M3C 2018 Homework 3

Due date/time: 29/11/2018, 23:59 GMT

Getting Started: If needed, update your local copy of the course repo (*sync* and *pull*). The homework/hw3/ directory contains the files, *hw3_template.py*, *hw3_template.f90*, and *hw3_main.f90*. Make a new directory outside of the course repo, and place *dev* copies of the template files named in this new directory. Ultimately, the final files that you submit should be titled *hw3.py* and *hw3.f90*. Browse through the three files provided; the template files contain a number of function/subroutine headers, and you will have to add code as described below. First, however, you should *add your name and 8-digit college id to the docstring/comment at the top of each file.*

In this assignment, we will consider a modified version of the tribe competition model implemented in HW1. Let P_{C0} be the probability of a village being a C in the proceeding generation as defined in HW1. Here, we modify the model so that P_{C0} is replaced with $P_C = \gamma P_{C0} + \frac{(1-\gamma)}{2}$ where γ is a new model parameter with $0 \leq \gamma \leq 1$. The probability of a village being an M in the next generation is then $P_M = 1 - P_C$. The initial condition has also been modified; now only the central village is an M while all other villages are Cs. Aside from these changes, the model remains the same as in M3C 2018 Homework 1. You have been provided with Python and Fortran routines named *simulate2* and *simulate2_f90* that implement *m* trials of this modified model.

Note: As you work through the tasks below, you may add module variables and additional subroutines and functions as needed. Do not remove module variables or modify input/output of existing routines without first asking for permission (unless comments in the code explicitly allow for such modifications).

1.(35 pts) Develop the Fortran subroutine *simulate2_omp* as a parallelized version of *simulate2_f90* The routine should output the final state matrix, *s*, for all *m* trials as well as the fraction of cooperators, f_c, at all *nt+1* time steps averaged over the *m* trials. The subroutine should be carefully parallelized with OpenMP. You should avoid generating nested parallel regions, but otherwise, your OpenMP implementation should be both sensible and thorough. The number of threads used within any parallel regions should correspond to the module variable, *numthreads*. Add a comment below the subroutine header concisely explaining your approach to parallelization.

Note 1: *simulate2_f90* is an almost direct "translation" of *simulate2* into Fortran. As a result, it uses memory inefficiently, particularly as *m* becomes large. Your parallelized code should avoid similar inefficiencies where possible.

Note 2: *simulate2_f90* and *simulate2_omp* should produce the same output if the same sequences of random numbers are used in the two routines.

2. (35 pts) Complete the function *performance* in *hw3_dev.py* so that it analyzes the parallel performance and speedup of *simulate2_omp* and the relative performances of the Python, Fortran, and Fortran+OpenMP implementations of the model. You may fix $b = 1.1, e = 0.01, \gamma = 0.95$ for this question. Your function should make one or more figures illustrating the most important qualitative trends. Clearly and concisely discuss and explain these trends in the docstring of the function. You are not required to run calculations with more than 2 threads.

3. (15 pts) Investigate the influence of the 'new' model parameter, γ. Fix *N=51* and *e=0.01* and constrain your study to $1 < b \leq 1.5$, $0.8 \leq \gamma \leq 1$. Complete the function *analyze* in *hw3_dev.py* so that it creates 1 or more figures illustrating the most important trends that you find. Explain the displayed trends in the function docstring. *There is no penalty for using loops in Python for this question.*

4. (15 pts) Complete the function *visualize* in *hw3_dev.py* so that it generates an animation illustrating the evolution of village affiliations for your choice of model parameters. It is sufficient to consider the case, *N=21*, but feel free to select larger sizes. Submit your animation file with the filename *hw3movie.xxx* with *xxx* replaced as appropriate (e.g. *mp4*).

See lab 7 and Installing ffmpeg *for more information on creating mp4 movies in Python.*

Notes:

1. All figures created by your code should be well-made and properly labeled. The title of each figure should include your name and the name of the function which created it. **Submit figures files corresponding to all of the figures generated by the *performance* and *analyze* functions with your codes.** If you are submitting more than 10 figures in total, please think carefully about presenting your results more concisely.

2. It is expected that your python functions will take several minutes to run. It is suggested that you start out with smaller problem sizes and then move to larger sizes as your code develops and as you prepare to generate your final figures. If a function is taking

more than an hour to run, check with the instructor to see if this is ok.

3. Your Fortran codes should run to double-precision accuracy.

4. The code in the *name==main* section of your python script should call your *performance* and *analyze* functions and generate the figures you are submitting.

5. You may use any python routines in *numpy*, *scipy*, *matplotlib*, and *time*. Check first if you would like to utilize other modules.

Submitting the assignment

To submit the assignment for assessment, go to the course blackboard page, and click on the "Assignments" link on the left-hand side of the page, and then click on Homework 3. Click on "Write Submission" and add the statement: "This is all my own unaided work unless stated otherwise."

Click on "Browse My Computer" and upload your final files, *hw3.f90*, *hw3.py*, *hw3xx.png*, and *hw3movie.xxx*. Finally, click "Submit".