

# *PISM*, A PARALLEL ICE SHEET MODEL:

## INSTALLATION MANUAL

CONSTANTINE KHROULEV  
ED BUELER

### CONTENTS

Introduction	2
1. Libraries and programs needed by PISM	3
2. Installation Cookbook	4
2.1. Installing PISM on a Debian-based Linux system	4
2.2. Generic PISM installation, using PETSc source code	6
2.3. Installing PISM on Mac OS X	9
2.4. If nothing else works: modifying makefiles	10
3. Quick tests of the installation	11
4. Installing Python packages	12
5. Rebuilding PISM documentation	13

---

*Date:* September 29, 2009. Get PISM stable version by

`svn co http://svn.gna.org/svn/pism/tags/stable0.2 pism0.2`

Based on PISM revision 792 (stable0.2) and PETSC release 2.3.3 or 3.0.0.

## INTRODUCTION

This *Installation Manual* describes how to download the PISM source code and install PISM and the libraries it needs. Information about PISM, including a *User's Manual*, is online at

`www.pism-docs.org`

The fastest path to a fully functional PISM installation is to use a Linux system with a Debian-based package system (e.g. Ubuntu), subsection 2.1, and then install Python packages following section 4.

*WARNING:* PISM is an ongoing project. Ice sheet modeling is complicated and not mature (in 2008, anyway). Please don't trust the results of PISM or any other ice sheet model without a fair amount of exploration. Also, please don't expect all your questions to be answered here. Write to us with questions:

`help@pism-docs.org`

*Copyright (C) 2004–2009 Ed Bueler and Constantine Khroulev*

*This file is part of PISM.*

*PISM is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.*

*PISM is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.*

*You should have received a copy of the GNU General Public License along with PISM; see COPYING in PISM root directory. If not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA*

## 1. LIBRARIES AND PROGRAMS NEEDED BY PISM

The following table lists the *required* dependencies for PISM, arranged alphabetically.

Required Library	Comment
GSL ( <a href="http://www.gnu.org/software/gsl">www.gnu.org/software/gsl</a> )	
MPI ( <a href="http://www-unix.mcs.anl.gov/mpi">www-unix.mcs.anl.gov/mpi</a> )	
NetCDF ( <a href="http://www.unidata.ucar.edu/software/netcdf">www.unidata.ucar.edu/software/netcdf</a> )	version $\geq$ 3.6.0
PETSc ( <a href="http://www-unix.mcs.anl.gov/petsc">www-unix.mcs.anl.gov/petsc</a> )	version $\geq$ 2.3.3 or 3.0.0

One library is only used for the earth deformation (“isostatic rebound”) model

Recommended Library	Comment
FFTW ( <a href="http://www.fftw.org">www.fftw.org</a> )	if not present set PISM_HAVE_FFTW=0

In addition, Python (<http://python.org/>) is needed both in the PETSc installation process and in scripts related to PISM pre- and post-processing, while Subversion (<http://subversion.tigris.org/>) is usually needed to download the PISM code. Both should be included in any Linux/Unix distribution.

These Python packages are needed to do all the examples in the *User’s Manual* by running python scripts.

Recommended Python Package	Importance
matplotlib ( <a href="http://matplotlib.sourceforge.net">matplotlib.sourceforge.net</a> )	used in some scripts
netcdf4-python ( <a href="http://code.google.com/p/netcdf4-python">code.google.com/p/netcdf4-python</a> )	used in <i>most</i> scripts
numpy ( <a href="http://numpy.scipy.org">numpy.scipy.org</a> )	used in <i>most</i> scripts
scikits.delatunay ( <a href="http://scipy.org/scipy/scikits">scipy.org/scipy/scikits</a> )	used in only one script

## 2. INSTALLATION COOKBOOK

**2.1. Installing PISM on a Debian-based Linux system.** Of all operating systems it is probably quickest to install all the PISM prerequisites on a Debian-based Linux system using a package manager. PISM requires PETSc revision 2.3.3 or 3.0.0 or newer, so you should only use this method if an up-to-date PETSc package is available.

