PyPetsc Python Bindings - Reference Manual

version: 1.0

Kishor Gawande
kishor.gawande@nicta.com.au

December 1, 2006

1 Overview

PyPetsc is a Python wrapper module built on top of PETSc. Using PyPetsc, you can access the PETSc API in your Python scripts. This module also includes examples in Python, which replicate the C examples distributed with PETSc.

This document is useful for developers who want to use the PETSc library in their Python code. It covers installation, APIs, examples, Howtos and bugs. Some familiarity with the PETSc API is assumed. It does not cover design aspects of PyPetsc. These can be found the PyPetsc Design Reference Manual.

2 Supported Platforms

PyPetsc is currently being tested on following platforms.

- Mac OS X (10.4 with additional packages installed from darwinports)
- Linux Ubuntu 6.06

Support for other platforms is planned and will be added in the near future.

3 Installation

PyPetsc depends on a number of external packages. Please ensure that these packages are installed on your system. In the case of PETSc, follow the instructions given below *exactly*.

- Python version 2.4 or later.
- PETSc version 2.3.2-p3 or later. The source distribution of this version is included in elefant/externalpackages. Alternatively, your can download it from:
  http://www-unix.mcs.anl.gov/petsc/petsc-as/download/index.html

  Note: the current release of PyPetsc has not been tested with versions of PETSc other than above. So it is safe to use the source distribution included in the externalpackages directory.

  ```
  In the directory you wish to install PETSc:
  $ gunzip -c petsc-2.3.2-p3.tar.gz | tar -xof -

  $ cd petsc-2.3.2-p3

  $ export PETSC_DIR=$PWD
  #If you have csh then use setenv instead

  ## Define PETSC_DIR variable into environment rc file of your shell

  $ ./config/configure.py  --with-cc=gcc   --download-f-blas-lapack=1 \
  --download-mpich=1 --with-shared --with-dynamic --with-python \
  --with-clanguage=C++ --with-c-support \

  $ make all test

  Set environment variable PETSC_ARCH
  PETSC_ARCH: this environment/make variable is used to specify the
  ```

```
configuration that should currently be used. It corresponds to the
configuration files in bmake/${PETSC_ARCH}.

I have the following configuration on my MAC OS X
$ export PETSC_ARCH=darwin8.7.0-c-debug
```

- ctypes
  The ctypes package is included in Python 2.5 by default. If you are using older version of Python then follow these instructions:

  Download the latest source distribution of ctypes from
  http://sourceforge.net/projects/ctypes/

  To install ctypes from source, unpack the distribution, enter the ctypes-x.y.z source directory, and type:

  ```
  $ python setup.py build
  $ python setup.py test
  $ python setup.py install
  ```

- GCC - XML
  The ctypes code generator uses GCC-XML (from http://www.gccxml.org/) to parse C header files into an XML description. **Unfortunately the latest official version 0.6.0 does not fullfill the ctypes code generator's requirements.** Therefore please download the latest developer version of GCC-XML as follows:

  The source is accessed by checking out a read-only version of the GCC-XML source code. Use the following commands:

  ```
  cvs -d :pserver:anoncvs@www.gccxml.org:/cvsroot/GCC\_XML login
  (just press enter when prompted for a password)

  Follow this command by checking out the source code:
  cvs -d :pserver:anoncvs@www.gccxml.org:/cvsroot/GCC\_XML co gccxml
  ```

  Follow the installation instructions from the website at http://www.gccxml.org/HTML/Install.html

- ctypes code generator
  There is currently no official release for the code generator, but you can get it from the code generator SVN repository at http://svn.python.org/projects/ctypes/trunk/ctypeslib/

  ```
  $ svn checkout http://svn.python.org/projects/ctypes/trunk/ctypeslib/ ctypeslib
  $ cd ctypeslib
  $ python setup.py build
  $ python setup.py install
  ```

  This will install h2xml.py and xml2py.py Python scripts in your Python site-packages folder.

After installing the above packages, install PyPetsc:

```
$ cd $ELEFANT_DIR/extensionmodules/pypetsc
$ python setup.py install
```

Now you are ready to use PyPetsc!

