A Quick Introduction to TMM

Tatsuro Tanioka

September 14, 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 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 "Conifgure 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 he 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. For the ones that I have 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 you need to 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
```

1.2.4 Step 3: Compiling the model

Here, let's try compiling biogeochemical MOPS.

1. For each model there are model-specific source codes (`$TMMROOT/models/current/mops2/src/`); Matlab scripts (`$TMMROOT/models/current/mops2/matlab/`) to generate input data and read model output; and run scripts and other runtime data such as namelists in `$TMMROOT/models/current/mops2/runscripts/`.

   First, we create a new "Run directory". I make a new base directory `Runs` and in that directory, I make subdirectories for specific experiments. I call it `Runs/MOPS/Test_spinup` and copy here all the files needed.

```
$ cd ~/TMM2
$ mkdir -p Runs/MOPS/TMM_spinup
$ cd Runs/MOPS/TMM_spinup
$ cp -p $TMMROOT/models/current/mops2.0/src/Makefile .
$ cp -p -R $TMMROOT/models/current/mops2.0/matlab/* .
$ cp -p $TMMROOT/models/current/mops2.0/runscripts/* .
```

2. Compile mops. Make sure that all the modules are loaded before you compile mops.

```
$ module load intel
$ module load impi/intel
$ module load cmake
$ make clean all
$ make mops
```

   If compiled properly, you'd find a new executable "mops" created along with a bunch of objective .o files.

```
$ ls
BGC_INI.o                          n7fluxes28.m
BGC_MODEL.o                        n7physics.m
CAR_CHEM.o                         n7tracers28.m
CAR_INI.o                          n7tracersavg28.m
external_forcing_mops_biogeochem.o perry1996-runoff-noarctic_noname.txt
insolation.o                       perry1996-runoff_noname.txt
load_output.m                      petsc_matvec_utils.o
load_output_time_avg.m             petsc_signal_utils.o
load_pco2.m                        process_output.m
Makefile                           runscript
make_input_files_for_mops_model.m  runscript_msi
make_rivers.m                      tmm_external_bc.o
misfit_mops_biogeochem.o           tmm_forcing_utils.o
mops                               tmm_forward_step.o
mops_biogeochem_copy_data.o        tmm_main.o
mops_biogeochem_diagnostics.o      tmm_monitor.o
mops_biogeochem_ini.o              tmm_profile_utils.o
mops_biogeochem_misfit.o           tmm_timer.o
mops_biogeochem_model.o            tmm_write.o
mops_biogeochem_set_params.o
```

3. Edit the file `make_input_files_for_mops_model.m`. First thing to do is to make sure that variable `base_path` point to the right directory for the TMM configuration.

```
% make_input_files_for_mops_model.m

% Set toplevel path to GCMs configuration
% base_path='/data2/spk/TransportMatrixConfigs/MITgcm_2.8deg';
% base_path='/data2/spk/TransportMatrixConfigs/MITgcm_ECCO';
% base_path='/data2/spk/TransportMatrixConfigs/MITgcm_ECCO_v4';
base_path='~/TMM2/MITgcm_2.8deg';

addpath(genpath('~/TMM2/tmm_matlab_code'));% add tmm_matlab_code to the search path
oceanCarbonBasePath='~/TMM2/OceanCarbon';  % add OceanCarbon to the search path
atmosDataPath=fullfile(oceanCarbonBasePath,'AtmosphericCarbonData');
```

In the same matlab file, there are different switches with 0's and 1's. For this spin-up exercise, we **couple MOPS to a simple OCMIP-like carbon model** and **fix atmospheric** pCO_2 **at 280 ppm**. So set the switches as following:

```
% make_input_files_for_mops_model.m
...
periodicForcing=1
periodicMatrix=1

dt=43200; % time step to use (43200s for ECCO and MIT2.8; 28800s for any other TMMs)

rearrangeProfiles=1
bigMat=0
writeFiles=1
writeTMs=1
useCoarseGrainedMatrix=0
writePCFiles=0