- (1) Find and install (using `apt-get` or `synaptic`, for example) the following packages:

<i>Package name</i>	<i>Comments</i>
<code>petsc-dev</code>	depends on <code>libpetsc2.3.3-dev</code> ; this package depends on <code>libpetsc2.3.3</code> , <code>libopenmpi-dev</code> , <code>libx11-dev</code> , <code>libblas-dev</code> , <code>liblapack-dev</code> , <code>gfortran</code> , and others; note <code>libopenmpi-dev</code> depends on <code>libopenmpi1</code> and replaces <code>openmpi-bin</code>
<code>libgs10-dev</code>	
<code>libfftw3-dev</code>	
<code>netcdf-bin</code>	(and perhaps grab <code>nco</code> at the same time)
<code>libnetcdf-dev</code>	
<code>g++</code>	without this you may see a compiler error: “ <code>gcc: error trying to exec 'cc1plus'</code> ”
<code>subversion</code>	
<code>python-dev</code>	(helps with scripts ... perhaps not essential)

- (2) Get the PISM stable0.2 branch:

```
$ svn co http://svn.gna.org/svn/pism/tags/stable0.2 pism0.2
```

- (3) Based on the locations of installed files, add the lines similar to

```
export PETSC_DIR=/usr/lib/petscdir/2.3.3/
export PETSC_ARCH=linux-gnu-c-opt
export PATH=/home/user/pism0.2/bin/:/home/user/pism0.2/util/:$PATH
```

to your `.bashrc` equivalent. (See “Installed files” reported for `libpetsc2.3.3-dev` using `synaptic`, for example.)

- (4) Next, build PISM:

```
$ cd pism0.2
$ make all
```

- (5) Try it:

```
$ mpiexec -n 2 pismv -y 1
```

If you see “Writing model state to file...” at the end then the run completed.

- (6) See the *Getting Started* section *User’s Manual* to continue.

Running PISM might produce a warning message like the following:

```
$ mpiexec -n 2 pismv -y 1
libibverbs: Fatal: couldn't read uverbs ABI version.
```

```
-----
[0,1,0]: OpenIB on host user-laptop was unable to find any HCAs.
Another transport will be used instead, although this may result in
lower performance.
```

---

```
. . .
PISMV (verification mode)
initializing Test A ...
```

```
. . .
According to this comment found online at http://lists.alioth.debian.org/pipermail/pkg-openmpi-commits/2007-August/000044.html,
```

```
    To get rid of the warning, you can either disable the use of the "openib" btl
    in /etc/openmpi/openmpi-mca-params.conf or pass "--mca btl ^openib" to
    mpirun or mpiexec. The configuration file contains a line to disable InfiniBand,
    you just have to comment it out.
```

*NCVIEW*. There seems to be no *ncview* Debian package yet. But this sequence worked with minimal pain. Grab latest source from

```
    http://meteora.ucsd.edu/~pierce/ncview\_home\_page.html
```

Basically just get the `.tar.gz` and follow install instructions in top-level "README". But error about missing X headers etc is resolved by two things, first to get Debian `xorg-dev` package, and second to tell the `ncview configure` script where to find X:

```
$ ./configure --x-libraries=/usr/lib
```

## 2.2. Generic PISM installation, using PETSc source code.

### 2.2.1. *Installing prerequisites.*

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 listed in section 1, *but* an up-to-date PETSc distribution may not be available in your system's package repositories. You will need Python and Subversion installed, but these are included in all current Linux distributions. To use the (recommended) graphical output of PISM you will need an X Windows server.
2. As PISM is currently under rapid development, we distribute it only as compilable source code. On the other hand, there are several software libraries needed by PISM. Therefore the "header files" for these libraries are required for building PISM. In particular this means that the "developer's versions" of the libraries are needed if the libraries are downloaded from package repositories like Debian.
3. PISM uses NetCDF (= *network Common Data Form*) for an input and output file format. If it is not already present, install it using the instructions at the webpage or using a package management system.
4. PISM uses the GSL (= *GNU Scientific Library*) for certain numerical calculations and special functions. If it is not already present, install it using the instructions at the webpage or using a package management system.
5. PISM optionally uses the "FFTW" library (FFTW = *Fastest Fourier Transform in the West*) in one approximation of the deformation of the solid earth (bed) under ice loads.<sup>1</sup> If you want the functionality of this bed deformation model, which is coupled to the ice flow and which we recommend, install FFTW or check that it is installed already. If FFTW is *not installed*, however, turn off PISM's attempt to build with it by setting the environment variable PISM\_HAVE\_FFTW=0. If this library is absent, all of PISM will work *except* for this bed deformation model.
6. You will need a version of MPI (= *Message Passing Interface*). 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 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. For example, you can add it 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 so that there is no need to retype it each time you use MPI.

