PETSc for Python

Release X.X.X

Lisandro Dalcin

April 05, 2011

Contents

Abstract

This document describes petsc4py, a Python port to the PETSc libraries.
PETSc (the Portable, Extensible Toolkit for Scientific Computation) is a suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations. It employs the MPI standard for all message-passing communication.
This package provides an important subset of PETSc functionalities and uses NumPy to efficiently manage input and output of array data.
A *good friend* of petsc4py is:

- mpi4py: Python bindings for MPI, the *Message Passing Interface*.

Other two projects depend on petsc4py:

- slepc4py: Python bindings for SLEPc, the *Scalable Library for Eigenvalue Problem Computations*.
- tao4py: Python bindings for TAO, the *Toolkit for Advanced Optimization*.

1 Overview

PETSc is a suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations. It employs the MPI standard for all message-passing communication.

PETSc is intended for use in large-scale application projects [petsc-efficient], and several ongoing computational science projects are built around the PETSc libraries. With strict attention to component interoperability, PETSc facilitates the integration of independently developed application modules, which often most naturally employ different coding styles and data structures.

PETSc is easy to use for beginners [petsc-user-ref]. Moreover, its careful design allows advanced users to have detailed control over the solution process. PETSc includes an expanding suite of parallel linear and nonlinear equation solvers that are easily used in application codes written in C, C++, and Fortran. PETSc provides many of the mechanisms needed within parallel application codes, such as simple parallel matrix and vector assembly routines that allow the overlap of communication and computation. In addition, PETSc includes growing support for distributed arrays.

1.1 Features

XXX To be written ...

1.2 Components

PETSc components provide the functionality required for many parallel solutions of PDE's.

Vec Provides the vector operations required for setting up and solving large-scale linear and nonlinear problems. Includes easy-to-use parallel scatter and

gather operations, as well as special-purpose code for handling ghost points for regular data structures.

Mat A large suite of data structures and code for the manipulation of parallel sparse matrices. Includes four different parallel matrix data structures, each appropriate for a different class of problems.

PC A collection of sequential and parallel preconditioners, including (sequential) ILU(k), LU, and (both sequential and parallel) block Jacobi, overlapping additive Schwarz methods and (through BlockSolve95) ILU(0) and ICC(0).

KSP Parallel implementations of many popular Krylov subspace iterative methods, including GMRES, CG, CGS, Bi-CG-Stab, two variants of TFQMR, CR, and LSQR. All are coded so that they are immediately usable with any preconditioners and any matrix data structures, including matrix-free methods.

SNES Data-structure-neutral implementations of Newton-like methods for nonlinear systems. Includes both line search and trust region techniques with a single interface. Employs by default the above data structures and linear solvers. Users can set custom monitoring routines, convergence criteria, etc.

TS Code for the time evolution of solutions of PDEs. In addition, provides pseudo-transient continuation techniques for computing steady-state solutions.

2 Tutorial

XXX To be written ... Any contribution welcome!

3 Installation

3.1 Requirements

You need to have the following software properly installed in order to build *PETSc for Python*:

- Any MPI implementation [1] (e.g., MPICH or Open MPI), built with **shared libraries**.
- PETSc 2.3.2/2.3.3/3.0.0/3.1 release, built with **shared libraries** [2].
- Python 2.4/2.5/2.6/2.7 [3].
- NumPy package.

[1] Unless you have appropiatelly configured and built PETSc PETSc without MPI (configure option `--with-mpi=0`).

[2] In order to build PETSc with shared libraries, you have to pass `--with-shared` option to PETSc's **configure** script.

[3] You may need to use a parallelized version of the Python interpreter with some MPI-1 implementations (e.g. MPICH1).

3.2 Using pip or easy_install

If you already have a working PETSc, set environment variables

`PETSC_DIR` and perhaps `PETSC_ARCH` to appropriate values:

```
$ export PETSC_DIR=/path/to/petsc
$ export PETSC_ARCH=linux-gnu
```

Note: If you do not set these environment variables, the install process will attempt to download and install PETSc for you.

Now you can use **pip**:

```
$ [sudo] pip install [--user] petsc4py
```

Alternatively, you can use *setuptools* **easy_install** (deprecated):

```
$ [sudo] easy_install petsc4py
```

3.3 Using distutils

Downloading

The *PETSc for Python* package is available for download at the project website generously hosted by Google Code. You can use **wget** to get a release tarball:

```
$ wget http://petsc4py.googlecode.com/files/petsc4py-X.X.X.tar.gz
```

Building

After unpacking the release tarball:

```
$ tar -zxf petsc4py-X.X.X.tar.gz
$ cd petsc4py-X.X.X
```

the distribution is ready for building.

Note: **Mac OS X** users employing a Python distribution built with **universal binaries** may need to set the environment variables `MACOSX_DEPLOYMENT_TARGET`, `SDKROOT`, and `ARCHFLAGS` to appropriate values. As an example, assume your Mac is running **Snow Leopard** on a **64-bit Intel** processor and you want to override the hard-wired cross-development SDK in Python configuration, your environment should be modified like this:

```
$ export MACOSX_DEPLOYMENT_TARGET=10.6
$ export SDKROOT=/
$ export ARCHFLAGS='-arch x86_64'
```

Some environmental configuration is needed to inform the location of PETSc. You can set (using **setenv**, **export** or what applies to you shell or system) the environmental variables

`PETSC_DIR`, and `PETSC_ARCH` indicating where you have built/installed PETSc:

```
$ export PETSC_DIR=/usr/local/petsc
$ export PETSC_ARCH=linux-gnu
```

Alternatively, you can edit the file `setup.cfg` and provide the required information below the `[config]` section:

```
[config]
petsc_dir  = /usr/local/petsc
petsc_arch = linux-gnu
...
```

Finally, you can build the distribution by typing:

```
$ python setup.py build
```

Installing

After building, the distribution is ready for installation.

You can do a site-install type:

```
$ python setup.py install
```

or, in case you need root privileges:

```
$ su -c 'python setup.py install'
```

This will install the `petsc4py` package in the standard location `prefix/lib/pythonX.X/site-packages`.

You can also do a user-install type. Threre are two options depending on the target Python version.

- For Python 2.6 and up:

  ```
  $ python setup.py install --user
  ```

- For Python 2.5 and below (assuming your home directory is available through the `HOME` environment variable):

  ```
  $ python setup.py install --home=$HOME
  ```

  and then add `$HOME/lib/python` or `$HOME/lib64/python` to your `PYTHONPATH` environment variable.

References

[petsc-user-ref] Satish Balay, Kris Buschelman, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith and Hong Zhang. PETSc Users Manual. ANL-95/11 - Revision 2.1.5. Argonne National Laboratory. 2004

[petsc-efficient] Satish Balay, Victor Eijkhout, William D. Gropp, Lois Curfman McInnes and Barry F. Smith. Efficient Management of Parallelism in Object Oriented Numerical Software Libraries. Modern Software Tools in Scientific Computing. E. Arge, A. M. Bruaset and H. P. Langtangen, editors. 163–202. Birkhauser Press. 1997.