Calling Fortran

June 1, 2016

```
In [1]: # Just to know last time this was run:
        import time
        print time.ctime()
```

```
Wed Jun  1 17:41:04 2016
```

1 I Calling Fortran from Python

This is part of the Python lecture given by Christophe Morisset at IA-UNAM. More informations at: http://python-astro.blogspot.mx/

```
In [2]: import numpy as np
```

The following is part of this excellent web page: http://nbviewer.ipython.org/github/jrjohansson/scientific-python-lectures/blob/master/Lecture-6A-Fortran-and-C.ipynb

```
In [3]: # simple python algorithm: example of a SLOW implementation
        # Why? Because the loop is implemented in python.
        def py_dcumsum(a):
            b = np.empty_like(a)
            b[0] = a[0]
            for n in range(1,len(a)):
                b[n] = b[n-1]+a[n]
            return b
```

```
In [4]: # The numpy version of the cumsum
        def numpy_cumsum(a):
            return np.cumsum(a)
```

We write here a fortran function with some special code to interact with python

```
In [5]: %%writefile dcumsum.f
        c File dcumsum.f
              subroutine dcumsum(a, b, n)
              double precision a(n)
              double precision b(n)
              integer n
```

```
        cf2py  intent(in) :: a
        cf2py  intent(out) :: b
        cf2py  intent(hide) :: n

               b(1) = a(1)
               do 100 i=2, n
                   b(i) = b(i-1) + a(i)
        100    continue
               end

Overwriting dcumsum.f


In [6]: # Compiling. On my OSX, gfortran is used
        !f2py --f77exec=gfortran -c dcumsum.f -m dcumsum

running build
running config_cc
unifing config_cc, config, build_clib, build_ext, build commands --compiler options
running config_fc
unifing config_fc, config, build_clib, build_ext, build commands --fcompiler optio
running build_src
build_src
building extension "dcumsum" sources
f2py options: []
f2py:> /var/folders/bb/jg97y_ln7cn8wbgbl8zs8rvr0000gn/T/tmpk6kMIp/src.macosx-10.5-x
creating /var/folders/bb/jg97y_ln7cn8wbgbl8zs8rvr0000gn/T/tmpk6kMIp/src.macosx-10.5
Reading fortran codes...
        Reading file 'dcumsum.f' (format:fix,strict)
Post-processing...
        Block: dcumsum
                        Block: dcumsum
Post-processing (stage 2)...
Building modules...
        Building module "dcumsum"...
                Constructing wrapper function "dcumsum"...
                  b = dcumsum(a)
        Wrote C/API module "dcumsum" to file "/var/folders/bb/jg97y_ln7cn8wbgbl8zs8
  adding '/var/folders/bb/jg97y_ln7cn8wbgbl8zs8rvr0000gn/T/tmpk6kMIp/src.macosx-10.
  adding '/var/folders/bb/jg97y_ln7cn8wbgbl8zs8rvr0000gn/T/tmpk6kMIp/src.macosx-10.
copying /Users/christophemorisset/anaconda/lib/python2.7/site-packages/numpy/f2py/s
copying /Users/christophemorisset/anaconda/lib/python2.7/site-packages/numpy/f2py/s
build_src: building npy-pkg config files
running build_ext
customize UnixCCompiler
customize UnixCCompiler using build_ext
customize Gnu95FCompiler
Found executable /usr/local/bin/gfortran
```

```
customize Gnu95FCompiler
customize Gnu95FCompiler using build_ext
building 'dcumsum' extension
compiling C sources
C compiler: gcc -fno-strict-aliasing -I/Users/christophemorisset/anaconda/include -

creating /var/folders/bb/jg97y_ln7cn8wbgbl8zs8rvr0000gn/T/tmpk6kMIp/var