*From now on this manual will assume the use of bash.*

---

<sup>1</sup>Bueler and others (2007). *Fast computation of a viscoelastic deformable Earth model for ice sheet simulation*, Ann. Glaciol. 46, pp. 97-105.

7. PISM uses PETSc (= *Portable Extensible Toolkit for Scientific Computation*). Let's get this out of the way: note "PETSc" is pronounced "pet-see". Download the PETSc source by grabbing the current gzipped tarball:

`www-unix.mcs.anl.gov/petsc/`

PISM requires a version of PETSc which is 2.3.3 or 3.0.0 or later. The "lite" form of PETSc is fine if you are willing to depend on an internet connection for accessing the PETSc documentation.

You should configure and build PETSc *essentially* as described on the PETSc installation page, but it might be best to read the following comments on the PETSc configure and build process first:

- (i) Untar in your preferred location, but note PETSc should *not* be configured (next) using root privileges. Note that you will need to define the environment variable `PETSC_DIR` before configuring PETSc (next). For instance, once you have entered the PETSc directory just untarred, `export PETSC_DIR=$PWD`. (Or `PETSC_DIR='pwd'`. In this case, note the use of the backprime (*accent-grave*) character, and not the single apostrophe '.)
- (ii) When you run the configure script in the PETSc directory, the following options are recommended; note PISM uses shared libraries by default:
 

```
$ export PETSC_ARCH=linux-gnu-opt
$ ./config/configure.py --with-shared --with-dynamic --with-debugging=no
```

 Turning off the inclusion of the debugging symbols gives a significant speed improvement. Using shared and dynamic libraries may be unwise on certain clusters, etc.; check with your system administrator. Note that there is no PISM use of Fortran, and that it is sometimes convenient to have PETSc grab a local copy of BLAS and LAPACK rather than using the system-wide version. So one may add "`--with-fortran=0 --download-c-blas-lapack=1`" to the other configure options.
- (iii) 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 `mpicxx` and `mpicc`, and either `mpiexec` or `mpirun`, in subdirectory `bin/` and MPI library files in subdirectory `lib/`.
- (iv) 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`.
- (v) Configuration of PETSc for a batch system requires special procedures described at the PETSc documentation site. One starts with a configure option `--with-batch=1`. See the "Installing on machine requiring cross compiler or a job scheduler" section of the PETSc installation page.
- (vi) 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.)

- (vii) 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; as noted you will need the X header files for this to work.
- (viii) Finally, you will want to set the `PETSC_DIR` and 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-cxx
export PATH=$PETSC_DIR/externalpackages/mpich2-1.0.4p1/$PETSC_ARCH/bin/:$PATH
```

 could appear in one of those files.

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

2.2.2. *Installing PISM itself.* At this point you have configured the environment which PISM needs. You are ready to build PISM itself, which is a much quicker procedure!

8. Get the latest source for PISM using the Subversion version control system:
  - (a) Check the website <http://www.pism-docs.org/> for the latest version of PISM.
  - (b) Do
 

```
$ svn co http://svn.gna.org/svn/pism/tags/stable0.2 pism0.2
```
  - (c) A directory called “`pism0.2/`” will be created. Note that in the future when you enter that directory, `svn update` will update to the latest revision of PISM.<sup>2</sup>
9. Build PISM:<sup>3</sup>

```
$ cd pism0.2
$ make
```

If your operating system does not support shared libraries, then do `export PISM_STATIC=1` before building. This might be necessary if you’re building on a Cray XT5 or a Sun Opteron Cluster, for example.

Note that the “`make`” puts executables, including `pismd`, `pismr`, `pismv`, and `pisms`, in the `pism0.2/bin/` subdirectory. Object files created during the build process (located in the `build` subdirectory) are not deleted, so do “`make clean`” to delete these if space is an issue.

10. PISM executables can be run most easily by adding the directories `pism0.2/bin/` and `pism0.2/util/` 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/pism0.2/bin:/home/user/pism0.2/util:$PATH
```

---

<sup>2</sup>Of course, after `svn update` you will `make all` to recompile and relink PISM. Alternatively, you can `make update`, which will run `svn update` and then rebuild PISM if necessary.