4 Directory Structure

We briefly describe the directory structure and the files included as a part of the PyPetsc distribution:

- `$ELEFANT_DIR/extensionmodules/pypetsc/`

  This folder contains two main files, *petsc.py* and *pypetsc.py*. petsc.py is generated during installation. pypetsc.py is wrapper file. It provides easy-to-use Python interface to PETSc functions. It is recommended that you import pypetsc.py in your Python programs. setup.py is installation script and it installs above files into python site-packages path.

- `$ELEFANT_DIR/extensionmodules/pypetsc/doc`

  This folder contains documentation files in tex format.

- `$ELEFANT_DIR/extensionmodules/pypetsc/codegen`

  This folder contains script for generating petsc.py python wrapper.

- `$ELEFANT_DIR/extensionmodules/pypetsc/examples`

  This folder contains Python examples for Linear Solvers (KSP), Vector operations and Matrix operations. Other examples from the PETSc distribution will be added soon. These Python examples are exact copies of the C examples available in the PETSc source distribution. To run a KSP example

  ```
  $ cd $ELEFANT_DIR/extensionmodules/pypetsc/examples/ksp
  $ python ex1.py
  ```

5 Getting started

In order to use PyPetsc effectively you need to be familiar with PETSc. Therefore it is highly recommended that you read the PETSc user manual before using PyPetsc. Comprehensive documentation for PETSc is available at http://www-unix.mcs.anl.gov/petsc/petsc-as/

5.1 Your first PyPetsc program

Include the following modules in your Python script:

```
from ctypes import *
import pypetsc
```

Create a help string:

```
help = "Solves a tridiagonal linear system with KSP.\n\n"
```

Most PyPetsc programs begin with a call to `pypetsc.wPetscInitialize(help)` This function initializes PETSc and MPI. It is an easy-to-use wrapper for the standard PETSc function `PetscInitialize(int *argc,char ***argv,char *file,char *help)`. This function also passes the command line arguments to `PetscInitialize()`. 'help' is an optional character string that will be printed if the program is run with the -help option. For more information about this API please refer to the PETSc documentation.

All PETSc programs should call `PetscFinalize()` as their final (or nearly final) statement. In PyPetsc, you can need to invoke `pypetsc.PetscFinalize()`.

5.2 Simple PyPetsc Example

We walk through a simple PyPetsc example in this section. It solves the one dimensional Laplacian problem with finite differences. The code can be found in

