.

`-constant_nu` [*30.0*]: If this option is used then the MacAyeal velocities (see `-mv` below) are computed with a constant viscosity. If this option is not used then the viscosities are computed by a nonlinear iteration. The argument is given in units of MPa a, and the default value is 30 MPa a, the value given in [38].

`-d`: Specifies diagnostic (X Windows) viewers. See Diagnostic viewers section below.

`-datprefix` [*PISM*] [`pisms -ismip H` *only*]: Specify base name for ISMIP-HEINO deliverable `.dat` files. See also `-no_deliver`.

`-dbig`: Specifies larger (about twice linear dimensions) diagnostic viewers. See Diagnostic viewers section.

`-dx`: *Use of this option is not recommended.* `dx` is computed internally as `2*Lx/(Mx-1)`. The user may alter `Lx` or `Mx` at the command line.

`-dy`: *Use of this option is not recommended.* `dy` is computed internally as `2*Ly/(My-1)`. The user may alter `Ly` or `My` at the command line.

`-dz`: *Use of this option is not recommended.* `dz` is computed internally as `Lz/(Mz-1)`. The user may alter `Lz` or `Mz` at the command line.

`-e` [*1.0*]: Flow enhancement factor.

`-eo` [`pismv` *only*]: See section 4.

`-eisII` [*A*] [`pisms` *only*]: Choose single character name of EISMINT II [34] simplified geometry experiment. Allowed values are A, B, C, D, E, F, G, H.

`-force_quarter_year` [`pisms -ismip H` *only*]: ISMIP-HEINO specifies rigid 0.25 year time steps. This will violate both the diffusivity and the CFL parts of the adaptive time-stepping scheme. This overrides the adaptime time-stepping and does 0.25 year time steps anyway.

`-gk` [`pisms,pismr` *only*]: Sets the flow law to Goldsby-Kohlstedt. Same as `-law 4`. See `-law` for more complete option choice of flow law.

`-id`: Sets the x grid index at which a sounding diagnostic viewer (section C) is displayed. The integer argument should be in the range $0, \ldots,$ `Mx` $- 1$. The default is (`Mx` $- 1)/2$ at the center of the grid. For example, `pismv -test G -d t -id 10`.

`-if`: The model can be restarted from either a PETSc binary file written by the model, e.g. `foo.pb` from `-o foo` or `-o foo -of p`, or from a NetCDF file `foo.nc` written by the model, e.g. `foo.nc` from `-o foo -of n`. Compare `-bif`.

`-ismip` [*H*] [`pisms` *only*]: Choose ISMIP simplified geometry experiment. Only "H" for HEINO is allowed at this time.

`-isoflux`: Isothermal runs can be done by two methods. One may simply initialize with a constant temperature field and then turn off temperature evolution (see `-no_temp`); in this case the horizontal velocity and the horizontal mass flux are computed by a vertical numerical integral. If the option `-isoflux` is used then the mass flux is computed by the usual isothermal, Glen formula without reference to the temperature field or the chosen flow law.

`-jd`: Sets the y grid index at which a sounding diagnostic viewer (section C) is displayed. The integer argument should be in the range $0, \ldots,$ `My` $- 1$. The default is (`My` $- 1)/2$ at the center of the grid. For example, `pismv -test G -d t -jd 10`.

`-kd` [*0*]: Sets the z grid index at which map-plane diagnositic views are shown. Compare `pismv -test G -Mz 101 -d T` and `pismv -test G -Mz 101 -d T -kd 50`. See section C.

`-law` [*0*]: Allows choice of thermocoupled flow law. The options are in table 8 below. Note that a "flow law" here means the function $F(\sigma, T, P, d)$ in the relation

$$\dot{\epsilon}_{ij} = F(\sigma, T, P, d)\, \sigma'_{ij}$$

where $\dot{\epsilon}_{ij}$ is the strain rate tensor, σ'_{ij} is the stress deviator tensor, T is the ice temperature, $\sigma^2 = \frac{1}{2}\|\sigma'_{ij}\|_F$ so σ is the second invariant of the stress deviator tensor, P is the pressure, and d is the grain size. That is, we are addressing isotropic flow laws only, and one can choose the scalar function. Note that the inverse form of such a flow law in needed for ice shelves and ice streams:

$$\sigma'_{ij} = 2\nu(\dot{\epsilon}, T, P, d)\, \dot{\epsilon}_{ij}$$

Here $\nu(\dot\epsilon, T, P, d)$ is the effective viscosity. The need for this inverse form of the flow law explains the "hybrid" law `-law 4` (or `-gk`).

