Cells on a Sphere Simulation

Eduardo Pavinato Olimpio

July 30, 2015

1 Movement Simulations

In order to understand and get insights on the movement of the organoids, we performed some simulations of cells on a sphere. The cells are treated as self-propelled particles (SPP) in an onverdamped regime. The basis for the simulation is found in the reference [1]. All the code is found in

1.1 Compiled FORTRAN in Python

Before explaining the simulations, we have made use of some compiled FORTRAN code in Python. This has the advantage of (i) speed and (ii) a famous Delaunay triangulation for the sphere is written in FORTRAN. This is achieved using a numpy modeule called `f2py` (`http://docs.scipy.org/doc/numpy-dev/f2py/`). To make it work on Windows, you have to follow the instruction below.

1.1.1 Use Pure Python (no distribution)

Sorry, but I could only make it work on the pure Python installation (`https://www.python.org/downloads/`). If you have any distribution (e.g. Anaconda) you will need to uninstall it to avoid conflict. It can be possible to make it work with Anaconda with some tuning in the Windows register, but it would take too much time. Moreover, I found it much more easy if you use the 32-bit Python interpreter. In Linux things run more smoothly.

When you switch back to the normal Python, you will need to get the binaries of the packages in `http://www.lfd.uci.edu/~gohlke/pythonlibs/`. Then use the command line and go to the forder where the binaries are and type:

```
$PYTHON_PATH$\Scripts\pip.exe install <name_of_the_binary>
```

where we assume that the installation of Python is in the folder `$PYTHON_PATH$`. Even in a 64-bit system I recommend the use of the 32-bit Python interpreter in case of compiled code. In this case, you will need the win32 version of the wheel files in the website given above.

Required Packages for the simulation:

1. numpy
2. scipy
3. matplotlib

1.1.2 Install a FORTRAN COMPILER (I used G95)

You will need a Fortran compiler native for Windows. I used G95 which worked nicely. The installation procedure is quite simple and you can obtain the compiler at `http://www.fortran.com/the-fortran-company-homepage/whats-new/g95-windows-download/`.

1.1.3 Install MinGW

Now you can download the nice distribution of MinGW available in `https://github.com/develersrl/gccwinbinaries`. This installation already take care of all the environment variable that needs to be adjusted. MinGW essentially installs the GCC libraries for compiling the code using `f2py`. You can find more information about MinGW in `http://www.mingw.org/`.

1.2 Test if it works

Write the below fortran code in a file `foo.f95`.

```
subroutine foo(n, a)
  integer, intent(in)  :: n
  real,    intent(out) :: a(n,n)
  do i=1,n
    a(i,i) = 1.0
  end do
end subroutine foo
```

Now run the command:

```
f2py foo.f95 -m foo -h foo.pyf
```

if it gives no error[1] then run:

```
f2py -c --fcompiler=g95 --compiler=mingw32 foo.pyf foo.f95
```

If the compilation gives no error, you can use your library in Python. To do it, write the following code:

```
from numpy import *
import foo

print(foo.foo.__doc__)  # shows python interface generated by f2py
a = foo.foo(4)
print(a)
```

When you run this you will see how the foo function is called and the matrix a (an identity matrix).

References

[1] Rastko Sknepnek and Silke Henkes. Active swarms on a sphere. *Physical Review E*, 91, 2014.

[1] Sometimes it gives an error related to the `f2py` script path. This can be easily fixed by editing the first line of `$PYTHON_PATH$/Scripts/f2py.py` in order to comply with `$PYTHON_PATH$`