`$ELEFANT_DIR/extensionmodules/pypetsc/examples/ksp/ex1.py.`

The corresponding C version can be found in

`$PETSC_DIR/src/ksp/ksp/examples/tutorials/ex1.c.`

This sequential code illustrates the solution of a linear system with KSP, the interface to the preconditioners, the Krylov subspace methods, and the direct linear solvers of PETSc.

```
##
# Solves a tridiagonal linear system with pypetsc.KSP in Python
# Concepts: pypetsc.KSP solving a system of linear equations
# Processors: 1
# C Example: ${PETSC_DIR}/src/ksp/ksp/examples/tutorials/ex1.c
#

help = "Solves a tridiagonal linear system with KSP.\n\n"

#import modules
from ctypes import *
import pypetsc

## Main routine
#
def main():

    # Initialize petsc
    pypetsc.wPetscInitialize(help);

    size = pypetsc.PetscMPIInt()
    ierr = pypetsc.MPI_Comm_size(pypetsc.PETSC_COMM_WORLD, byref(size)); pypetsc.CHKERRQ(ierr);

    if (size.value != 1) :
        print "This is a uniprocessor example only!";
        return;

    n = pypetsc.PetscInt(10)
    ierr = pypetsc.PetscOptionsGetInt(pypetsc.PETSC_NULL, "-n", byref(n), pypetsc.PETSC_NULL);
    pypetsc.CHKERRQ(ierr);

    # Create vectors. Note that we form 1 vector from scratch and
    # then duplicate as needed.
    x = pypetsc.Vec()  # Approx solution
    ierr = pypetsc.VecCreate(pypetsc.PETSC_COMM_WORLD, x); pypetsc.CHKERRQ(ierr);
    ierr = pypetsc.PetscObjectSetName(cast(x, pypetsc.PetscObject), "Solution");
    pypetsc.CHKERRQ(ierr);
    ierr = pypetsc.VecSetSizes(x, pypetsc.PETSC_DECIDE, n); pypetsc.CHKERRQ(ierr);
    ierr = pypetsc.VecSetFromOptions(x); pypetsc.CHKERRQ(ierr);
    b = pypetsc.Vec() # RHS
    ierr = pypetsc.VecDuplicate(x, b); pypetsc.CHKERRQ(ierr);

    u = pypetsc.Vec() # Exact solution
    ierr = pypetsc.VecDuplicate(x, u); pypetsc.CHKERRQ(ierr);

    # Create matrix. When using pypetsc.MatCreate(), the matrix format can
    # be specified at runtime.
    # Performance tuning note: For problems of substantial size,
    # pre-allocation of matrix memory is crucial for attaining good
    # performance. See the matrix chapter of the users manual for details.
    A = pypetsc.Mat() # Linear system matrix
    ierr = pypetsc.MatCreate(pypetsc.PETSC_COMM_WORLD, A); pypetsc.CHKERRQ(ierr);
    ierr = pypetsc.MatSetSizes(A, pypetsc.PETSC_DECIDE, pypetsc.PETSC_DECIDE, n, n);
    pypetsc.CHKERRQ(ierr);
    ierr = pypetsc.MatSetFromOptions(A); pypetsc.CHKERRQ(ierr);

    # Assemble matrix
    col = pypetsc.PetscIntArray(3)
    value = pypetsc.PetscScalarArray(3)

    neg_one = pypetsc.PetscScalar(-1.0)
    one = pypetsc.PetscScalar(1.0)

    value[0] = -1.0
    value[1] = 2.0
    value[2] = -1.0

    for i in range (n.value - 1):
         col[0] = i - 1
         col[1] = i
         col[2] = i + 1
         ierr = pypetsc.MatSetValues(A, 1, byref(pypetsc.PetscInt(i)), 3, col, value, \
                                                             pypetsc.INSERT_VALUES);
         pypetsc.CHKERRQ(ierr);

    i = n.value - 1;
    col[0] = n.value - 2
    col[1] = n.value - 1
    ierr = pypetsc.MatSetValues(A, 1, byref(pypetsc.PetscInt(i)), 2, col, value, \
                                                        pypetsc.INSERT_VALUES);
    pypetsc.CHKERRQ(ierr);

    i = 0;
    col[0] = 0
    col[1] = 1
    value[0] = 2.0
    value[1] = -1.0
    ierr = pypetsc.MatSetValues(A, 1, byref(pypetsc.PetscInt(i)), 2, col, value, \
                                                              pypetsc.INSERT_VALUES);
    pypetsc.CHKERRQ(ierr);

    ierr = pypetsc.MatAssemblyBegin(A, pypetsc.MAT_FINAL_ASSEMBLY);
    pypetsc.CHKERRQ(ierr);
    ierr = pypetsc.MatAssemblyEnd(A, pypetsc.MAT_FINAL_ASSEMBLY);
    pypetsc.CHKERRQ(ierr);

    # Set exact solution; then compute right-hand-side vector.
    ierr = pypetsc.VecSet(u, one); pypetsc.CHKERRQ(ierr);
    ierr = pypetsc.MatMult(A, u, b); pypetsc.CHKERRQ(ierr);

    # - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    # Create the linear solver and set various options
    # - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

    # Create linear solver context
    ksp = pypetsc.KSP(); # linear solver context
    ierr = pypetsc.KSPCreate(pypetsc.PETSC_COMM_WORLD, ksp);pypetsc.CHKERRQ(ierr);

    # Set operators. Here the matrix that defines the linear system
    # also serves as the preconditioning matrix.
    ierr = pypetsc.KSPSetOperators(ksp, A, A, pypetsc.DIFFERENT_NONZERO_PATTERN);
    pypetsc.CHKERRQ(ierr);

    # Set linear solver defaults for this problem (optional).
    # - By extracting the pypetsc.KSP and pypetsc.PC contexts from the pypetsc.KSP context,
    #   we can then directly call any pypetsc.KSP and pypetsc.PC routines to set
    #   various options.
    # - The following four statements are optional; all of these
    #   parameters could alternatively be specified at runtime via
    # pypetsc.KSPSetFromOptions();
    pc = pypetsc.PC();   # preconditioner context
    ierr = pypetsc.KSPGetPC(ksp, pc); pypetsc.CHKERRQ(ierr);
    ierr = pypetsc.PCSetType(pc, pypetsc.PCJACOBI); pypetsc.CHKERRQ(ierr);
    ierr = pypetsc.KSPSetTolerances(ksp, 1.e-7, pypetsc.PETSC_DEFAULT, pypetsc.PETSC_DEFAULT,  \
                                                              pypetsc.PETSC_DEFAULT);
    pypetsc.CHKERRQ(ierr);

    ierr = pypetsc.KSPSetFromOptions(ksp); pypetsc.CHKERRQ(ierr);

    # - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    # Solve the linear system
    # - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

    ierr = pypetsc.KSPSolve(ksp, b, x); pypetsc.CHKERRQ(ierr);
    # View solver info
    ierr = pypetsc.KSPView(ksp,  pypetsc.PETSC_VIEWER_STDOUT_WORLD()); pypetsc.CHKERRQ(ierr);

    # - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    # Check solution and clean up
    # - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

    # Check the error
    ierr = pypetsc.VecAXPY(x, neg_one, u);pypetsc.CHKERRQ(ierr);

    norm = pypetsc.PetscReal(0); # norm of solution erro
    its = pypetsc.PetscInt(0);
    ierr = pypetsc.VecNorm(x, pypetsc.NORM_2, byref(norm)); pypetsc.CHKERRQ(ierr);
    ierr = pypetsc.KSPGetIterationNumber(ksp, byref(its)); pypetsc.CHKERRQ(ierr);

    print "Norm of error ", norm.value, "Iterations ", its.value

    # Free work space. All PETSc objects should be destroyed when
    # they are no longer needed
    ierr = pypetsc.VecDestroy(x); pypetsc.CHKERRQ(ierr);
    ierr = pypetsc.VecDestroy(u); pypetsc.CHKERRQ(ierr);
    ierr = pypetsc.VecDestroy(b);pypetsc.CHKERRQ(ierr);
    ierr = pypetsc.MatDestroy(A);pypetsc.CHKERRQ(ierr);
    ierr = pypetsc.KSPDestroy(ksp);pypetsc.CHKERRQ(ierr);
    ierr = pypetsc.PetscFinalize();pypetsc.CHKERRQ(ierr);


## Startup routine
if __name__ == "__main__":
    main()
```