TABLE 8. Choosing the flow law.

Flow Law	**Option**	**Comments and Reference**
Paterson-Budd law	`-law 0`	Fixed Glen exponent $n = 3$. There is a split "Arrhenius" term $A(T) = A\exp(-Q/RT^*)$ where $(A = 3.615\times10^{-13}\,\mathrm{s^{-1}\,Pa^{-3}}, Q = 6.0\times10^4\,\mathrm{J\,mol^{-1}})$ if $T^* < 263$ K and $(A = 1.733\times10^{3}\,\mathrm{s^{-1}\,Pa^{-3}}$, $Q = 13.9\times10^4\,\mathrm{J\,mol^{-1}})$ if $T^* > 263$ K and where T^* is the homologous temperature [33].
Cold part of Paterson-Budd	`-law 1`	Regardless of temperature, the values for $T^* < 263$ K in the Paterson-Budd law above apply. This is the flow law used in the thermomechanically coupled exact solutions Tests **F** and **G** described in [9, 8] and run by `pismv -test F`, `pismv -test F`.
Warm part of Paterson-Budd	`-law 2`	Regardless of temperature, the values for $T^* > 263$ K in Paterson-Budd apply.
Hooke law	`-law 3`	Fixed Glen exponent $n = 3$. Here $A(T) = A\exp(-Q/(RT^*) + 3C(T_r - T^*)^\kappa)$; values of constants as in [20, 35].
Hybrid of Goldsby-Kohlstedt and Paterson-Budd	`-law 4`	Goldsby-Kohlstedt law with a combination of exponents from $n = 1.8$ to $n = 4$ [14] in grounded shallow ice approximation regions. Paterson-Budd flow for ice streams and ice sheets. See mask for SIA versus stream versus shelf by `-d m`.

`-Lx`: The x direction half-width of the computational box, in kilometers. See section 3.

`-Ly`: The y direction half-width of the computational box, in kilometers. See section 3.

`-Lz`: The height of the computational box for the ice (excluding the bedrock thermal model part). See section 3.

`-maxdt` [*60.0*]: The maximum time-step in years. The adaptive time-stepping scheme will make the time-step shorter than this as needed for stability, but not longer. See section 3.

`-Mbz` [*1*]: Number of grid points in the bedrock for the bedrock thermal model. The highest grid point corresponds to the base of the ice $z = 0$, and so `Mbz`> 1 is required to actually have bedrock thermal model. Note this option is unrelated to the bed deformation model (glacial isostasy model); see option `-bed_def` for that.

`-Mmax` [`pisms -eisII` *only*]: Set value of $M_{\max}$ for EISMINT II.

`-mu_sliding` [*3.17e-11*]: The sliding law parameter in SIA regions of the ice.

`-mv`: Use the MacAyeal-Morland equations [27] for ice shelves and dragging ice shelves (i.e. ice streams) where so-indicated by the mask. To view the mask use `-d m`.

`-mv_eps` [*1.0e15*]: The numerical scheme for the MacAyeal-Morland equations computes an effective viscosity which which depends on velocity and temperature. After that computation, this constant is added to the effective viscosity (to keep it bounded away from zero). The units are kg $\mathrm{m}^{-1}\,\mathrm{s}^{-1}$.

In fact there is a double regularization by default because the Schoof regularization mechanism described in equation (4.1) of [41] is also used. Turn off this lower bound mechanise by `-mv_eps 0.0` to exclusively use the Schoof regularization mechanism; see `-reg_vel_schoof` and `-reg_length_Schoof` below. Note `mv_eps` is set to zero automatically when running `pismv -test I`.

`-mv_rtol` [*1.0e-4*]: The numerical scheme for the MacAyeal-Morland equations does a nonlinear iteration wherein velocities (and temperatures) are used to compute a vertically-averaged effective viscosity which is used to solve the equations for horizontal velocity. Then the new velocities are used to recompute an effective viscosity, and so on. This option sets the relative change tolerance for the effective viscosity.

