Fenics Ice Sheet Model Readme

March 16, 2020

This document briefly outlines how to get started with the Fenics ice sheet model.

1 Installation

The ice sheet mdoel is built using the open source Python finite element software Fenics, and depends on the package tlm-adjoint for implementing inversion and error propagation capabilities. The simplest way to install Fenics and tlm-adjoint is to create a conda environment.

1.1 Installing Fenics

1. Install Anaconda. This can be either Anaconda itself, or miniconda, which is a stripped down version. Ensure the Python version is greater than 3.6. Installer can be found here: https://www.anaconda.com/distribution/

2. Create a new conda environment.
conda create -n fenics -c conda-forge fenics fenics-dijitso fenics-dolfin fenics-ffc fenics-fiat fenics-libdolfin fenics-ufl

3. Enter the conda environment:
source activate fenics

4. Make sure the pip package manager is up to date:
pip install --upgrade pip

5. Install the following packages:
conda install matplotlib numpy ipython scipy

6. Install hdf5 for python:
`http://docs.h5py.org/en/latest/index.html`
pip install h5py

7. Install pyrevolve:
https://github.com/opesci/pyrevolve

Change to directory where you would like to download pyrevolve to. You can delete the pyrevolve directory after finishing this step.

git clone https://github.com/opesci/pyrevolve.git
cd pyrevolve/
python setup.py install

8. Install mpi4py:
http://mpi4py.scipy.org/docs/
pip install mpi4py

9. To enter this environment:
source activate fenics

10. To exit:
source deactivate fenics

1.2 tlm_adjoint

1. Clone the git repository to the local drive where you want it to live:
git clone https://github.com/jrmaddison/tlm_adjoint.git

1.3 Fenics ICE

1. Clone the git repository to the local drive where you want it to live:
git clone https://github.com/cpk26/fenics_ice.git

1.4 Modifying the Python Path

Modify the default paths python looks for modules to include tlm_adjoint and fenics ice.
Add to the end of .bashrc:
PYTHONPATH=¨${PYTHONPATH}:/PATH/TO/tlm_adjoint/python:/PATH/TO/fenics_ice/code¨
export PYTHONPATH

2 Program structure

2.1 Directory Structure

```
fenics_ice
└─ code
   └─ model.py
```

```
|
|  |
|  └── solver.py
├── runs
|  ├── process_eigendec.py
|  ├── run_balancemeltrates.py
|  ├── run_eigendec.py
|  ├── run_errorprop.py
|  ├── run_forward.py
|  ├── run_inv.py
|  ├── run_invsigma.py
|  └── run_momsolve.py
├── scripts
|  └── ismipc
├── aux
|  ├── gen_ismipC_domain.py
|  └── test_domains.py
├── input
|  └── ismipc
└── output
   └── ismipc
```

2.2 Overview

The core of the ice sheet model is in two files: `/code/model.py` and `/code/solver.py`. These are utilized by the python scripts in the `/runs` folder, which execute specific parts of a simulation. The scripts there are generic to any simulation. Each specific simulation then has its own primary folder in the `/scripts` folder, which will call program files in tt /runs with specific parameters and data files.

The bash scripts in `/scripts` are where parameters and data file locations are specified. The data and parameters are used by the program files in `/runs` to create a model object (via a class defined in `model.py`) and subsequently a solver object (via a class defined in `solver.py`). The model object contains all the necessary data for a simulation, such as topography, constants, and velocity observations for inversions. The solver object contains the ice sheet physics/inversion code. The model object is passed as a parameter to your solver object. This object then allows you to solve the SSA equations on your domain, invert for basal drag or B_{glen}, and perform uncertainty quantification.

The `/aux` folder contains auxillary files; in here, the file `gen_ismipC_domain.py` generates the ismipC domain, based off definitions in `test_domains.py`. The `/input` folder is where input files, such as topography and ice thickness, for specific simulations are located. Similarily, the `/output` folder is where output is stored from specific simulations.

3 Tutorial: A walkthrough IsmipC