This program generates the following output:

```
$elefant/extensionmodules/pypetsc/examples/ksp % python ex1.py
KSP Object:
  type: gmres
    GMRES: restart=30, using Classical (unmodified) Gram-Schmidt Orthogonalization
    with no iterative refinement
    GMRES: happy breakdown tolerance 1e-30
  maximum iterations=10000, initial guess is zero
  tolerances:  relative=1e-07, absolute=1e-50, divergence=10000
  left preconditioning
PC Object:
  type: jacobi
  linear system matrix = precond matrix:
  Matrix Object:
    type=seqaij, rows=10, cols=10
    total: nonzeros=28, allocated nonzeros=50
      not using I-node routines
Norm of error  1.88411095042e-15 Iterations  5
```

Here are the key differences between the C and the Python programs:

- Instead of #include files use import module

- Use `wPetscInitialize()` wrapper function instead of `PetscInitialize()`
- Prefix PETSc functions with name space `pypetsc`
- To pass an argument by reference, use ctypes `byref()` function
- To create an array of int or float, use the wrapper functions

5.3 API Description

Most of the APIs are the same as in PETSc, therefore you can refer to the PETSc reference manual. However, we have added the following wrappers over the standard PETSc APIs. They provide an easy-to-use interface to PETSc from within Python.