In particular, the nonlinear part of the iteration requires that successive values $\nu^{(k)}$ of the vertically-averaged effective viscosity satisfy

$$\frac{\|(\nu^{(k)} - \nu^{(k-1)})H\|_2}{\|\nu^{(k)}H\|_2} \le \text{mv\_rtol}$$

in order to end the iteration with $\nu = \nu^{(k)}$. See also `-ksp_rtol`, a PETSc option below, as one may want to require a high relative tolerance for the linear iteration as well.

`-Mx` [*61*]: Number of grid points in x horizontal direction.

`-My` [*61*]: Number of grid points in y horizontal direction.

`-Mz` [*31*]: Number of grid points in z (vertical) direction.

`-no_mass`: Forces PISM not to change the ice thickness. No time steps of the mass conservation equation are computed.

`-noreport` [`pismv` *only*]: Do not report errors at the end of a verification run.

`-no_spokes` [*0*]: The strain heating term can be smoothed by non-physical averaging the neighboring horizontal neighbors (those which are within the ice) [9]. The integer parameter controls the number of neighboring grid points over which the average is computed. For instance, `-no_spokes 0` is no smoothing while `-no_spokes 3` is smoothing over the 3-neighborhood of horizontal grid points, that is, over a distance of `3*dx`.

`-no_temp`: Do not change temperature or age values within the ice. That is, do not do time steps of energy conservation and age equation.

`-o` [*unnamed.pb*]: Give name of output file: `-o foo` writes an output file named `foo.nc`. See the description of option `-if`. Default name is `unnamed.nc` under `pismr`, `simp_exper.nc` under `pisms`, and `verify.nc` under `pismv`.

`-ocean_kill`: If used with input from a NetCDF initialization file which has ice-free ocean mask, will zero out ice thicknesses in areas that were ice-free ocean at time zero. Has no effect when used in conjunction with `-no_mass`.

`-of` [*p*]: Format of output file(s). Possible values are `n` for model state written to a NetCDF file `foo.nc` and `m` for selected variables written to an ASCII Matlab file `foo.m`. PISM can be restarted using `-if` from the output `.nc` files. Multiple files can be written, for instance, `-o foo -of nm` writes both `foo.nc` and `foo.m`.

`-pause` [`pisms -ross,pismv -test I` *only*]: Pause for given number of seconds at the end of the run when the results are displayed.

`-prefix` [`pisms -ross` *only*]: Set the data file prefix for the EISMINT Ross ice shelf validation [28]. See `-ross`.

`-plastic_reg` [*0.01 m/a*] [`pismv -test I` *only*]: Set the value of ϵ regularization of plastic till; this is the second "ϵ" in formula (4.1) in [41]. See `src/iceExactStreamModel.cc`.

`-regrid`: See section 3.

`-regrid_vars`: See section 3.

`-reg_length_schoof` [*1000 km*]: Set the "L" in formula (4.1) in [41]. To use the regularization described by Schoof, one must set `-mv_eps 0.0` to turn off the other regularization mechanism, otherwise there is a double regularization.

`-reg_vel_schoof` [*1 m/a*]: Set the *first* "ϵ" in formula (4.1) in [41]. To use the regularization described by Schoof, one must set `-mv_eps 0.0` to turn off the other regularization mechanism, otherwise there is a double regularization. Use `-plastic_reg` above to set the second "ϵ" in formula (4.1) of [41].

`-Rel` [`pisms -eisII` *only*]: Set value of R_{el} for EISMINT II.

`-ross` [`pisms` *only*]: Run the EISMINT Ross ice shelf validation [28]. Requires data from `http://homepages.vub.ac.be/~phuybrec/eismint/iceshelf.html`.

`-run` [*ST*] [`pisms -ismip H` *only*]: ISMIP-HEINO has several run names: ST, T1, T2, B1, B2, S1, S2, S3.

`-Sb` [`pisms -eisII` *only*]: Set value of S_b for EISMINT II.

`-showobsvel` [`pisms -ross -d c` *only*]: Show the map of observed velocity for the EISMINT Ross ice shelf validation [28]. See `-ross` and `-pause`.

`-ST` [`pisms -eisII` *only*]: Set value of S_T for EISMINT II.

`-tempskip` [*1*]: Number of mass-balance steps to perform before a temperature step is executed. A maximum value of `-tempskip 5` is recommended to avoid too many CFL violations.

`-test` [*A*] [`pismv` *only*]: Choose single character name of verification test. Allowed values are A, B, C, D, E, F, G, H, I. See the Verification section.