READ_SWRAD=0                    % Read short-wave radiation?
useCarbon=1                     % Use simple inorganic carbon model?
useAtmModel=0                   % Use prognostic 1-box atmosphere?
pCO2atm_ini=280.0               % Initial pco2?
useTimeVaryingPrescribedCO2=0   % Use prescribed pco2 pathway?
useVirtualFlux=1                % Use DIC and Alk to calculate E-P?
empScaleFactor=1.0              % Scaling factor for E-P (default = 1)
%-----------------------------------
% Modified by Tatsuro Tanioka 200907 to allow for Atmospheric CO2 option
% For a prescribed pCO2 run, useTimeVaryingPrescribedCO2=1 and choose a scenario

% Available options: 'historical', 'RCP3PD', 'RCP45', 'RCP6' and 'RCP85'
co2Scenario='RCP85';
%-----------------------------------
```

Then open MATLAB and run `make_input_files_for_mops_model.m`

```
$ module load matlab
$ matlab -nodesktop
                                  < M A T L A B (R) >
                        Copyright 1984-2019 The MathWorks, Inc.
                    R2019a Update 5 (9.6.0.1174912) 64-bit (glnxa64)
                                    July 31, 2019
To get started, type doc.
For product information, visit www.mathworks.com.

>> make_input_files_for_mops_model
```

This creates a bunch of periodic forcing files (`xxx_01, xxx_02,...`), initial tracer concentrations (`po4ini.petsc, no3ini.petsc,...`), and binary files (.bin and .petsc) related to model geometry and forcing.

1.2.5 Step 4: Running the model

MSI systems use job queues to efficiently and fairly manage when computations are executed. The queuing system at MSI is called PBS (Portable Batch System) and to submit a job to a PBS queue users create PBS job scripts. PBS script contains information on the resources requested for calculation, as well as the commands for executing the calculation.

Below is the custom PBS script for submitting a new job to run MOPS2 using Mesabi. It's called `runscript_msi` and should be in the current directory already. Here is the first 11 lines:

```
1 #!/bin/bash -l
2 #PBS -l walltime=06:00:00,nodes=1:ppn=24,pmem=2580mb
3 #PBS -m abe
4 #PBS -j oe
5 #PBS -M tatsurobkkuk@gmail.com
6
7 cd $PBS_O_WORKDIR
8
9 module load intel
10 module load impi/intel
11 module load cmake
```

The first line defines which type of shell the script will be read. Here we will use the `bash`. The second line contains the PBS resource request. The current job will require about 6 hours, 1 node each with 24 processor cores (ppn), and 2580 megabytes of memory per core (pmem).

The two lines containing `#PBS -m abe`, and `#PBS -M tatsurobkkuk@gmail.com`. are both commands having to do with sending message emails to the user. The first of these lines instructs the PBS system to send a message email when the job aborts, begins, or ends. The second command specifies the email address to be used. Using the message emails is recommended because the reason for a job failure can often be determined using information in the emails. The seventh line sets the directory at which commands are executed and the lines 8-11 loads the necessary software modules.

The follwing lines (14~) contain the commands to execute and start a specific program. Different flags need to be change accordingly depending on the nature of experiments. For more information on different options available, read "HOWTO.txt" by S. Khatiwala in `$TMMROOT`.

