PETSC/TAO Introduction

Using the Extreme cluster:

ssh netID@login-2.extreme.uic.edu

To use on the cluster, just run

```
module purge
module load tools/petsc-3.7.3
```

You may also add these two lines to ~/.bash_profile (or ~/.profile or ~/.bashrc)[1], so that they are run every time you log in.

Then try a simple case under your home directory

```
mkdir test
cp /export/home/zhangx/petsc/tutorials/* test/*
make minsurf2                    // or try other cases like eptorsion2
mpirun -np 2 ./minsurf2          // two processes
```

See also the demo instructions in the first project package.
Read the cluster's FAQ page at: http://rc.uic.edu/resources/faq/
Also read the introduction slides of the Extreme cluster on Blackboard.

Learning TAO and PETSc

1. Read the highlighted parts in PETSc_Annoated.pdf on Blackboard.
2. Read the highlighted parts in TAO_Annotated.pdf on Blackboard.
3. Peruse demo_PETSc.cpp in the first project folder.
4. Function helps can be found online by typing the function name, e.g. MatCreate at http://www.mcs.anl.gov/petsc/petsc-current/docs/manualpages/Mat/MatCreate.html#MatCreate

Installation

To install PETSc on your own machine, follow the guidelines below. Since June 2015, TAO has been subsumed into PETSc, and there is no more need of installing TAO separately.

Linux users:

Below is how I installed PETSc on my Linux machine. Everything now works well.

[1] You may try and pick one of the three. Below I will only use ~/.bash_profile as an example.

1. Make sure you have already got g++ and gfortran installed.

2. Download the source code of PETSc by

```
git clone -b maint https://bitbucket.org/petsc/petsc petsc
```

or just download from https://www.mcs.anl.gov/petsc/download/index.html, and unzip.

3. Set the PETSC_DIR environment variable:

```
cd petsc  // the directory where you put the petsc stuffs; make sure the file “configure” is in it.
export PETSC_DIR=$PWD
```

4. Run configuration by (the following should be typed in a **single** line)

```
./configure --with-cc=gcc --with-cxx=g++ --with-fc=gfortran --with-shared-libraries
--download-fblaslapack --download-mpich
```

After completion, there is a summary shown. Take note of the values of PETSC_DIR and PETSC_ARCH. Call them xxx (which should be identical to the one set in Step 3) and yyy (e.g. arch-linux2-c-debug) respectively. Then in your ~/.bash_profile, add

```
export PETSC_DIR=xxx
export PETSC_ARCH=yyy
```

So every time you log in, these paths will have been set up properly.

5. Compile by

```
make all test
```

Pay attention to the last few lines after completion. If they report that test cases succeeded, then done.

6. To confirm that the installation is successful, do

```
cd $PETSC_DIR/src/tao/unconstrained/examples/tutorials
make minsurf2
mpiexec -np 2 ./minsurf2           // two processes
```

If the system can’t find mpiexec, just provide

```
export PATH=$PATH:$PETSC_DIR/$PETSC_ARCH/bin
```

and add this line to ~/.bash_profile.

Mac Users: roughly the same as Linux, but with the following caveats.

1. To install MPI, you may use

```
brew install mpich
```

2. Step 4 - the configure command could fail to work on mac. In this case, just run

```
sudo python setup.py install
```

The python file has the configure command with all the parameters set up for mac

3. The environment variable PETSC_ARCH may need to be set to empty, because the lib directory is located directly inside the home folder

```
export PETSC_ARCH=
```

I have not tried on Mac so I am just copying some ideas given by a student (thanks!).

Windows Users:

I used Cygwin to install PETSc. However I couldn't make it with MPI. So I settled for running on a single process, which seems great enough for finding and fixing most bugs. After it works on Cygwin (with a single process), I start debugging on the cluster with multiple processes. This usually saves me a lot of time, compared with debugging directly on the cluster from scratch.

1. Make sure you have g++ and gfortran included when installing Cygwin.

2. Download the source code of PETSc by

```
git clone -b maint https://bitbucket.org/petsc/petsc petsc
```

or just download from https://www.mcs.anl.gov/petsc/download/index.html, and unzip.

3. Set the PETSC_DIR environment variable:

```
cd petsc  // the directory where you put the petsc stuffs; make sure the file "configure" is in it.
export PETSC_DIR=$PWD
```

4. Run configuration by (the following should be typed in a **single** line)

```
./configure --with-cc=gcc --with-cxx=g++ --with-fc=gfortran --with-shared-libraries
--download-fblaslapack --with-mpi=0
```

After completion, there is a summary shown. Take note of the values of PETSC_DIR and PETSC_ARCH. Call them xxx (which should be identical to the one set in Step 3) and yyy (e.g. cygwin-cxx-debug) respectively. Then in your ~/.bash_profile, add

```
export PETSC_DIR=xxx
export PETSC_ARCH=yyy
```

So every time you start up Cygwin, these paths will have been set up properly.

5. Compile by

```
make all test
```

Pay attention to the last few lines after completion. If they report that test cases succeeded, then done. However in general it reports errors because we need the step 6 below to get things work. So don't worry.

6. On Cygwin (no need to do this step on Linux/Mac), you need to set the path of dynamic libraries such as libpetsc.so. Different from Linux which uses LD_LIBRARY_PATH, Cygwin uses PATH (oh my God). So run

```
export PATH=$PATH:$PETSC_DIR/$PETSC_ARCH/lib
```

Add this line to ~/.bash_profile below the two lines in Step 4.

7. To confirm that the installation is successful, do

```
cd $PETSC_DIR/src/tao/unconstrained/examples/tutorials

make minsurf1

./minsurf1          // single process
```

I also use WinSCP to remotely access files: https://winscp.net/eng/index.php. If you want further convenience by mapping the server as a new drive on Windows, try this free software: https://www.eldos.com/sftp-net-drive/download-release.php#product

Reading (to be completed):

On printing:

http://www.mcs.anl.gov/petsc/petsc-3.4/src/sys/examples/tutorials/ex2.c.html

On creating vectors:

On creating matrices:

On

MatCreateMAIJ(S, nclasses, *big_matrix);

#row(big_matrix) = #row(S) * nclass,

#column(big_matrix) = #column(S) * nclass

and local sizes also multiplied by nclass

demo_PETSc.cpp (300 lines) contains examples of manipulating matrices. Detailed comments are given. Read and learn. Compile by "make demo" and run by "mpirun -n 2 ./demo_PETSc".

Use PetscSplitOwnership with caution. It provides a local (or global) length given a global (or local) length. The result is determined via a simple formula. If a vector x is created by

VecCreateMPI(PETSC_COMM_WORLD, PETSC_DECIDE, 100, &x)

then by calling PetscSplitOwnership(PETSC_COMM_WORLD, &n, 100), n is assigned the right number of local entries of x on my process. However, if x is created by specifying the number of local elements:

VecCreateMPI(PETSC_COMM_WORLD, 40, 100, &x)   // for rank=0

VecCreateMPI(PETSC_COMM_WORLD, 60, 100, &x)   // for rank=1

then PetscSplitOwnership will NOT return the right local size. You have to use VecGetLocalSize to find it. In general, I recommend using VecGetLocalSize as much as possible.

Also can do the similar job is:

MatGetOwnershipRange(M, &first_row, &last_row);

MatGetOwnershipRangeColumn(*M, &first_column, &last_column);

But they can be called only after MatXXXSetPreallocation is already called.