Fortran/C Interface: f2py

Andy Turner, ARCHER CSE Team
a.turner@epcc.ed.ac.uk

Reusing this material

http://www.archer.ac.uk
support@archer.ac.uk

Why interface to Fortran/C

- Provide glue to dynamically organise code
  - Complex software coordination provided by Python
- Performance of compiled codes with flexibility of Python
  - *e.g.* incorporate Python analysis and visualisation into existing codebase
  - Provide flexible way to extract results from code using Python
- Reuse code that you already have
  - Gradually introduce new functionality using Python

What is required?

- Name of external function
- Types of arguments to be passed from Python to external functions:
  - Integers, real numbers, arrays, characters?
- Sequence of arguments
- Are the arguments input parameters, output parameters or to be modified by the external function?
- Packaged in a way that can be imported by Python
- *f2py* provides a way to do this simply and quickly

f2py: Interfacing to Fortran

- Provides a way to describe external functions and their arguments
- Packages-up the external code in a way that can be imported and used by Python
- You need to provide:
  - The Fortran source code (to be compiled)
  - A file describing the external function and arguments (f2py can help you generate this)

Example: array_sqrt.f90

```
! Example Fortran: sqrt of array
subroutine array_sqrt(n, a_in, a_out)
	implicit none
	integer, intent(in) :: n
	real*8, dimension(n), intent(in) :: a_in
	real*8, dimension(n), intent(out) :: a_out
	integer :: i
	do i = 1, n
		a_out(i) = sqrt(a_in(i))
	end do
end subroutine array_sqrt
```

Create signature file

- f2py can try to create the signature file automatically:

```
f2py array_sqrt.f90 -m farray -h array_sqrt.pyf
```

- The Python module will be called: farray
- Signature in text file called: “array_sqrt.pyf”

Produce compiled library

- Once you have verified that the signature file is correct
- Use f2py to compile the library file that can be imported into Python:

```
f2py -c array_sqrt.pyf array_sqrt.f90
```

- Produces a library file called: farray.so

archer

epcc

Calling from Python

```
>>> from farray import array_sqrt
>>> import numpy as np
>>> a = np.array([1.0,2.0,3.0,4.0])
>>> array_sqrt(a)
array([ 1.        ,  1.41421356,  1.73205081,  2.        ])
```

archer

epcc

f2py: Interfacing to C

- f2py is the simplest way to interface C to Python
- Basic procedure is very similar to Fortran
- Differences:
  - You must write the signature file by hand
  - You must use the intent(c) attribute for all variables
  - You must define the function name with the intent(c) attribute
  - Only 1D arrays can be handled by C, if you pass a multidimensional array you must compute the correct index.
- Build in exactly the same way as Fortran example (but with different source code!)

Example: Signature file

```
python module farray
  interface
    subroutine array_sqrt(n,a_in,a_out)
      intent(c) :: array_sqrt
      intent(c)  ! Adds to all following definitions
      integer, optional,intent(in),check(len(a_in)>=n),depend(a_in) ::
n=len(a_in)
      real*8 dimension(n),intent(in) :: a_in
      real*8 dimension(n),intent(out),depend(n) :: a_out
    end subroutine array_sqrt
  end interface
end python module farray
```

Other Options for C

- Native Python interface
  - Fully-flexible and portable
  - Complex and verbose
  - Best if you are interfacing a large amount of code and/or have a large software development project
- Cython
  - Standard C-like Python (or Python-like C)
  - (I have never had much success…)
- SWIG
  - Very generic and feature-rich
  - Supports multiple languages other than Python (e.g. Perl, Ruby)

Summary

- f2py is a simple way to call Fortran/C code from Python
  - Simpler for Fortran than for C
  - Care needed when using multidimensional arrays in C
- Calling sequence is converted to something more Pythonic:

  `array_sqrt(n, a_in, a_out)`, becomes:
  `a_out = array_sqrt(a_in)`

- Fortran/C can give better performance than Python