`-time1` [`pisms -ismip H` *only*]: ISMIP-HEINO requires writing 2D planform `.dat` files at four times in the interval $[150, 200] \times 10^3$ years. This specifies the time. Note `test/showheino.m` will compute the time from the other `.dat` files.

`-time2` [`pisms -ismip H` *only*]: See `-time1` explanation.

`-time3` [`pisms -ismip H` *only*]: See `-time1` explanation.

`-time4` [`pisms -ismip H` *only*]: See `-time1` explanation.

`-Tmin` [`pisms -eisII` *only*]: Set value of $T_{\min}$ for EISMINT II.

`-tune` [`pisms -ross` *only*]: Tune the vertically averaged hardness $\bar{B}$ in the EISMINT Ross ice shelf validation [28]. See `-ross`. A range of values can be given; see `src/iceROSSModel.cc`.

`-verbose`: Increased verbosity of standard output. Can be given without argument ("`-verbose`") or with a level which is one of the integers 0,1,2,3,4,5 ("`-verbose 2`"). The full scheme is given in table 9.

TABLE 9. Controlling the verbosity level to standard out.

Level	**Option**	**Meaning**
0	`-verbose 0`	never print to standard out *at all*
1	`-verbose 1`	less verbose than default
2	[`-verbose 2`]	default verbosity
3	`-verbose 3`	somewhat verbose; expanded description of grid at start
	or `-verbose`	and expanded information in summary
4	`-verbose 4`	
	or `-vverbose`	more verbose
5	`-verbose 5`	
	or `-vvverbose`	maximally verbose

`-vverbose`: See table 9.

`-vvverbose`: See table 9.

`-y` [*1000*]: Number of model years to run.

`-ye`: Model year at which to end the run.

`-ys` [*0*]: Model year at which to start the run.

Appendix B. PETSC command line options (for PISM users)

All PETSc programs accept command line options which control the manner in which PETSc distributes jobs among parallel processors, how it solves linear systems, what additional information it provides, and so on. The PETSc manual (`http://www.mcs.anl.gov/petsc/petsc-as/snapshots/petsc-current/docs/manual.pdf`) is the complete reference on these options. Here we list some that are perhaps most useful to PISM users.

`-da_processors_x`: Number of processors in x direction.

`-da_processors_y`: Number of processors in y direction.

`-display` [*A*]: s a *final* option, `-display :0` seems to frequently be needed to let PETSc use Xwindows when running multiple processes.

`-help`: Gives PISM help message and then a brief description of many PETSc options.

`-info`: Gives excessive information about PETSc operations during run. Option `-verbose` for PISM (above) is generally more useful, except possibly for debugging.

`-ksp_rtol` [*1e-5*]: For solving the ice stream and shelf equations with high resolution on multiple processors, it is recommended that this be tightened. For example,

```
$  mpiexec -n 8 pismv -test I -Mx 5 -My 769
```

works poorly on a certain machine, but

```
$  mpiexec -n 8 pismv -test I -Mx 5 -My 769 -ksp_rtol 1e-10
```

works fine.

`-ksp_type` [*gmres*]: Based on one processor evidence from `pismv -test I`, the following are possible choices in the sense that they work and allow convergence at some reasonable rate: `cg`, `bicg`, `gmres`, `bcgs`, `cgs`, `tfqmr`, `tcqmr`, and `cr`. It appears `bicg`, `gmres`, `bcgs`, and `tfqmr`, at least, are all among the best.

`-log_summary`: At the end of the run gives a performance summary and also a synopsis of the PETSc configuration in use.

`-pc_type` [*ilu*]: Several options are possible, but for solving the ice stream and shelf equations we recommend only `bjacobi`, `ilu`, and `asm`. Of these it is not currently clear which is fastest; they are all about the same for `pismv -test I` with high tolerances (e.g. `-mv_rtol 1e-7 -ksp_rtol 1e-12`).

`-v`: Show version number of PETSc.

Appendix C. PISM runtime diagnostic viewers

Many basic views of the changing state of the ice model are available at the command line by using the options "`-d`" and "`-dbig`" with additional arguments. For instance:

```
$  pismv -test G -d hTf -dbig c
```

shows a map-plane views of surface elevation ("`h`"), temperature at the level specified by `-kd` ("`T`"), rate of change of thickness ("`f`") and of vertically-averaged horizontal ice speed ("`c`").