```
# 360 days per year with a time step of 2 steps per day:
14 mpiexec -np 24 -hostfile $PBS_NODEFILE ./mops \   # number of cores, models
15   -numtracers 9 \   # number of tracers (i.e. state variables)
16   -i po4ini.petsc,dopini.petsc,oxyini.petsc,phyini.petsc,zooini.petsc,detini.petsc,
no3ini.petsc,dicini.petsc,alkini.petsc \ # files for initialization of BGC state variables
17   -me Ae \   # the name of the explicit transport matrix
18   -mi Ai \   # the name of the implicit transport matrix
19   -t0 0.0 -iter0 0 \ # starting time[years] starting time[timesteps]: for initial run
20   -deltat_clock 0.0013888888888889 \  # ocean timestep length[years]: 2 timesteps/day
21   -max_steps 2160000 \   # total number of timesteps to be evaluated (here 3000 yrs)
22   -write_time_steps 72000 \ # output frequency(in timesteps, here every 100 yrs)
23   -o po4out.petsc,dopout.petsc,oxyout.petsc,phyout.petsc,zooout.petsc,detout.petsc,
no3out.petsc,dicout.petsc,alkout.petsc \ # files for output of 9 BGC state variables
24   -external_forcing \ # calculate BGC explicitly
25   -use_profiles \
26   -nzeuph 2 \   # number of layers in euphotic zone (2 for MIT2.8, 6 for ECCO)
27   -biogeochem_deltat 43200.0 -days_per_year 360.0 \ # ocean timestep[seconds]
28   -burial_sum_steps 720 \   # sum burial over a period of 720 timesteps = 1 yr
29   -pco2atm 280.0 \   # use fixed pCO2 of 280 ppm
30   -use_virtual_flux \ # use the global surface mean DIC and Alk to calculate E-P
31   -periodic_matrix \  # use of periodic transport matrix
32   -matrix_cycle_period 1.0 -matrix_num_per_period 12 \ # the unit of time is year and
# circulation has a periodicity of 1 year; monthly mean tranport matrix (12 TMs/year)
33   -periodic_biogeochem_forcing \  # periodic biogeochemical forcing
34   -periodic_biogeochem_cycle_period 1.0 -periodic_biogeochem_num_per_period 12 \
35   -num_biogeochem_steps_per_ocean_step 8 \ # the number of BGC timestep per ocean step
36   -separate_biogeochem_time_stepping \  # timestep BGC model separately from ocean step
37   -time_avg -avg_start_time_step 2159281 -avg_time_steps 60 \
# initial timestep for avg concentrations (for final year); avg over 60 timestep = 1 month
38   -avg_files po4avg.petsc,dopavg.petsc,oxyavg.petsc,phyavg.petsc,zooavg.petsc,
detavg.petsc,no3avg.petsc,dicavg.petsc,alkavg.petsc \   # avg file names
39   -calc_diagnostics -diag_start_time_step 2159281 -diag_time_steps 60 \
# initial timestep for diagnostic fluxes (for final year); avg over 60 timesteps = 1 month
40   -diag_files fbgc1.petsc,fbgc2.petsc,fbgc3.petsc,fbgc4.petsc,fbgc5.petsc,fbgc6.petsc,
fbgc7.petsc \ # diagnsotic flux files names
41    > log   # outputting to logfile
```

Currently BGC model writes the following 7 diagnostics into the different files:

1. `fbgc1.petsc`: primary production in each box [mmol P/m3/oceantimestep]

2. `fbgc2.petsc`: zooplankton grazing in each box [mmol P/m3/oceantimestep]
3. `fbgc3.petsc`: detritus sedimentation through upper boundary of each box [mmol P/m2/oceantimestep]
4. `fbgc4.petsc`: remineralization of detrius and DOP in each box [mmol P/m3/oceantimestep]
5. `fbgc5.petsc`: river runoff [mmol P/m3/oceantimestep]
6. `fbgc6.petsc`: nitrogen fixation [mmol N/m3/oceantimestep]
7. `fbgc7.petsc`: denitrification [mmol P/m3/oceantimestep]

1.2.6 Step 5: Processing the model outputs

To check that the run has finished, you should've received the email from MSI (if you set it so in the runscript) and you should see at the bottom of the log file the "Wall clock time". As the output files are binary and cannot be opened on its own, we have to convert from the binary format into either netcdf (.nc) or Matlab (.mat) files using Matlab scripts.

(Option 1, recommmended): To convert to **.nc** files you are going to use 5 scripts:

1. `n7tracers28.m`: This file converts each tracer snapshot .petsc file (e.g., po4put.petsc) into a single .nc file. The MATLAB syntax is:

   ```
   >> n7tracers28('filename1.nc')
   ```

   Make sure to set the basepath correctly and useCarbon=1 (when carbon model is used) and make sure that there is no .nc file with the same name already in the directory.

2. `n7tracersavg28.m`: This file converts each tracer time-averaged .petsc file (e.g., po4avg.petsc) into a single .nc file. The MATLAB syntax is:

   ```
   >> n7tracersavg28('filename2.nc')
   ```

   Make sure to set the basepath correctly and useCarbon=1 (when carbon model is used) and make sure that there is no .nc file with the same name already in the directory.

3. `n7fluxes28.m`: This file converts each diagnostic flux .petsc file (e.g., fbgc1.petsc) into a single .nc file. The MATLAB syntax is:

   ```
   >> n7fluxes28('filename3.nc')
   ```

   Make sure to set the basepath correctly and make sure that there is no .nc file with the same name already in the directory.

4. `n7physics.m`: This file makes single .c files with monthly mean temperature and salinity. The MATLAB syntax is:

   ```
   >> n7physics('filename4.nc')
   ```

   Make sure to set the basepath correctly and make sure that there is no .nc file with the same name already in the directory.

5. `load_pco2.m`: This file makes a global mean atmospheric pCO_2 file ("pco2.nc") and surface CO_2 air-sea flux file ("co2airseaflux.nc"). CO_2 is in ppm and CO_2 air-sea flux is in [mmol C/m2/timestep] (positive flux means CO_2 is going into the sea from air). The MATLAB syntax is

   ```
   >> load_pco2
   ```

Make sure to set the basepath and the CO_2 run options correctly.

6. `process_output.m` (optinal): To run all 5 scripts at once, edit and use this file. THe MATLAB syntax is

