

A Quick Introduction to TMM

Tatsuro Tanioka

September 11, 2020

Abstract

This document goes through how to use the Transport Matrix Model (TMM) using computational resources at the University of Minnesota's Minnesota Supercomputing Institution (MSI). All the source codes and documents (including this tutorial) that Tanioka made to TMM is available at my github website (https://github.com/tanio003/tmm/tree/TT_Release).

Contents

1	Setting up	1
1.1	Flow Chart	1
1.2	Steps	1
1.2.1	Step 0: Logging into MSI and Mesabi	1
1.2.2	Step 1: Installing and configuring PETSc	2
1.2.3	Step 2: Downloading all the scripts and transport matrices	3

1 Setting up

1.1 Flow Chart

Before you do anything, read "README.txt" by Samar Khatiwala at the following website: <https://github.com/samarkhatiwala/tmm>. I will go over each of these steps specifically aimed at audiences using the computational cluster *Mesabi*.

1. Installing and configuring PETSc
2. Downloading all the scripts and transport matrices into your own local directory
3. Compiling the model (we use the BGC model MOPS2 for this example)
4. Running the model
5. Processing the model outputs
6. Displaying the model outputs

1.2 Steps

1.2.1 Step 0: Logging into MSI and Mesabi

Open the terminal (assuming that you have a MAC or Linux environment) on your computer and log in to MSI with your x500 account:

```
$ ssh -Yt youremail@umn.edu
```

Log in to Mesabi:

```
$ ssh -X mesabi
```

Make a new directory called TMM2 in your home directory and enter into this directory. Everything related to TMM will go into this directory.

```
$ mkdir TMM2
$ cd TMM2
```

1.2.2 Step 1: Installing and configuring PETSc

Download the latest version of PETSc and save it in your TMM2 directory and unzip this package. If opened properly, you should see the new directory petsc-3.13.5.

```
$ wget http://ftp.mcs.anl.gov/pub/petsc/release-snapshots/petsc-lite-3.13.5.tar.gz
$ tar -xvf petsc-lite-3.13.5.tar.gz
$ ls
petsc-3.13.5
```

Import the required modules: (1) impi, (2) impi/intel, and (3) cmake. Also make sure that you are using python 3, not python 2 (= default for MSI).

```
$ module purge
$ module load intel
$ module load impi/intel
$ module load cmake
$ module load python3
$ module list
Currently Loaded Modulefiles:
 1) intel/2018.release(default)  4) cmake/3.10.2(default)
 2) intel/2018/release          5) python3/3.7.1_anaconda
 3) impi/intel(default)
```

Set up \$PETSC_DIR to your petsc-3.13.5 directory:

```
$ export PETSC_DIR=$HOME/TMM2/petsc-3.13.5
$ echo $PETSC_DIR
/.../TMM2/petsc-3.13.5
```

Configure PETSc. Although this part is quite tricky you can copy and use my config file “reconfigure-arch-linux-c-opt.py”. If the config file does not work properly, let me know and I can show you a way to compile without using this .py file.

```
$ cd petsc-3.13.5
$ cp ~/../tanio003/TMM2/petsc-3.13.5/config/reconfigure-arch-linux-c-opt.py config/
$ config/reconfigure-arch-linux-c-opt.py
```

Don’t worry about some warning signs. It takes few minutes to compile. If it’s compiled properly you should see the notice “Configure stage complete.” Then build PETSc library:

```
$ make all
```

Building process takes about 15-30 minutes. If you’re very lucky it will go through in a single shot. But in most cases, it fails during the middle of the process. Don’t worry if it fails the first time. Simply type “\$ make all” again and hopefully it will finish building from where it left off. If built properly, you should see the message “Now to check if the libraries are working do:...”. Then type,

```
$ make check
...
Completed test examples
```

If you get this far, you’ve managed to build the PETSc successfully and you’re ready to go to the next step. If it failed, read the error messages, debug, and try again. **Building PETSc is harder than it looks** so you need to be patient.

As more of a technical note, the procedure above uses the Intel compilers and Intel MPI library. By loading the cmake, the PETSc build system can learn more about the host machine.

In addition to taking advantage of compiler optimizations and vectorization, the procedure above builds PETSc against the Intel Math Kernel Library (MKL) for BLAS, LAPACK and ScaLAPACK which gives a performance gain over the reference implementations. For the FORTRAN compiler, we specifically need to use `mpiifort`, and not `mpif90` (the default compiler), because TMM codes are written in both F77 and F90. Also, since we don't require C++ for TMM we put the flag in the config file, `--with-cxx=0`. The reason we need to use MPI compilers, not regular gcc compilers, is because we want to run PETSc in a parallel mode (i.e., by using the command `mpiexec` in the runscript). For more details about building PETSc please check out <https://www.mcs.anl.gov/petsc/documentation/installation.html>.

1.2.3 Step 2: Downloading all the scripts and transport matrices

1. First download Matlab scripts from http://kelvin.earth.ox.ac.uk/spk/Research/TMM/tmm_matlab_code.tar.gz and put into the first level of your TMM2 folder Path. You could also get my copy.

```
$ cp -r ~/../tanio003/TMM2/tmm_matlab_code $HOME/TMM2/
```

2. Download transport matrices and related data for the model of your choice: <http://kelvin.earth.ox.ac.uk/spk/Research/TMM/TransportMatrixConfigs/> and put into the first level of your TMM2 folder Path. You can download all 6 configurations but I warn you that MITgcm_ECCO_v4 and UVicKielIncrIsopycDiffTransient take a very long time. You could also grab my copy (e.g., to copy MITgcm_ECCO).

```
$ cp -r ~/../tanio003/TMM2/MITgcm_ECCO $HOME/TMM2/
```

3. Download miscellaneous data called OceanCarbon from <http://kelvin.earth.ox.ac.uk/spk/Research/TMM/MiscData/>. You can get my copy by:

```
$ cp -r ~/../tanio003/TMM2/OceanCarbon $HOME/TMM2/
```

4. Download source codes for TMM and models:

```
$ git clone https://github.com/tanio003/tmm
```

This directory (`/TMM2/tmm`) contains the source codes from Khatiwala's master branch ("master") and my public release branch ("TT_Release"). For our exercise, we will be using some of my new codes so switch from the master branch to my branch in the newly created tmm directory:

```
$ cd tmm
(master) $ ls
driver HOWTO.txt LICENSE.txt models README.txt
(master) $ git checkout TT_Release
(TT_Release) $ ls
driver HOWTO.txt LICENSE.txt models README.txt Tutorial_MSI
```

Notice that in the branch `TT_Release`, there is a new directory `Tutorial_MSI`, which was not present in the master branch.

(Optional) If you want to make start changing your own changes, I would suggest making a new branch in your local computer (e.g., `yournewrepo`), and leave `master` and `TT_Release` untouched.

```
(TT_Release) $ git checkout -b yournewrepo
(yournewrepo) $ git branch --show-current
yournewrepo
```

5. Set the environment variable `TMMROOT` to point to the top level of the TMM directory.

```
(TT_Release) $ export TMMROOT=$HOME/TMM2/tmm
(TT_Release) $ echo $TMMROOT
/home/.../TMM2/tmm
```