5.4 wPetscInitialize(help)

This function initializes PETSc and MPI. It is an easy-to-use wrapper for the standard PETSc function `PetscInitialize(int *argc,char ***argv,char *file,char *help)`. This function also passes the command line arguments to `PetscInitialize()`. 'help' is an optional character string that will be printed if the program is run with the -help option. For more information about this API please refer to the PETSc documentation.

Input Parameters:

- help: [optional] A help message to be printed. Use `PETSC_NULL` if you don't want a message.

Output:

- None.

5.5 array PetscMalloc(n, type)

This function is semantically the same as the one provided by PETSc, however, it has a slightly different syntax. It allocates a memory block, and returns a ctype array of the appropriate type. If you want to allocate 10 integer bytes then don't pass `n` as `10 * sizeof(PetscInt)`, instead call this function as: `array = PetscMalloc(10, PetscInt)`

Input Parameters:

- `n`: Number of bytes to be allocated.
- `type`: Data type, could be `PetscScalar`, `PetscInt`, etc.

Output:

- Returns a ctype array of '`type`'.

Examples:

- `extensionmodules/pypetsc/examples/ksp/ex3.py`

5.6 PetscFree(buff)

This function frees a memory block that was previously allocated by `PetscMalloc()`.

Input Parameters:

- `buff`: Address of buffer.

Output:

- None.

Examples:

- `extensionmodules/pypetsc/examples/ksp/ex3.py`

5.7 a VecGetArray(x)

This function is semantically the same as the one provided by PETSc, but it has a slightly different syntax. It returns a pointer to a contiguous array that contains this particular processor's portion of the vector data.

Input Parameters:

- `x`: The vector

Output:

- Returns location to put pointer to the array.

Examples:

- `extensionmodules/pypetsc/examples/vec/ex3.py`
- `extensionmodules/pypetsc/examples/vec/ex5.py`
- `extensionmodules/pypetsc/examples/vec/ex6.py`

5.8 VecRestoreArray(x, a)

Restores a vector after VecGetArray() has been called.

Input Parameters:

- `x`: The vector
- `a`: Location of a pointer to an array, returned by `VecGetArray()`.

Output:

- None.

Examples:

- `extensionmodules/pypetsc/examples/vec/ex3.py`
- `extensionmodules/pypetsc/examples/vec/ex5.py`
- `extensionmodules/pypetsc/examples/vec/ex6.py`

5.9 PETSC_VIEWER_STDOUT_WORLD()

Use this function to `#define PETSC_VIEWER_STDOUT_WORLD` Creates an ASCII PetscViewer shared by all processors in a communicator.

Examples:

- `extensionmodules/pypetsc/examples/ksp/ex1.py`
- `extensionmodules/pypetsc/examples/ksp/ex3.py`

5.10 array PetscIntArray(n)

Returns an array of type `PetscInt`.

Input Parameters:

- `n`: Size of the array.

Output:

- An array of type `PetscInt`.

Examples:

- `extensionmodules/pypetsc/examples/ksp/ex1.py`

5.11 array PetscScalarArray(n)

Returns an array of type `PetscScalar`.

Input Parameters:

- `n`: Size of the array.

Output:

- An array of type `PetscScalar`.

Examples:

- `extensionmodules/pypetsc/examples/ksp/ex1.py`

5.12 PetscObject ToPetscObject(obj)

Converts the input object type to PetscObject type.

Input Parameters:

- `obj`: An object which needs to be cast as `PetscObject`.

Output:

- An object of type `PetscObject`.

Examples:

- `extensionmodules/pypetsc/examples/ksp/ex3.py`

5.13 addr GetArrayAddressByOffset(array, offset, type)

Returns the address of ’`array`’. The address is calculated as ’`offset`’ + base address of ’`array`’. The returned pointer is of type ’`type`’.

Input Parameters:

- `array`: The input array.
- `offset`: A value that the pointer will be advanced by.
- `type`: Return type of address.

Output:

- Returns a pointer of type ’`type`’ to the address of ’`array`’ + ’`offset`’.

