UNIVERSITY of York

Getting along and working together

Fortran-Python Interoperability

Jacob Wilkins

Fortran **AND** Python working together?

- Two very different philosophies
- Two very different code-styles
- Two very different purposes
- Going to be hard to get them to work together

More likely than you think

- All possible thanks to F2Py
- F2Py now standard part of NumPy
- Makes Fortran modules importable into Python

I know what you’re thinking

- Great! So now I have to rewrite my codebase!

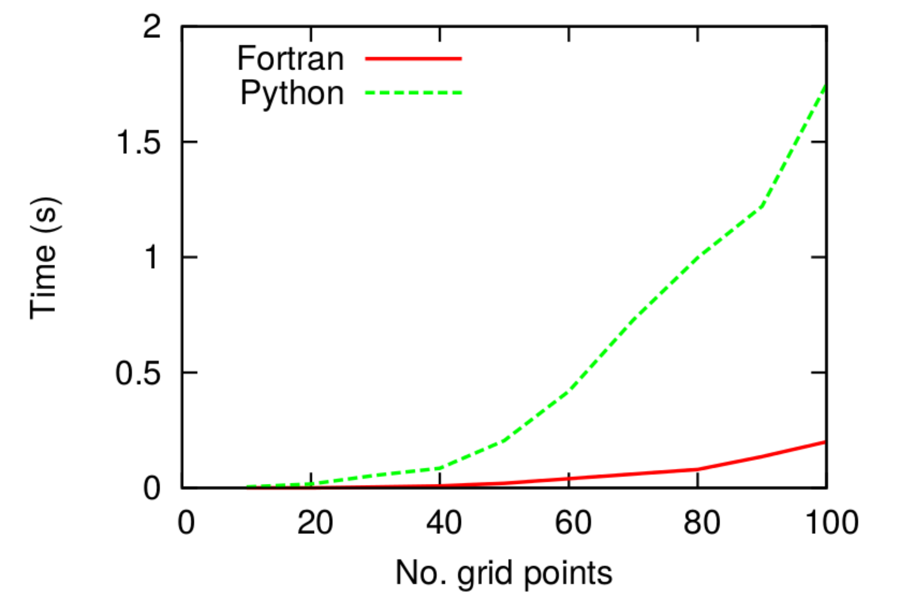
Not necessarily

- Optional imports
- Preprocessed directives (Perceived as comments by Fortran)
- Just an ordinary Fortran module

Why do we care?

- Python makes it faster to prototype code
- Python has many useful libraries and interfaces
- Fortran is faster to crunch numbers
- Fortran has strict typing (often a benefit)

Why do we care?



Hello, World!

- Let’s conform to stereotypes for a bit.
- We’re going to write “Hello, World!” as a subroutine in Fortran
- Compile with the command below
- Use: `from FILENAME import FUNCTION` to import my Fortran
- Call my function from a python script

```
       f2py -m OUTPUT_NAME --f90flags=FORTRAN_FLAGS \
F2PY_FLAGS -c F90_FILE
```

Here's one I made earlier

- Fortran (hello_fort.f90):

```fortran
subroutine hello()
  print*, "Hello, World!"
end subroutine hello
```

- Python (hello.py):

```python
from hello_fort import hello

hello()
```

- Bash:

```bash
f2py -m hello_fort -c hello_fort.f90; python hello.py
```

Your Turn

- Write a matrix multiplication as a function in a module in Fortran
- Compile with the command below
- Use: `from FILENAME import MODULE` to import your Fortran module
- Use: `print module_name.function_name.__doc__` to see the interface to your new wrapped routine
- Call the function as you would any ordinary Python function in line with the interface given by the docstring

```
      f2py -m OUTPUT_NAME --f90flags=FORTRAN_FLAGS \
F2PY_FLAGS -c F90_FILE
```

Caveats

- F2Py usually pretty good at dependencies
- May sometimes screw up
- We can give it extra instructions to guide it

Might not be happy

```fortran
subroutine swap(n, array, x, y, swapcount)
  !Swap indices x & y with each other
  !array is an array of length N
  implicit none
  integer, intent(in) :: n, x, y
  integer, intent(inout) :: swapcount
  integer, intent(inout) :: array(n)
  integer :: t
  t = array(x)
  array(x) = array(y)
  array(y) = t
  if ( x .ne. y ) then
     swapcount = swapcount + 1
  end if
end subroutine swap
```

Probably happy

```
subroutine swap(n, array, x, y, swapcount)
  !Swap indices x & y with each other
  !array is an array of length N
  implicit none
  integer, intent(in) :: n, x, y
  integer, intent(inout) :: swapcount
  integer, intent(inout) :: array(n)
  integer :: t
!f2py intent(inout) array
!f2py depend(n) array
  ...
  t = array(x)
  array(x) = array(y)
  array(y) = t
  if ( x .ne. y ) then
     swapcount = swapcount + 1
  end if
end subroutine swap
```

Changing values

- We have seen how we can pass arguments to Fortran functions
- What if we want to change variables?
- Interface to module gives module level variables (both ways)
- Treat them as you would Python module variables!

Here’s one I made earlier

- Fortran (`ModVarExam.f90`):

```
module variable
  integer :: a = 10
contains
  subroutine say()
    print*, a
  end subroutine say
end module variable
```

- Python (`ModVarExam.py`):

```
from ModVarExam import variable
print variable.a
variable.a = 17
variable.say()
```

- Bash:

```
f2py -m ModVarExam -c ModVarExam.f90; python ModVarExam.py
```

Your Turn

- Using the provided `duffing_fort.f90` and `duffing.py`
  - Perform a parameter scan of D
  - Pull the name of the output from the module
  - Use `npy.loadtxt` and `matplotlib` to plot the result of the oscillator

```
f2py -m duffing_fort -c duffing_fort.f90; python duffing.py
```

More Advanced Happiness

- F2Py does this for us automatically usually
- We can get an inside look at what it thinks

```
      f2py -m OUTPUT_NAME --f90flags=FORTRAN_FLAGS \
-h SIGNATURE_NAME F2PY_FLAGS F90_FILE
```

More Features

- Can build GUI interfaces to Fortran easily
- Can still use OpenMP and MPI in Fortran with python on top
- There are implementations of f2py which allow derived data types
- See: `https://github.com/jameskermode/f90wrap`

Example

- ParallelOMP + GUI + Fortran= Impossible

Example

- ParallelOMP + GUI + Fortran= Impossible
- Right?

Example

- ParallelOMP + GUI + Fortran= Impossible
- Right?
- I have provided a toy code I wrote as a demonstration of OMP
- Pushes data into Fortran from Python GUI
- Pulls data back to plot Fortran into GUI