<sup>3</sup>Please report any problems you meet at this build stage by sending us the output: [help@pism-docs.org](mailto:help@pism-docs.org).

**2.3. Installing PISM on Mac OS X.** Please refer to section 2.2 for the description of PISM's use of prerequisites and generic installation instructions. This section describes an adaptation of the same process to the Mac OS X operating system.

1. As PISM is distributed as compilable source code only, you will need software developer's tools, XCode and the *X window system*, X11. Both packages can be installed by either downloading them from Apple Developer Connection or using the Mac OS X installation DVD.

2. The use of MacPorts is recommended, as it significantly simplifies installing many open-source libraries. Download the package from the MacPorts homepage, install and then add `/opt/local/bin` and `/opt/local/sbin` to your `PATH`:

```
export PATH=/opt/local/bin:/opt/local/sbin:$PATH
```

3. Note that it is not necessary to install Python, as it is bundled with the operating system (as are BLAS, LAPACK, OpenMPI and Subversion). The easiest way to get SciPy et al to work on an Intel-based Mac with Mac OS X is to use SciPy Superpack from <http://macinscience.org/>. This package includes `numpy`, `SciPy` and `matplotlib`, as well as `ipython`, a very useful interactive Python shell. Unfortunately, Scipy Superpack does not support PowerPC-based Macs; we recommend using Enthought Python Distributions on older Macs.

4. After installing MacPorts, do

```
$ sudo port install netcdf ncview gsl fftw-3
```

to install NetCDF, ncview, GSL and FFTW.

5. At this point, all the PISM prerequisites except PETSc are installed. Download the latest PETSc tarball from the PETSc website<sup>4</sup>. Untar, then change to the directory just created. The next three commands completed the PETSc installation:

```
$ export PETSC_DIR=$PWD; export PETSC_ARCH=macosx;
```

```
$ ./config/configure.py --with-shared --with-dynamic --with-debugging=no \  
    --with-fortran=0 --with-blas-lapack-dir=/usr/ --with-mpi-dir=/usr/
```

```
$ make all test
```

6. Finally, to build PISM, you should follow the steps described in 2.2.2, with one modification: after getting PISM sources, do

```
$ export CONFIG=config/macosx_macports
```

```
$ make all
```

The first command includes the file `config/macosx_macports` into `build/Makefile`, modifying compiler and linker flags. See section 2.4 for more info.

Note that it might also be necessary to add

```
export DYLD_LIBRARY_PATH=/home/user/pism0.2/lib/
```

to your `.bashrc` or equivalent to tell the dynamic library loader where to find PISM libraries.

---

<sup>4</sup>Version 3 is recommended; 2.3.3 will work, but building it on Mac OS X is tricky.

**2.4. If nothing else works: modifying makefiles.** On some platforms it is necessary to use compiler or linker flags different from ones used on Linux, the default. This can be done by editing the top-level makefile (`Makefile`) or `build/Makefile`, but here is a more robust approach.

Set the environment variable `CONFIG` to `config/foo` to include the file `config/foo` into the makefile. It is included *after* all the flags, both compiler and linker, are defined, so it is possible to override any of them. This is the way the PISM build process works on Mac OS X, using `CONFIG=config/macosx_macports`. There are two advantages of this approach: first, settings for different systems can be kept separate. Second, this allows building PISM on different systems using the same set of makefiles, so an `svn update` is less likely to break the build process on your system.

### 3. QUICK TESTS OF THE INSTALLATION

Once you're done with the installation, a few tests can confirm that PISM is functioning correctly.

1. Try a MPI four processor verification run:

```
mpiexec -n 4 pismv -test G -y 200
```

If you see some output and a final `Writing model state to file 'verify.nc' ... done` then PISM completed successfully. At the end of this run you get measurements of the difference between the numerical result and the exact solution.<sup>5</sup>

This should work even if there is only one actual processor (core) on your machine, in which case MPI will just run multiple processes on the one processor.

2. Try a verification run using the PETSc viewers (under the X window system):

```
pismv -test G -y 200 -d HTc
```

When using such viewers and `mpiexec` the additional final option `-display :0` is sometimes required to enable MPI to use X, like this:

```
mpiexec -n 2 pismv -test G -y 200 -d HTc -display :0
```

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

```
test/verifynow.py
```

or, on an  $N$  processor machine,

```
test/verifynow.py -n N
```

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 ([help@pism-docs.org](mailto:help@pism-docs.org)). See section *Verification* of the *User's Manual* for more on PISM verification.