```
>> process_output
```

(Option 2): To convert to **.mat** files you are going to run 2 scripts:

1. `load_output.m`: This file converts each tracer snapshot .petsc file (e.g., po4put.petsc) to .mat file. Make sure to edit the `base_path` and add `tmm_matlab_code` to search path in lines 2 and 3.

2. `load_output_time_avg.m`: This file converts each tracer average concentration .petsc file (e.g., po4avg.petsc) to .mat file. Make sure to edit the `base_path` and search path correctly.

3. Unfortunately I don't have scripts for diagnostic flux .petsc files, physics files, and CO_2 files. But this should not be too hard to do and all you have to do is to modify other .m files.

1.2.7 Step 6: Displaying the model outputs

(Option 1): To view .nc files, I recommend the software Ferret developed by NOAA. It is already installed in MSI and to launch Ferret, type on the command line:

```
$ ls *.nc
filename1.nc filename2.nc filename3.nc filename4.nc co2airseaflux.nc pco2.nc
$ module load ferret
$ ferret
       NOAA/PMEL TMAP
       FERRET v6.82
       Linux 2.6.32-279.1.1.el6.x86_64 64-bit - 08/03/12
       14-Sep-20 14:49

yes? load filename1.nc
yes? sho d
     currently SET data sets:
   1> ./filename1.nc  (default)
 name     title                                I          J          K          L
 PO4                                           1:360      1:160      1:15       1:6
 DOP                                           1:360      1:160      1:15       1:6
 OXYGEN                                        1:360      1:160      1:15       1:6
 PHYTO                                         1:360      1:160      1:15       1:6
 ZOO                                           1:360      1:160      1:15       1:6
 DET                                           1:360      1:160      1:15       1:6
 NO3                                           1:360      1:160      1:15       1:6
 DIC                                           1:360      1:160      1:15       1:6
 ALK                                           1:360      1:160      1:15       1:6
```

I won't go into too much detail here but to learn more about Ferret, visit NOAA's website and take the tutorial at https://ferret.pmel.noaa.gov/Ferret/documentation/ferret-tutorials. You can make all kinds of graphs and figures.

(Option 2): If you want to visualize .mat files I recommend the MATLAB package `m_map`. Again I would not go into much detail here but if you want to learn about it, download this package and take the tutorial from the developer's website: https://www.eoas.ubc.ca/~rich/map.html. Compared to Ferret, you have more freedom for customizing graphs but the downside is that you are going to have to write much longer codes.