Examples:

- `extensionmodules/pypetsc/examples/mat/ex2.py`

6 HOW TOs

How to pass parameters by reference to a function?

ctypes exports the `byref()` function, which is used to pass parameters by reference. The same effect can be achieved with the `pointer()` function, although `pointer()` does a lot more work since it constructs a real pointer object. Therefore it is faster to use `byref()` if you don’t need a pointer object in Python itself:

```
size = pypetsc.PetscMPIInt()
ierr = pypetsc.MPI_Comm_size(pypetsc.PETSC_COMM_WORLD, byref(size));
```

How to create an Array?

Use the wrapper function `PetscIntArray()`.

```
col = pypetsc.PetscIntArray(3)
col[0] = i - 1
col[1] = i
col[2] = i + 1

value = pypetsc.PetscScalarArray(3)
value[0] = -1.0
value[1] = 2.0
value[2] = -1.0
```

How to use `PetscMalloc()` and `PetscFree()`?

Here is an example which dynamically allocates an array of `PetscInt`:

```
m = 20;
rows = pypetsc.PetscMalloc(4 * m, pypetsc.PetscInt)

i=0;
while (i < m+1) :
    rows[i] = i; # bottom
     rows[3 * m - 1 + i] = m * (m + 1) + i;
     i = i + 1;

print rows[0]

# Free memory
pypetsc.PetscFree(rows)

print rows[0]
```

How to advance a pointer by some offset value?

C example

```
PetscScalar *x

ierr = PetscMalloc(size*sizeof(PetscScalar),&x);CHKERRQ(ierr);
for (i=0; i<size; i++) {
  x[i] = 1.0;
}

. . .

ierr = VecCreateSeqWithArray(MPI_COMM_SELF,size2,x+size1,&X2);
```

Note address of `x + size1` is passed to `VecCreateSeqWithArray`. To do this in Python

```
x = pypetsc.PetscMalloc(size * size, pypetsc.PetscScalar);
i = 0
while (i < size) :
    x[i] = 1.0
    i = i + 1

# Get the address of x at offset size1
ptr = pypetsc.GetArrayAddressByOffset(x, size1, c_double)
ierr = pypetsc.VecCreateSeqWithArray(pypetsc.MPI_COMM_SELF, size2, ptr, byref(X2));
```

How to create a callback function in Python?

Refer to `extensionmodules/pypetsc/examples/ksp/dynamic/ex1callbackPY.py`. Here is simple code snippet:

```
## Callback function
#
def converge(KSP,PetscInt,PetscReal,KSPConvergedReason, mctx):
    print "Call back - Interation", PetscInt
    return 0


# your main code
# Set the Python call back function
# Create function type object for callback function
CONV_FUN = CFUNCTYPE(pypetsc.PetscErrorCode, POINTER(pypetsc._p_KSP), c_int, c_double, /
                        POINTER(pypetsc.KSPConvergedReason), c_void_p)
fn_conv = CONV_FUN(converge)
ierr = pypetsc.KSPSetConvergenceTest(ksp, fn_conv, 0); pypetsc.CHKERRQ(ierr);
```

How to make a function exported by a shared library a callback function in Python?
Refer to `extensionmodules/pypetsc/examples/ksp/dynamic/ex1callbackC.py`. Here is a simple code snippet:

```
# Set convergence test call back function from dynamic library
# Load dynamic library using ctypes
callbackDLL = CDLL("libcallback.dylib")
# Create callable function prototype
CONV_FUN = CFUNCTYPE(pypetsc.PetscErrorCode, POINTER(pypetsc._p_KSP), c_int, c_double,  /
                              POINTER(pypetsc.KSPConvergedReason), c_void_p)
# Just cast the function pointer from library to prototype
fn_conv = cast(callbackDLL.test, CONV_FUN)
# Set callback
ierr = pypetsc.KSPSetConvergenceTest(ksp, fn_conv, 0); pypetsc.CHKERRQ(ierr);
```

7 Known Issues

- Some functions and constants from PETSc are not yet available in PyPetsc.
- Currently, PyPetsc has only been tested on MAC OS X and Linux Ubuntu 6.06 platform.
- PyPetsc has not been tested on any parallel platform.
- PyPetsc has not been tested for performance issues or memory leakages.