*Next steps.* Start with the *User's Manual*, which has a “Getting started” section. A copy is online at the PISM homepage and documentation page [www.pism-docs.org](http://www.pism-docs.org), along with a *Reference Manual* and additional technical documentation.

Completely up-to-date *Manuals* can be built from L<sup>A</sup>T<sub>E</sub>X source in the `doc\` subdirectory, as described in the next section.

A final reminder with respect to installation: Let's assume you have checked out a copy of PISM using Subversion, as in step 8a above. You can update your copy of PISM to the latest version by `make update` in the PISM directory. This will run `svn update` and then `make all install` if there were updates from the source code repository.

---

<sup>5</sup>Bueler and others, 2007. *Exact solutions to the thermomechanically coupled shallow ice approximation: effective tools for verification*, J. Glaciol. 53 (182), pp. 499–516

## 4. INSTALLING PYTHON PACKAGES

If you're lucky, you might be able to install all the Python packages mentioned in section 1 using a package manager. On the other hand, the Python packages below are not currently available in Debian package repositories. They are easy to install using Python `setuptools`, however; these tools are included with recent versions of Python. And you might not be using a package manager!

*Python module netCDF3, from package netcdf4-python.* Typical users have NetCDF 3.6.2. You can install `netcdf4-python`, to get `netCDF3` needed by PISM scripts, by downloading a tarball from the project homepage <http://code.google.com/p/netcdf4-python/>. Go to that site to identify the latest one, and download with a web browser or do something like the following, using the appropriate version number:

```
$ wget http://netcdf4-python.googlecode.com/files/netCDF4-NUMBER.tar.gz
$ tar -xvf netCDF4-NUMBER.tar.gz
```

Enter the directory you just untarred and install:

```
$ cd netCDF4-NUMBER/
$ sudo NETCDF3_DIR=/usr/ python setup-nc3.py install
```

assuming that NetCDF was installed in the `/usr/` tree.

Under Mac OS X,

```
$ sudo NETCDF3_DIR=/opt/local/ python setup-nc3.py install
```

If you actually use NetCDF 4.0,

```
$ sudo NETCDF4_DIR=/usr/ HDF5_DIR=/usr/ python setup.py install
```

assuming that HDF5 and NetCDF were installed in the `/usr/` tree.

The scripts in directories `util/`, `examples/eisgreen`, `examples/eisross`, `examples/mismip`, and so on, which need `netCDF3`, should now work.

*Python package scikits.delauay.* One PISM script, `examples/eisross/ross_plot.py`, needs this. Here's how to get it:

```
$ svn co http://svn.scipy.org/svn/scikits/trunk/delaunay
$ cd delaunay
$ sudo python setup.py install
```

## 5. REBUILDING PISM DOCUMENTATION

You might want to rebuild the documentation from source, as PISM and its documentation evolve together. These tools are required:

---

L <sup>A</sup> T <sub>E</sub> X	<a href="http://www.latex-project.org">www.latex-project.org</a>	for rebuilding this <i>Installation Manual</i> , the <i>User's Manual</i> and the <i>Reference Manual</i> from source
doxygen	<a href="http://www.doxygen.org">www.doxygen.org</a>	for rebuilding the <i>Reference Manual</i> and the <i>C++ class Browser</i> from source

---

To rebuild PISM documentation, change to the PISM root directory and do

<code>make userman</code>	to build the <i>User's Manual</i> , <code>doc/manual.pdf</code>
<code>make refman</code>	to build the <i>Reference Manual</i> , <code>doc/refman.pdf</code>
<code>make installation</code>	to build this document, the <i>Installation Manual</i> , <code>doc/installation.pdf</code>
<code>make browser</code>	to build the <i>C++ class browser</i> , <code>doc/doxy/index.html</code>
<code>make summary</code>	to build a PISM executive summary, <code>doc/pism_summary.pdf</code>
<code>make fullbib</code>	to build a complete bibliography from <code>doc/ice_bib.bib</code> , <code>doc/fullbib.pdf</code>

The default make targets are `installation userman browser refman`.