creating /var/folders/bb/jg97y_ln7cn8wbgbl8zs8rvr0000gn/T/tmpk6kMIp/var/folders
creating /var/folders/bb/jg97y_ln7cn8wbgbl8zs8rvr0000gn/T/tmpk6kMIp/var/folders/bb
creating /var/folders/bb/jg97y_ln7cn8wbgbl8zs8rvr0000gn/T/tmpk6kMIp/var/folders/bb/
creating /var/folders/bb/jg97y_ln7cn8wbgbl8zs8rvr0000gn/T/tmpk6kMIp/var/folders/bb/
creating /var/folders/bb/jg97y_ln7cn8wbgbl8zs8rvr0000gn/T/tmpk6kMIp/var/folders/bb/
creating /var/folders/bb/jg97y_ln7cn8wbgbl8zs8rvr0000gn/T/tmpk6kMIp/var/folders/bb/
compile options: '-I/var/folders/bb/jg97y_ln7cn8wbgbl8zs8rvr0000gn/T/tmpk6kMIp/src.
gcc: /var/folders/bb/jg97y_ln7cn8wbgbl8zs8rvr0000gn/T/tmpk6kMIp/src.macosx-10.5-x86
In file included from /Users/christophemorisset/anaconda/lib/python2.7/site-package
                 from /Users/christophemorisset/anaconda/lib/python2.7/site-package
                 from /Users/christophemorisset/anaconda/lib/python2.7/site-package
                 from /var/folders/bb/jg97y_ln7cn8wbgbl8zs8rvr0000gn/T/tmpk6kMIp/sr
                 from /var/folders/bb/jg97y_ln7cn8wbgbl8zs8rvr0000gn/T/tmpk6kMIp/sr
/Users/christophemorisset/anaconda/lib/python2.7/site-packages/numpy/core/include/n
 #warning "Using deprecated NumPy API, disable it by " \
  ^
/var/folders/bb/jg97y_ln7cn8wbgbl8zs8rvr0000gn/T/tmpk6kMIp/src.macosx-10.5-x86_64-2
 static int f2py_size(PyArrayObject* var, ...)
            ^
gcc: /var/folders/bb/jg97y_ln7cn8wbgbl8zs8rvr0000gn/T/tmpk6kMIp/src.macosx-10.5-x86
In file included from /Users/christophemorisset/anaconda/lib/python2.7/site-package
                 from /Users/christophemorisset/anaconda/lib/python2.7/site-package
                 from /Users/christophemorisset/anaconda/lib/python2.7/site-package
                 from /var/folders/bb/jg97y_ln7cn8wbgbl8zs8rvr0000gn/T/tmpk6kMIp/sr
                 from /var/folders/bb/jg97y_ln7cn8wbgbl8zs8rvr0000gn/T/tmpk6kMIp/sr
/Users/christophemorisset/anaconda/lib/python2.7/site-packages/numpy/core/include/n
 #warning "Using deprecated NumPy API, disable it by " \
  ^
compiling Fortran sources
Fortran f77 compiler: gfortran -Wall -g -ffixed-form -fno-second-underscore -m64 -f
Fortran f90 compiler: /usr/local/bin/gfortran -Wall -g -fno-second-underscore -m64
Fortran fix compiler: /usr/local/bin/gfortran -Wall -g -ffixed-form -fno-second-und
compile options: '-I/var/folders/bb/jg97y_ln7cn8wbgbl8zs8rvr0000gn/T/tmpk6kMIp/src.
gfortran:f77: dcumsum.f
/usr/local/bin/gfortran -Wall -g -m64 -Wall -g -undefined dynamic_lookup -bundle /v
Removing build directory /var/folders/bb/jg97y_ln7cn8wbgbl8zs8rvr0000gn/T/tmpk6kMIp
```

```
In [7]: # Importing the function as if it where a python package
        import dcumsum
```

```
In [8]: a = np.linspace(10,100, 1000)
```

```
In [9]: %timeit py_dcumsum(a)

1000 loops, best of 3: 295 µs per loop


In [10]: %timeit numpy_cumsum(a)

The slowest run took 208.30 times longer than the fastest. This could mean that an
100000 loops, best of 3: 4.04 µs per loop


In [11]: %timeit a.cumsum()

The slowest run took 18.30 times longer than the fastest. This could mean that an i
100000 loops, best of 3: 3.44 µs per loop


In [12]: %timeit dcumsum.dcumsum(a)

The slowest run took 5.84 times longer than the fastest. This could mean that an in
1000000 loops, best of 3: 1.39 µs per loop
```