The option `-d` is followed by a space and then a list of single-character names of the diagnositic viewers. The option `-dbig` works exactly the same way, with the same list of single-character names available. The bigger viewers take precedence, so that "`-d hT -dbig T`" shows only two viewers, namely a regular size viewer for surface elevation and a larger viewer for temperature.

The single character diagnostic viewer names are:

`a`: Map-plane view of accumulation in meters per year.

`b`: Map-plane view of bed elevation in meters above sea level.

`c`: Map-plane view of horizontal speed, namely the absolute value of the vertically-averaged horizontal velocity. Displayed as log base ten of speed in meters per year.

`D`: Map-plane view of diffusivity coefficient D in mass balance equation in m^2/s. Meaningful only in regions of shallow ice flow.

`E`: Map-plane view of age of the ice, in years.

`e`: Age in a vertical column (sounding); in years. See `-id`, `-jd` to set sounding location.

`F`: Map-plane view of basal geothermal heat flux, in milliWatts per meter squared.

`f`: Map-plane view of thickening rate of the ice, in meters per year.

`G`: Map-plane view of grain size, in millimeters. Displayed at chosen elevation above base; see option `-kd`.

`g`: Grain size in a vertical column (sounding); in millimeters. See `-id`, `-jd` to set sounding location.

`H`: Map-plane view of thickness in meters.

`h`: Map-plane view of ice surface elevation in meters above sea level.

`i`: Map-plane view of vertically-averaged effective viscosity times thickness; on i offset grid. Only meaningful in ice streams and shelves.

`j`: Map-plane view of vertically-averaged effective viscosity times thickness; on i offset grid. Only meaningful in ice streams and shelves.

`k`: Iteration monitor for the Krylov subspace routines (KSP) in Petsc. Shows norm of residual versus iteration number.

`L`: Map-plane view of basal melt water thickness in meters.

`l`: Map-plane view of basal melt rate in meters per year.

`m`: Map-plane view of mask for flow type: **1** = grounded shallow ice sheet flow, **2** = dragging ice shelf, **3** = floating ice shelf.

`N`: Produces two viewers, namely the i offset and j offset grid versions of the rate of change of the vertically-averaged effective viscosity times thickness. Only meaningful in ice streams and shelves.

`n`: Map-plane view of the log base ten of the vertically-averaged effective viscosity times thickness on the regular grid. Only meaningful in ice streams and shelves.

`P`: *ONLY AVAILABLE for* `pismv`. Map-plane view of comPensatory heating term Σ_C in thermocoupled verification tests F and G. Displayed at chosen elevation above base; see option `-kd`.

`p`: Map-plane view of bed uplift rate in meters per year.

`q`: Map-plane view of basal sliding speed. Displayed as log base ten of speed in meters per year.

`R`: Map-plane view of basal frictional heating in milliWatts per meter squared.

`r`: Map-plane view of surface temperature in Kelvin.

`S`: Map-plane view of strain heating term Σ in temperature equation, in Kelvin per year. Displayed at chosen elevation above base; see option `-kd`.

`s`: Strain heating term Σ in vertical column (sounding). See `-id`, `-jd` to set sounding location.

`T`: Map-plane view of absolute ice temperature in Kelvin. Displayed at chosen elevation above base; see option `-kd`.

`t`: Absolute ice temperature in vertical column (sounding). See `-id`, `-jd` to set sounding location.

`u`: Map-plane view of vertically averaged horizontal velocity in the x-direction; in meters per year.

`v`: Map-plane view of vertically averaged horizontal velocity in the y-direction; in meters per year.

`x`: x-component of horizontal velocity in vertical column (sounding). See `-id`, `-jd` to set sounding location.

`X`: Map plane view of x-component of horizontal velocity, in meters per year. Displayed at chosen elevation above base; see option `-kd`.

`y`: y-component of horizontal velocity in vertical column (sounding). See `-id`, `-jd` to set sounding location.

`Y`: Map plane view of y-component of horizontal velocity, in meters per year. Displayed at chosen elevation above base; see option `-kd`.

`z`: Vertical velocity (w-component of velocity) in vertical column (sounding). See `-id`, `-jd` to set sounding location.

`Z`: Map plane view of vertical velocity, in meters per year. Displayed at chosen elevation above base; see option `-kd`.