Lab 5: PETSc and Krylov Methods

1 PETSc Installation

For the PETSc installation, please refer to the *install.sh* script in the appendix. Unless noted, the installation, and *petsc* folder is in PETSC_ROOT=*gpfs/accounts/ners570f20_class_root/ners570f30_class/$USER*
Logging is used for each step, as per the install script.

1. To download PETSc with git:

   git clone -b release https://gitlab.com/petsc/petsc.git

2. To load modules:

   module load gcc openmpi openblas lapack python3.6-anaconda

3. To configure PETSc with release mode, in the class user directory, with openblas, and optimization and debugging compiler options:

   –with-debugging=0 –prefix=$PETSC_ROOT –with-openblas=1

   –with-openblas-dir=$OPENBLAS_ROOT

   -COPTFLAGS=-g -O3 -CXXOPTFLAGS=-g -O3 -FOPTFLAGS=-g -O3

4. To configure PETSc to have the examples and tutorials, the base configuration appears to be adequate.

5. To run the tests, in the *petsc* directory,

   make test

   The test results were as follows, and only a small number of tests failed, primarily seeming to do with a *window_sync-flavor* module that is likely not installed for this configuration:

```
  Summary
# -------------
# FAILED diff-vec_is_sf_tutorials-ex1_7+sf_window_sync-fence_sf_window_flavor-
dynamic
vec_is_sf_tutorials-ex1_7+sf_window_sync-active_sf_window_flavor-dynamic
vec_is_sf_tutorials-ex1_7+sf_window_sync-lock_sf_window_flavor-dynamic
diff-vec_is_sf_tutorials-ex1_2+sf_window_sync-fence_sf_window_flavor-create
diff-vec_is_sf_tutorials-ex1_2+sf_window_sync-fence_sf_window_flavor-dynamic
diff-vec_is_sf_tutorials-ex1_2+sf_window_sync-active_sf_window_flavor-create
vec_is_sf_tutorials-ex1_2+sf_window_sync-active_sf_window_flavor-dynamic
diff-vec_is_sf_tutorials-ex1_2+sf_window_sync-lock_sf_window_flavor-create
```

```
diff-vec_is_sf_tutorials-ex1_2+sf_window_sync-lock_sf_window_flavor-dynamic
diff-vec_is_sf_tests-ex4_2_window+sf_window_sync-fence_sf_window_flavor-dynamic
vec_is_sf_tests-ex4_2_window+sf_window_sync-active_sf_window_flavor-dynamic
vec_is_sf_tests-ex4_2_window+sf_window_sync-lock_sf_window_flavor-dynamic
diff-vec_is_sf_tutorials-ex1_3+sf_window_sync-fence_sf_window_flavor-dynamic
...
# success 7271/9161 tests (79.4%)
# failed 54/9161 tests (0.6%)
# todo 225/9161 tests (2.5%)
# skip 1611/9161 tests (17.6%)
#
# Wall clock time for tests: 2797 sec
# Approximate CPU time (not incl. build time): 9637.589999999878 sec
#
# To rerun failed tests:
#     /usr/bin/gmake -f gmakefile test test-fail=1
#
# Timing summary (actual test time / total CPU time):
#   ksp_ksp_tutorials-ex56_2: 926.77 sec / 1267.83 sec
#   ksp_ksp_tutorials-ex70_fetidp_lumped: 83.10 sec / 345.95 sec
#   ksp_ksp_tutorials-ex43_3: 70.63 sec / 91.58 sec
#   ksp_ksp_tutorials-ex70_fetidp_saddlepoint_lumped: 68.00 sec / 276.56 sec
#   mat_tests-ex33_2: 66.24 sec / 89.44 sec
```

6. To install the PETSc build, it is ensured that the environmental variables are first set, PETSC_DIR=$PETSC_ROOT, and PETSC_ARCH="". Then, in the the *make all* command is issued, based on the final output of the *configure* command

   make PETSC_DIR=$PETSC_ROOT/petsc PETSC_ARCH=arch-linux-c-opt all

7. To view the source code, the *ex15* documenation is viewed on the petsc website.

8. To find and compile the example tutorials, the *KSP* Krylov solver examples can be found in:

   $PETSC_DIR/petsc/src/ksp/ksp/tutorials

   The specific program of interest can be made as:

   make clean && make ex15.ext

   where ext is either *c* or *f90* for the C or Fortran languages.