The Fortran call is still 2 times faster than the numpy object method, and 10 times faster than the loop.

1.0.1 cython

```
In [13]: # Integration of a function by summing values
         def f(x):
            return x**2 - x
         def integrate_f(a, b, N):
             s  = 0
             dx = float(b - a) / N
             for i in range(N):
                s += f(a + i*dx)
             return s*dx

In [16]: # To allow the use of %%cython
         %load_ext Cython

In [17]: %%cython
         cdef double cy_f(x):
            return x**2 - x
         def cy_integrate_f(double a, double b, int N):
             cdef int i
             cdef double s, dx
             s  = 0
             dx = (b - a) / N
             for i in range(N):
                 s += cy_f(a + i*dx)
             return s*dx
```

```
In [18]: %timeit integrate_f(0,3,10^3)
```

```
The slowest run took 6.90 times longer than the fastest. This could mean that an in
100000 loops, best of 3: 3.21 µs per loop
```

```
In [19]: # Really faster!!!
         %timeit cy_integrate_f(0,3,10^3)
```

```
The slowest run took 10.89 times longer than the fastest. This could mean that an i
1000000 loops, best of 3: 744 ns per loop
```

```
In [20]: # Same values are obtain (hopefully!)
         print integrate_f(0,3,10^3), cy_integrate_f(0,3,10^3)
```

```
3.55555555556 3.55555555556
```

Let's now compare when doing havy matrix operations, taken from http://technicaldiscovery.blogspot.mx/2011/06/speeding-up-python-numpy-cython-and.html

```
In [21]: dx = 0.1
         dy = 0.1
         dx2 = dx*dx
         dy2 = dy*dy

         # The looping way
         def py_update(u):
             nx, ny = u.shape
             for i in xrange(1,nx-1):
                 for j in xrange(1, ny-1):
                     u[i,j] = ((u[i+1, j] + u[i-1, j]) * dy2 +
                               (u[i, j+1] + u[i, j-1]) * dx2) / (2*(dx2+dy2))

         def calc(N, Niter=100, func=py_update, args=()):
             u = np.zeros([N, N])
             u[0] = 1
             for i in range(Niter):
                 func(u,*args)
             return u
```

```
In [22]: %timeit calc(20)
```

```
10 loops, best of 3: 38.1 ms per loop
```

```
In [23]: # The numpy way
         def num_update(u):
             u[1:-1,1:-1] = ((u[2:,1:-1]+u[:-2,1:-1])*dy2 +
                             (u[1:-1,2:] + u[1:-1,:-2])*dx2) / (2*(dx2+dy2))
```

```
In [24]: %timeit calc(20, func=num_update)

1000 loops, best of 3: 1.05 ms per loop


In [25]: %%cython
         cimport numpy as np

         def cy_update(np.ndarray[double, ndim=2] u, double dx2, double dy2):
             cdef unsigned int i, j
             for i in xrange(1,u.shape[0]-1):
                 for j in xrange(1, u.shape[1]-1):
                     u[i,j] = ((u[i+1, j] + u[i-1, j]) * dy2 +
                               (u[i, j+1] + u[i, j-1]) * dx2) / (2*(dx2+dy2))

In [26]: %timeit calc(20, func=cy_update, args=(dx2, dy2))

1000 loops, best of 3: 354 µs per loop
```