Time integration with PETSc

Kevin R. Green, Raymond J. Spiteri

Department of Computer Science
University of Saskatchewan

Math 314 - Department of Mathematics & Statistics
University of Saskatchewan
Fall 2017

Outline

Outline

Source code for example problems

All example codes from these lectures are available in the

```
Course Materials / PETSc / Examples
```

directory on Blackboard.
The (updated) pdf of these slides is also available on Blackboard.

PETSc

https://www.mcs.anl.gov/petsc/

PETSc (|ˈ petsi:|) stands for the Portable, Extensible Toolkit for Scientific computing.

It provides a suite of libraries for the numerical solution of partial differential equations (PDEs) and related problems.

Portable Extensible Toolkit for Scientific computing

- Architecture
  - tightly coupled (e.g. Cray, Blue Gene)
  - loosely coupled such as network of workstations
  - GPU clusters (many vector and sparse matrix kernels)
- Operating systems (Linux, Mac, Windows, BSD, proprietary Unix)
- Any compiler
- Real/complex, single/double/quad precision, 32/64-bit int
- Usable from C, C++, Fortran 77/90, Python, and MATLAB
- Free to everyone (2-clause BSD license), open development
- 10^{12} unknowns, full-machine scalability on Top-10 systems
- Same code runs performantly on a laptop

Portable **Extensible** Toolkit for Scientific computing

Philosophy: Everything has a plugin architecture

- Vectors, Matrices, Coloring/ordering/partitioning algorithms
- Preconditioners, Krylov accelerators
- Nonlinear solvers, Time integrators
- Spatial discretizations/topology

Portable Extensible **Toolkit** for Scientific computing

Algorithms, (parallel) debugging aids, low-overhead profiling.
Composability

- Try new algorithms by choosing from product space and composing existing algorithms (multilevel, domain decomposition, splitting).

Experimentation

- It is **not** possible to pick the best solver *a priori*.
  What will deliver best/competitive performance for a given physics, discretization, architecture, and problem size?
- PETSc's response: expose an algebra of composition so new solvers can be created at runtime.
- Important to keep solvers decoupled from physics and discretization because we also experiment with those.

Portable Extensible Toolkit for **Scientific computing**

- Computational Scientists
  - PyLith, Underworld, PFLOTRAN, MOOSE, Proteus, PyClaw, CHASTE
- Algorithm Developers (iterative methods and preconditioning)
- Package Developers
  - SLEPc, TAO, Deal.II, Libmesh, FEniCS, PETSc-FEM, MagPar, OOFEM, FreeCFD, OpenFVM
- Hundreds of tutorial-style examples
- Hyperlinked manual, examples, and manual pages for all routines
- Support from petsc-maint@mcs.anl.gov, petsc-users@mcs.anl.gov

Role of PETSc

Developing parallel, nontrivial PDE solvers that deliver high performance is still difficult and requires months (or even years) of concentrated effort.

PETSc is a toolkit that can ease these difficulties and reduce the development time, but it is not a black-box PDE solver, nor a silver bullet.

— Barry Smith

PETSc installation

Installing on your own machine (instructions to follow)

Installing on a cluster:

- Don't do it (generally).
- Use pre-built install.
- Consult with computing staff if not available.

Obtaining PETSc source code

- Specific releases, zipped tarballs
  `http://www.mcs.anl.gov/petsc/download/`
- Git repository [preferred]
  `https://bitbucket.org/petsc/petsc/`

Why is Git preferred?

- Public development repository, you can get *any* development version.
- All releases are just tags: eg. v3.7.6
- Easily rollback changes and releases.

Unpacking PETSc

Clone from repository:

```
$ git clone https://bitbucket.org/petsc/petsc.git
$ git checkout v3.7.6
```

OR

Unpack the archive:

```
$ tar xzf petsc.tar.gz
```

Configuring PETSc

- Set environment variable `PETSC_DIR` to the installation root directory
- Choose a value for `PETSC_ARCH` (perhaps `c-debug` for a standard debug build)
- Run the configuration utility
  - `$PETSC_DIR/configure [--help]`
- May require additional dependencies
  - Read the output message if the configure fails, should tell you why, and suggest configure options to help

External libraries

External libraries can be downloaded during the configuration phase.

For example, MUMPS and SuperLU_dist (efficient algorithms for direct solving of linear systems) can be obtained with

```
--download-mumps --download-scalapack
```

and

```
--download-superlu_dist --download-metis --download-parmetis
```

respectively.

Buidling PETSc

After successful configure, run

```
$ make
$ make test
$ make streams
```

Enabling PETSc

PETSc requires a couple of environment variables to be set:

- `PETSC_DIR` - the top level directory of PETSc
- `PETSC_ARCH` - the *architecture* for which PETSc has been built

Allows multiple `ARCH` types built on a given system, which can easily be swapped for your specific application code.
ie., having a version configured with debugging flags enabled, and a version without.

Generally on a cluster, `PETSC_ARCH` is left empty, and only an optimized build is maintained.

Using PETSc on plato.usask.ca

I've built a version of PETScfor use throughout these lectures. Add the following to your ~/.bashrc file:

```
source /home/krg958/petsc/math314.env
```

This sets up the required environment for compiling PETSc application programs. The programs can then be run as any other MPI programs. eg. with the Slurm job scheduler (template given with examples):

```
$ sbatch submit_template.slurm
```

More info about submitting jobs on plato can be found here
It is important that you edit at least the following for each submission:

- executable name
- maximum runtime
- maximum RAM

Installing yourself

Exercise 1: Personal installation

Install `PETSc v3.7.6` on a machine you use regularly. Set it up with the following properties

- configured with MUMPS with debugging symbols
- `PETSC_ARCH=MUMPS-debug`

Install a second version with

- configured with MUMPS and no debugging symbols
- `PETSC_ARCH=MUMPS-opt`

Testing your PETSc environment

Exercise 2: Testing memory bandwidth of your system

With `PETSC_DIR` and `PETSC_ARCH` set appropriately for your current system:

- Build the `MPIVersion.c` example code.
- Run the `MPIVersion` executable. What kind of memory bandwidth does it show? What does *memory bandwidth* mean?
- Try running with 1, 2, 4, 8, 16, 32 cores.

Recall that running jobs on a cluster should be done by submission with the appropriate job scheduler.

Outline

Outline

Outline

Outline

Outline